Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften (Dr. rer. pol.)
von der Fakultät für Wirtschaftswissenschaften
der Universität Fridericiana zu Karlsruhe
genehmigte Dissertation.

Creating Ontology-based Metadata by Annotation for the Semantic Web

von

Dipl.-Ing.(FH), Dipl.-Inf.wiss. Siegfried Handschuh

Tag der mündlichen Prüfung: 16. Februar 2005
Referent: Prof. Dr. Rudi Studer
Korreferent: Prof. Dr. Christof Weinhardt

To my family.

Acknowledgements

I would like to thank people that supported me during the process of researching and writing this thesis. First of all, I would like to express my gratitude to my advisor, Prof. Dr. Rudi Studer, for his support, patience, and encouragement throughout my PhD studies at the Institute AIFB at the University of Karlsruhe.

I am also grateful to Prof. Dr. Christof Weinhardt on my dissertation committee, and to Prof. Dr. Andreas Oberweis, who served on the examination committee.

My work very much profited from working with Prof. Dr. Steffen Staab. I learned a lot from him, he inspiried me with his ideas and he significantly improved my scientific skills, *e.g.* by teaching me how to successfully write papers. He and Prof. Dr. Gerd Stumme gave me invaluable advice on organizing research workshops.

My thanks also goes to all my colleagues in Karlsruhe at the Institute AIFB, the FZI and Ontoprise GmbH for providing a very fruitful and stimulating working atmosphere. In particular to Philipp Cimiano, Nenad and Ljiljana Stojanovic, Sudhir Agarwal and Alexander Mädche.

I'm especially indebted to Dr. Stefan Decker, for taking me on board his Onto-Agents project and the DARPA DAML program, which funded my work on semantic annotation.

Tanja Sollazzo, Guylaine Kepeden, Wolf Winkler, Mika Maier-Collin and Kai Kühn who wrote their diploma theses under my supervision, contributed to the annotation framework. Furthermore I would like to thank Leo Meyer and Matthias Braun for their implementation work.

My gratitude goes to Dr. Simon Beck for improving my time management skills in the final stage of my thesis writing, and my sister-in-law, Caroline for proof-reading drafts of this thesis - any remaining errors are of course mine.

I thank my friends Esther Stäheli and Sven Ruby for their devotion and putting up with me over a number of years; and Jorge Gonzalez for his advice and Julien Tane for the lively discussions.

My family, my parents Philomena and Karl receive my deepest gratitude and love for the many years of support and for always believing in me. Last, but not least, I would like to thank my wife Patricia for her patience, love and understanding and our adorable daughter Ella for being a part of our lives from now on.

Contents

List of Figures

List of Tables

Part I.

Foundations

"... whoever controls the vocabulary, controls the knowledge"
— George Orwell

1. Introduction

1.1. Motivation & Problem Description

Like all truly great ideas, Tim Berners-Lee's principle idea of the *Semantic Web* may be easily summarized: When computers not only retrieve, but also understand what data is available on the Web, we will have a new kind of Web and new types of intelligent applications in the Web. In the foreseeable future, however, machines will be too dumb to understand what people have put on the Web. Therefore, let us put *computer-understandable data* next to human-understandable data. Then, the computers will be smarter.

In order to make the vision of the Semantic Web come true, we need a number of building blocks, some of them elaborated on in recent writings [Fensel *et al.*, 2002; Hendler and Horrocks, 2002; Fensel *et al.*, 2003; DeRoure and Iyengar, 2002; Chen *et al.*, 2003; Constantopoulos *et al.*, 2000; Staab *et al.*, 2001a; Frank *et al.*, 2002a]. For instance, we need standardized languages to describe semantic data, i.e. data that is as computer-understandable as it is semantically self-describing, and we need programmes and protocols to actually exchange and understand semantic data. As a *primus inter pares*, however, we need semantic data.

The work in this thesis is about providing semantic data, a process often referred to as "Semantic Annotation" because it frequently involves the embellishment of existing data, e.g. plain text, that is only understandable for the human with semantic metadata that describes, e.g., the text. Understandably, Semantic Annotation is now one of the core challenges for building the Semantic Web.

Since Semantic Annotation is the key notion of this thesis we shall define it in more detail:

Definition 1.1.1
The term Semantic Annotation describes a process as well as the outcome of the process[1]*. Hence it describes i) the process of addition of semantic data or metadata to the content given an agreed ontology and ii) it describes the semantic data or metadata itself as a result of this process.*

[1] cf. with term "drawing"

1.2. Research Questions

The creation of Semantic Annotation raises a series of research questions. In the following the most important ones are outlined, which will be addressed by the work described in this thesis. Initially, the following question must be asked:

- *How can the creation of Semantic Annotation be supported? How can the complexity of creating Semantic Annotation be reduced?*

Thus the general question of how to create Semantic Annotation is investigated and possible solutions are described in this thesis.

The existing Web can be seen as a distributed information system consisting of HTML pages. The first step is the extension of purely syntactic information, e.g. HTML documents, with semantics. The challenge here is a knowledge capturing problem:

- *How may one turn existing syntactic resources into interlinked knowledge structures that represent the underlying information?*

However, in order to provide metadata about the content of a Web page, the author must first create the content and secondly annotate the content in an additional, *a-posteriori*, annotation step. The question here is:

- *How to combine the authoring of a Web page and the creation of Semantic Annotation describing its content?*

Nevertheless, providing plenty of semantic metadata by annotation, i.e. conceptual markup of text passages, is a laborious task. Therefore:

- *How can one increase the efficiency of the production of metadata? How can the annotation process be automated or semi-automated?*

Today, a large percentage of Web pages are dynamic. Annotating every single dynamic Web page generated from a database would be tedious. Here, the question is:

- *How to annotate the database so that it is reusable for site-specific Semantic Web purposes?*

Evaluating metadata creation or more general knowledge engineering is not that well researched. It lacks generic methods and concrete evaluation and comparison measures (compared to recall/precision in information retrieval) that may be applied for evaluating Semantic Annotation, (viz. ontology population). Furthermore, the measurement of inter-annotator agreement can be used to compare the human performance with semi-automatic approaches. Therefore, the two questions of

- *How can the metadata creation results between humans be evaluated? How can the inter-annotator agreement be formally compared?*
- *How do humans perform in metadata creation compared to information extraction techniques?*

will be addressed and results will be provided in the evaluation chapter of this thesis.

1.3. Approaches

The core of this thesis is the development of an annotation framework viz. CREAM to support the creation of metadata (CREAM — CREAting Metadata for the Semantic Web). CREAM comprises methods for:

- **Manual annotation**: The transformation of existing syntactic resources (*viz.* textual documents) into interlinked knowledge structures that represent relevant underlying information (Section 4.6.1 and 4.6.2).
- **Authoring of documents**: In addition to the annotation of existing documents the authoring mode lets authors create metadata — almost for free — while putting together the content of a document (Section 4.6.3).
- **Semi-automatic annotation**: Efficient semi-automatic annotation based on information extraction that is trained to handle structurally and/or linguistically similar documents (Section 5.1). Another approach for semi-automatic annotation is the self-annotating Web. The principle idea of the self-annotating Web is that it uses globally available Web data and structure to semantically annotate – or at least facilitate annotation of – local resources (Section 5.2).
- **Deep annotation**: Deep annotation results in a semantic mapping to the underlying database if the database owner cooperates in the Semantic Web and allows for direct access to the database (Section 6).

1.4. Reader's Guide

Every chapter is preceded with a brief introductory paragraph which explains how the work presented in the section fits in the overall structure of the thesis. The thesis is divided into four main parts, Fundamentals (I), Metadata for the Semantic Web (II), Evaluation (III) and Related Work & Outlook (IV). These four main parts are organized as follows:

Part I — Fundamentals:

- **Chapter 2** introduces metadata and ontology languages. It introduces metadata languages, i.e. the eXtensible Markup Language (XML), which can be used for modeling structures, the XPointer schema for addressing parts of a document, the Resource Description Framework (RDF), to express statements, RDF Schema to define a vocabulary for RDF. Finally, it presents OWL as a fully-fledged stack of ontology languages.
- **Chapter 3** is about the vision of the Semantic Web. It shows the Semantic Web Layer Cake based on the metadata languages introduced before. It presents how to apply these technologies in order to provide Semantic Annotation. It presents different points of view for the ontology and metadata creation, viz. the *knowledge meta process and knowledge process* as well as the *information foodchain for the Semantic Web*. It defines terms that are used with regard to metadata creation, it shows the different semantics of Semantic Annotation, gives an insight into first steps towards a methodology for Semantic Annotation and presents a layering of Annotation. It concludes with a comprehensive annotation model.

Part II — Metadata for the Semantic Web:

- **Chapter 4** presents a comprehensive annotation and authoring framework, *CREAM*, that allows for the creation of semantic metadata about static and dynamic Web pages, i.e. for Semantic Annotation of the *Shallow and Deep Web*. CREAM supports the manual and the semi-automatic annotation of static Web pages, the authoring of new Web pages with the simultaneous creation of metadata, and the deep annotation of Web pages defined dynamically by database queries. This chapter first describes case studies from which we took a major part of our experiences for guiding the development of CREAM. We describe there some of the requirements for Semantic Annotation in detail that we derived from the case studies. We derive the design and the modules of CREAM from the requirements elaborated previously. We specify how the meta ontology may modularize

the ontology description from the way the ontology is used in CREAM. Finally we explain the major modes of interaction within the annotation tool *OntoMat*, our reference implementation of CREAM.

- **Chapter 5** describes the techniques developed for semi-automatic annotation. These techniques are realized extensions, viz. plugins, of our annotation framework. The first extension – S-CREAM (Semi-automatic CREAtion of Metadata) – allows for creation of metadata and is trainable for a specific domain. The implementation of S-CREAM in the tool OntoMat supports the semi-automatic annotation of Web pages. This semi-automatic annotation is based on the information extraction component Amilcare. OntoMat extracts with the help of Amilcare knowledge structures from Web pages through the use of knowledge extraction rules. These rules are the result of a learning-cycle based on previously annotated pages. The second extension – PANKOW (Pattern-based Annotation through Knowledge on the Web), focuses on a method that combines the use of linguistic patterns to identify instances with the use of the WWW as a large corpus via a search engine.

- **Chapter 6** portrays an extension of the CREAM framework for metadata creation when Web pages are generated from a database *and* the database owner is cooperatively participating in the Semantic Web. In order to create metadata, the framework combines the presentation layer with the data description layer — in contrast to "conventional" annotation, which remains at the presentation layer. Therefore, we refer to the framework as *deep annotation*.[2]. This chapter describes the building blocks for deep annotation. Firstly, it elaborates the use cases. Then it continues with a description of the overall process, where we find the major requirements that must be provided: i) server side markup, ii) utilization of the information by annotation, iii) creation and exploitation of mapping and iv) query of the serving database.

- In **Chapter 7** we demonstrate the flexibility of the annotation framework by using it for **linguistic annotation** and **service annotation**. The first Section shows how to exploit the basic framework easily for linguistic annotation, while the second section shows how to apply the deep annotation framework also to annotation and composition of Web services.

Part III — Evaluation:

- **Chapter 8** deals with the evaluation of hands-on-experiences exploring an annotation experiment with human subjects. It describes an empirical

[2]The term "deep annotation" was coined by Carol Goble in the Semantic Web Workshop of WWW 2002.

evaluation study of ontology-based semantic annotation. Based on a given ontology and a set of documents, we analyze in this chapter inter-annotator-agreement between different humans. The evaluation study uses several standard and two original measures. The latter take into account a notion of sliding agreements between metadata – exploiting semantic background knowledge provided by the ontology.

- **Chapter 9** investigates the approach of semi-automatic annotation. Based on the evaluation study for manual annotation introduced in the previous Chapter, we adapt the measures to evaluate the similarity between the annotation produced by different users applying the process of manual annotation in comparison to applying the semi-automatic annotation process. Hence, the evaluation confirms that semi-automatic annotation is faster and produces more homogeneous metadata than manual annotation. The chapter describes the principal methods of evaluation and presents the results and a discussion.

Part IV — Related Work & Outlook:

- **Chapter 10** deals with the difficult task of giving an overview of related work on Semantic Annotation. Semantic Annotation as we have presented it in this thesis is a cross-sectional enterprise. Much work in a number of disciplines, like knowledge acquisition, computational linguistics, information retrieval, information science, information integration, databases, has researched and applied techniques for solving part of the overall problem of Semantic Annotation for the Semantic Web. This chapter gives an overview of related work from a number of different communities.

- **Chapter 11** concludes with a short summary of the methodological and technical results and sketches ideas for further research. It explains the main contributions of the work described in this thesis and lists a number of insights gained from doing this research. Additionally, unsolved questions and further research issues are defined.

2. Metadata and Ontology Languages

This Chapter provides a basic introduction to metadata (Section 2.1) and ontology languages. In the following sections different metadata languages are introduced, i.e. the eXtensible Markup Language (XML) (Section 2.2), which can be used for modeling structures, the XPointer Schema (Section 2.3) for addressing parts of a document, the Resource Description Framework (RDF) (Section 2.4) to express statements and RDF Schema (Section 2.4.2) to define a vocabulary for RDF. Furthermore, a formal definition of an Ontology (Section 2.5.1) is presented as used in this thesis. Finally, OWL (Section 2.6) is introduced as a fully-fledged stack of ontology languages.

2.1. Metadata

One way of extending the existing structure of the WWW with semantics and thus enabling the Semantic Web is by adding metadata. Metadata is a combined word from "meta" (Greek: between, after, later) and "data" and stands for "data about data", i.e. data that describes other data, viz. data that identifies and describes an information object. In the Semantic Web environment, metadata is understood as "data describing Web resources" [Manola and Miller, 2004]. The Digital Library Federation (DLF)[1], a coalition of 15 major research libraries in the USA, defines three types of metadata which can apply to objects in a digital library:

- **Descriptive Metadata**: This metadata describes the information object, relating to what the object contains or is about.
- **Administrative Metadata**: It indicates the who, what, where and how-aspects associated with the object's creation and preservation.
- **Structural Metadata**: It ties each object to others to make up logical units (for example, information that relates individual images of pages from a book to the others that make up the book itself).

[1] http://http://www.diglib.org/dlfhomepage.htm

In general, only descriptive metadata is visible to the users of a system, who search and browse it to find and assess the value of items in the collection. Administrative metadata is usually only used by those who maintain the collection, and structural metadata is generally used by the interface which compiles individual digital objects into more meaningful units (such a journal volumes).

In addition, metadata can be classified along the following dimensions:

- **Formality**: With regard to formality, metadata may range from very informal descriptions of documents, e.g. free text summaries of books, up to completely formal descriptions.
- **Containment**: In the context of containment, described data can be included in metadata, e.g. the describing tags are added to the existing code, while "others may be stored completely independently from the document they describe, such as a bibliography database that classifies the documents it refers to, but does not contain them".
- **Capturing of Information Content** (c.f. [Kashyap and Sheth, 1996]): Kashyap and Seth classify:
  - **Content Independent Metadata**: Examples of this type of metadata are *location*, *modification-date*.
  - **Content Dependent Metadata**: Examples are *size* of a document, *max-colors* of an image. A categorization of content dependent metadata is:
    - **Direct Content-based Metadata**: A popular example of this is a full-text index based on the text of the documents.
    - **Content-descriptive Metadata**: This type describes the content without direct utilization of the content of an information object. An example of this type is textual annotation describing the contents of an image. This type of metadata comes in two varieties:
      - **Domain-Independent Metadata**: Examples are HTML/SGML document type definitions.
      - **Domain Specific Metadata**: This metadata is specific to the application or subject domain of information. Examples of this metadata are *relief*, *land-cover* from the GIS domain and *area*, population from the Census domain.

The metadata under discussion in this thesis is, according to the above classification, *descriptive*, *formal*, *domain specific* metadata, based on ontologies for documents available on the Web. The metadata may be *contained* in the Web document or stored externally.

2.1.1. Metadata Standards

A well known use case for metadata is a library, where classification systems such as the Dewey Decimal Classification[2] or the machine readable MARC standard[3] enable easy access to literature. The MARC formats are standard for the representation and communication of bibliographic and related information in machine-readable form.

An example for a widely known and used metadata standard is Dublin Core[4]. It was invented for the format and content based processing of network resources, mainly the WWW. Its roots are in the online library community, its first workshop was held in 1995 by the OCLC (Online Computer Library Center) and the NCSA (National Center for Supercomputing Applications) in Dublin, Ohio, giving the initiative its name. Dublin Core comprises fifteen core elements, which can be used for the metatags in a HTML document, see e.g. [Kunze, 1999] for a description of the use of HTML with Dublin Core. Using Dublin Core it is possible to do a content based document classification, as for instance known in the classical library process (e.g. identification of author, title, subject, language, source etc.). The Dublin Core elements can be grouped into three element classes, describing content, intellectual property and instantiation [Hillmann, 2001] (see Table A.1).

Other types of Metadata Schema are MPEG-7[5], EAS (Encoded Archival Description)[6], IEEE LOM (Learning Objects Metadata)[7], ADL Scorm[8] and RSS (RDF Site Summary)[9].

2.1.2. Metalanguages

Metadata need to be expressed in a standardized way, i.e., the syntax of metadata is given by metalanguages. Metalanguages, languages for describing languages, can be used either as "underlying representation languages for ontologies" [Fensel, 2001, abstract] or as representation languages themselves. The representation languages, also called ontology languages, represent the resources, viz. the knowledge available on the Web, in the taxonomy of an ontology (explained later in Section 2.5).

[2] http://www.oclc.org/dewey/
[3] http://www.loc.gov/marc/gg
[4] http://dublincore.org
[5] http://xml.coverpages.org/mpeg7.html
[6] http://www.loc.gov/ead
[7] http://ltsc.ieee.org/wg12/
[8] http://www.adlnet.org/
[9] http://web.resource.org/rss

2.2. XML

In 1989, Tim Berners-Lee and Vinton Cerf created with the Hypertext Markup Language (HTML) the basic technology for the current WWW. The success of the Web was brought about by the simplicity of HTML for describing and rendering documents in the WWW. However, the disadvantages of HTML (cf. [Fensel, 2001, p.52]) became obvious by the enormous growth of the Web:

- Missing semantics inhibits automatic information retrieval and processing. The lack of structured data leads to a strong limitation of data exchange and makes it difficult for software agents to provide services.
- A customization of tags for different applications is not possible. Therefore, it can be stated that HTML may be regarded more as a "layout representation language" rather than a technology to support knowledge management, communication, and electronic commerce.

To overcome these shortcomings of HTML, the W3C introduced an additional standard for defining the data structure of Web documents: the eXtensible Markup Language (XML). XML, a "child" of the more complex Standard Generalized Markup Language (SGML), is a tag-based language for describing tree structures with a linear syntax. Providing seven different means for presenting information [Bray *et al.*, 2004], the metalanguage has the following advantages:

- XML tags define the structure of the data.
- XML provides arbitrary trees (graphs) as data structures.
- XML allows the definition of application-specific tags and can therefore be customized for the exchange of specific data structures.

Listing 2.1 shows an example of an XML document. It defines application-specific tags such as <person> and <firstname> ordered in an hierarchical structure. Note, that the structure of an XML document is not the same as the structure it represents. The object model of a document representing a business object is the model for a document not for a business object.

In order to model the arbitrary trees, various XML schemes exist which define a grammar for XML documents, e.g. Document Type Definition (DTD) and XML Schema[10]. Listing 2.2 shows the DTD and Listing 2.3 the XML Schema for the example in Listing 2.1. The XML schemes resemble ontologies because they define tags, their nesting, and attributes for tags [Fensel, 2001, p.72]. The

[10] http://www.w3.org/XML/Schema

schemes, however, lack any notion of inheritance, define the order in which tags appear in a document and can not express hierarchies in the manner required by ontologies.

Another strength of XML is the use of eXtensible Style Language[11] (XSL) , which defines how a browser should render XML documents in a more expressive way than Cascading Style Sheets[12] (CSSs) for HTML documents. That is to say, XSL can be used to create different pages from the same data sources and thus to "realize dynamically changing pages according to user preferences or contexts" [Fensel, 2001, p.57]. This is especially important for one-to-one marketing strategies in the WWW. However, XSL has more powerful uses than merely the creation of layouts. The disadvantage that XML does not provide standard DTDs, i.e. that different users can define different tags like <email> and <mail> for the same concept, is overcome by XSL. XSL can be used to translate XML documents using DTD1 into XML documents using DTD2 and thus facilitates the exchange of data between different users.

Concerning the weaknesses of XML, the use of different DTDs solved by mapping has been already mentioned. Another problem is inherent to XML: it allows users to create their own tags, e.g. <person>, which seems to carry some semantics. From a computational perspective, however, "these tags carry as much semantics as a tag like <H1> in HTML [Decker, 2004]. A computer does not by itself attempt to interpret the meaning of labels [Collier, 2001, p.2]. For example, it has no idea what a person is and how the concept "person" is related to a concept "Web page", e.g. the person is the creator of a Web page. Thus, it can be concluded that XML enables every user to enrich their Web sites with self-defining, invisible, machine-readable tags, but XML structures only the information and does not tell us anything about what the structure means.

```
<?xml version=1.0" encoding="ISO-8859-1"?>
<ontoweb>
  <Person>
    <firstname>Steffen</firstname>
    <lastname>Staab</lastname>
    <country>Germany</country>
  </Person>
  <Person>
    <firstname>Stefan</firstname>
    <lastname>Decker</lastname>
    <country>Ireland</country>
  </Person>
</ontoweb>
```

Listing 2.1: Example for XML

[11] http://www.w3.org/Style/XSL/
[12] http://www.w3.org/Style/CSS/

```
<!DOCTYPE ontoweb[
  <!Element ontoweb (Person+)>
  <!Element Person (firstname, lastname, country?)>
  <!Element firstname (#PCDATA)>
  <!Element lastname (#PCDATA)>
  <!Element country (#PCDATA)>
]>
```

Listing 2.2: Example for XML DTD

```
<schema>
  <element name="ontoweb">
  <complexType>
  <element name="Person" minOccurs="1" maxOccurs="unbounded">
    <sequence>
      <element name="firstname" type="string">
      <element name="lastname" type ="string">
      <element name="country" type="string" minOccurs="0">
    </sequence>
  </complexType>
  </element>
</schema>
```

Listing 2.3: Example for XML Schema

2.3. XML Pointer Language (XPointer)

As mentioned above in Section 2.1 metadata can be contained with the data or stored completely independently from the data, e.g. a Web document. For the latter one would need a reference from the metadata to the document. Hence, we describe here the parts of the XPointer language that are useful for the Semantic Annotation of a document, viz. the addressing of string and nodes.

XPointers are able to reference a defined subset of a XML-Document: Single Points, String, Nodes and arbitrary combinations of these.

The XPointer [DeRose *et al.*, 2001] design is factored into a basic framework *XPointer Framework* [Grosso *et al.*, 2002] and three additional schemes: *XPointer element()* [Grosso *et al.*, 2003], for addressing elements by their positions in the document tree, *XPointer xmlns()* [DeRose *et al.*, 2003], for binding namespace prefixes to namespace name and *XPointer xpointer()* [DeRose *et al.*, 2002], for full XPath-based addressing.

The framework and the schemes define the syntax and the semantic of XPointers. They extend the XPath recommendation (see next section) that describes a mechanism to address a node in an XML-Document.

An *XPointer* identifies fragments of an XML document (e.g. external parsed

entities[13]). The target XML document is called *resource*. The identified fragment is called *subresource*. A subresource is the content of a location-set. A location-set consists of a node (XPath node), points, and ranges. Possible nodes are root node, element node, attribute node, processor node, comment node, text node and namespace node. Ranges are a part of a document between a start and an end point. Ranges can cover several text nodes. An *XPointer processor* is a software component that interprets XPointers and localizes the corresponding fragment.

2.3.1. XPath Basics

The XPath standard is extensive and enables unique expressions to reference nodes. For a complete description of this standard see [Clark and DeRose, 1999]. To demonstrate the simplest form of addressing we take the following example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ontoweb>
  <Person>
    <firstname>Steffen</firstname>
    <lastname>Staab</lastname>
    <country>Germany</country>
  </Person>
  <Person>
    <firstname>Stefan</firstname>
    <lastname>Decker</lastname>
    <country>Ireland</country>
  </Person>
</ontoweb>
```

The simplest form of an XPath resembles a path expression from an operating system. The sequence of nodes along the hierarchy to the target node is separated by slashes. The expression

```
/ontoweb/person[2]/country
```

selects the element of type *country* with the content *Ireland*. The example shows that one references with square brackets *[ ]* the specific contextual position of an element. Here, the second <country> element is referenced. If a location path begins with a double slash, all nodes, which comply with the stated conditions, are located to their right. Hence, the expression

```
//person/country
```

references all <country> elements that are child elements of a <person> element. The reader may note, that an XPath expression might also refer to more than one node.

[13] An external parsed entity need not be well-formed. In particular, it may lack a single root element.

2.3.2. XPointer Standards

The syntactic scheme of an XPointer is:

```
URI of XML document + "#" + XPointer
```

The XPointer can consist of several parts, which are separated by white spaces. An XPointer part follows a syntax scheme that is defined in: *XPointer framework*, *XPointer element() scheme* and *XPointer xpointer() scheme*. The basic forms of these schemes are as follows:

All XPointer processors must be able to support the **XPointer framework**. However, it is optional that the processor have the facility to support the other schemes.

The XPointer frameworks define shorthand pointers. They have the basic form:

```
URI of XML document + "#" + Name
```

Name refers to an element, that has the ID attribute of the value *Name*. This attribute has to be defined in the DTD or XML schema corresponding to the XML document. Given is the following example document:

```
<ontoweb>
  <person id="6">Steffen</person>
  <person id="57">Rudi</person>
</ontoweb>
```

When the attribute *id* is declared in the DTD or XML schema, then the following expression

```
http://www.ontoweb.org/personlist.xml#57
```

is referencing the second *person*-Element.

The **XPointer element() scheme** defines a simple syntax to access elements via the tree structure of the document. For example:

```
http://www.ontoweb.org/personlist.xml#element(/1/5/3)
```

locates the third child element of the fifth child element of the first root node, whereas external entities can have several root nodes.

The **XPointer xpointer() scheme** provides the most comprehensive possibilities for referencing parts of an XML document. One can use all XPath expressions to identify a node. Additionally, the scheme defines concepts to address points and ranges inside nodes and beyond the border of nodes (see Section 2.3.3). Example:

```
...#XPointer(/person/researcher/attribute::name[@area="annotation"])
```

references the attribute *name* of all *researcher* elements that are children of the *person* element and their attribute *area* has the value *Annotation*.

The **XPointer xmlns() scheme** defines a syntax for XPointer parts, which allows a namespace to be defined for the following XPointer parts. To access the elements in the following document,

```
<organization xmlns="http://www.ontoweb.org/org">
    <name xmlns="http://www.ontoweb.org/person">Steffen Staab</name>
</organization>
```

the *xpointer()* function has to use the qualified element names. This is done with the help of *xmlns() scheme* as follows:

```
xmlns(c=http://www.ontoweb.org/org)
xmlns(p=http://www.ontoweb.org/person)
xpointer(/c:organization/p:name)
```

The XPointer framework offers the possibility for **custom XPointer schemes**.

```
...#mySchema(anyString)
```

However, one has also then to develop the XPointer processor for that custom scheme.

The custom schemes enable interested groups to develop their own scheme for specific applications. This exists for SVG (Scalable Vector Graphics)[14], for example:

```
MyDrawing.svg#svgView(viewBox(0,200,1000,1000))
```

The reader may note, that one could also envisage a custom scheme for Semantic Annotation.

2.3.3. Referencing to Text passages with XPointers

One requirement for Annotation is the referencing of single words or text parts in a document (e.g. a XHTML document). For this purpose one can use the extensions to XPath's basic node types, viz. *point* and in particular *range*.

A **point** represents a location in a document which it is possible to reference. A point can lay between two characters of a text node, and before and after any node of a document. A point has no dimension, i.e. it does not even contain a single character. A point is referenced by a container node, in which the point is located, and an index, that indicates the position of the point inside the node. The index is zero-based.

Figure 2.1 (from [DeRose *et al.*, 2002]) shows the points and nodes of the following document and the corresponding indexing.

```
<p>hello, <em>big </em>world</p>
```

With the XPointer node test *point()* one can directly reference points in a document. The expression

[14] http://www.w3.org/TR/SVG/linking.html#SVGFragmentIdentifiers

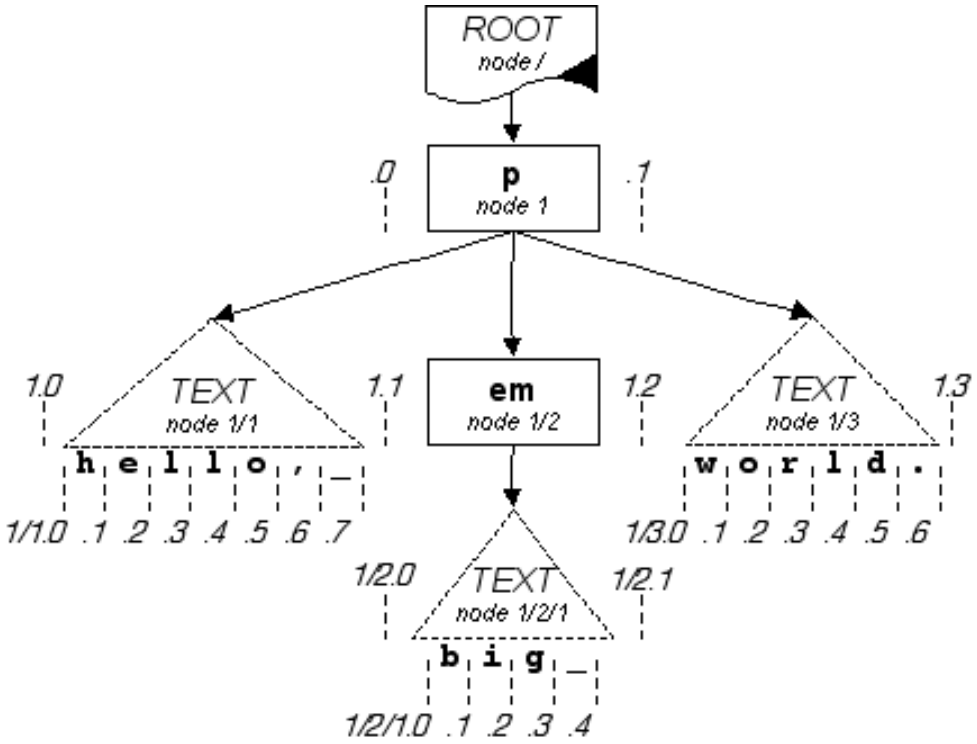

Figure 2.1.: Indexing of points and notes

```
xpointer(/p/em/point()[2])
```

references the point directly before the letter 'g', whereas the expression */p/em* defines the node, that includes the point, and *point()[2]* the point at the location with the index 2 inside the context node (c.f. Figure 2.1).

A **range** is defined by a start point and an end point. A range represents the XML structure and the content between the two given points.

The *string value* of a range is the concatenation of all characters which lay inside a text node and are located between the start and the end point. The XPointer recommendation *XPointer xpointer() scheme* offers several functions. The two most important functions for the purpose of annotation are the *range-to()* and the *string-range()* function.

The **range-to()** function returns to a given start and end point the corresponding range. This function is called:

```
context-location/range-to(end-location)
```

The start point of the resulting range is the start point of the context location; the end point of the range is the end point of the end location. For instance

```
<html>
   <body>
      <p>
      Today there will be a talk given by Tim <em>Berners-Lee</em> in
      room 244.
      </p>
   </body>
</html>
```

The expression

```
/html/body/p/range-to(/html/body/p/)
```

returns a range, with the start point immediately before the *p* element and the end point immediately after the *p* element. The expression

```
/html/body/p/point()[36]/range-to(/html/body/p/point()[41])
```

returns a range, with a start point immediately before the word *Tim* and the end point immediately after the *em* element. It represents therefore the document part:

```
Tim <em>Berners-Lee</em>
```

and its string value is *Tim Berners-Lee*.

The *range-to()* function is suitable to reference a range, when one knows the coordinates (container node and index) of the start and end points.

In its simplest form the **string-range()** function is called with a location-set and a search string.

```
string-range(locset, string)
```

As noted before, a location-set may consist of nodes (XPath nodes), points and ranges. For our purpose, only the cases are relevant, where locset is a node, to which an XPath expression refers. In the string-value of this node it is searched for according to the given string. The start and end point is the result of the function. For instance, the expression

```
string-range(/html/body,"Tim Berners-Lee")
```

results for the above document a range with a start point immediately before *Tim* (i.e. container node /html/body/p, Index:36) and the end point immediately after *Lee* (i.e. container node: /html/body/p/em, Index:11). The referenced document part of the range is:

```
Tim <em>Berners-Lee
```

The string value is *Tim Berners-Lee*.

It is possible that the search string occurs more than once in the node. Here, the function returns a range for every occurrence. Should the search string be an empty string, it will be found before the first character, between every character, and after the last character in the node.

Additional parameters are possible for the *string-range()*:

```
string-range(locset, string, number1, number2),
```

whereby

> *number1* - Start index of the range, relative to the index of the string found. The reader may note, that the string-range() functions use a 1 based indexing.
>
> *number2* - The number of characters in the range.

In the above example, the expression

```
string-range(/, "Tim Berners-Lee", 5, 11)
```

leads to a range with a start point immediately before *Berners*, and a end point immediately after *Lee*. The Reference document part and string value is each represented by *Berners-Lee*.

In conclusion, one can reference passages of text with ranges. The important XPointer functions for this purpose are the *range-to()* function and the *string-range()* function.

2.4. RDF and RDFS

The Resource Description Framework (RDF) [Manola and Miller, 2004] provides a means for adding metadata annotations to Web resources. RDF is a semantic data model and an attempt to address the aforementioned semantic limitations of XML. This data model consists of nodes connected by labeled arcs, where the nodes represent resources and the arcs represent properties of these resources. RDF Schema (RDFS) [Brickley and Guha, 2004] on the other hand, is used to define syntax and semantics of subsequent language layers (and even its own).

2.4.1. The RDF Data Model

A typical RDF statement consists of three elements synonymous with the subject, predicate, and object in a simple sentence. The basic element of RDF is the triple: a resource (the subject) is linked to another resource (the object) through an arc labeled with a third resource (the predicate). Therefore, a single triple is

a statement that a subject (e.g. a person, a car, a Web Site) stands in a specific relation (e..g. "is brother of"; "is driven by"; "is authored by") to an object (e.g. a person, Web Site). The object of a statement can be another resource, identified by a URI (Uniform Resource Identifier), a literal or a datatype value[15]. A URI[16] can be an HTML document identified by a Uniform Resource Locator (URL), a special form of a URI, such as http://www.w3.org/Overview.html, a specific HTML or XML element within a document [Kahan *et al.*, 2001, p.4], or an object that is not directly accessible via the Web such a printed book [Manola and Miller, 2004]. With URIs the markup content can be uniquely defined.

An example for a single triple is the following statement: "Siegfried cooperates with Steffen". The Subject and the Object of the statement need a URI to be identified, e.g. http://www.aifb.de/sha.html#sha to identify Siegfried and http://www.aifb.de/sst.html#sst to identify Steffen. Also the predicate needs an URI, e.g. http://annotation.org/iswc#cooperates for cooperates. The different parts of the statement correspond to the RDF triple schema as follows:

Subject (Resource)	http://www.aifb.de/sha.html#sha
Predicate (Property)	http://annotation.org/iswc#cooperates
Object (Resource)	http://www.aifb.de/sst.html#sst

The corresponding graphs looks like that shown in Figure 2.2.

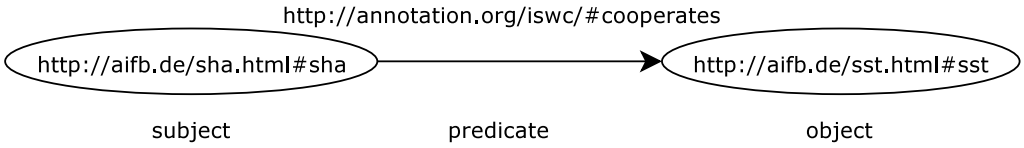

Figure 2.2.: RDF statement

In the following the URIs are abbreviated by using the XML namespace syntax. So instead of writing http://annotation.org/iswc#cooperates the namespace form iswc:cooperates is used with the assumption that the substitution of the namespace prefix iswc with http://annotation.org/iswc# is defined (e.g. see Listing 2.4).

A very useful feature for the annotation approach is the reification mechanism. RDF reification allows statements to be made about statements. This can be used to provide annotation meta-metadata[17], such as a creator, date of annotation, confidence factor. The RDF data model offers the predefined resource

[15] RDF has adopted the XML Schema model of datatypes

[16] For a clarification of the relationship between URIs, URLs, and URNs see: http://www.w3.org/TR/uri-clarification/

[17] Meta-metadata, because it is metadata of the statement and the statement is metadata of the annotated document.

rdf:statement and the properties *rdf:subject*, *rdf:predicate*, and *rdf:object* to reify a statement as a resource. Other properties, such as the date of annotation may then be attached to it. Figure 2.3 illustrates a reified statement.

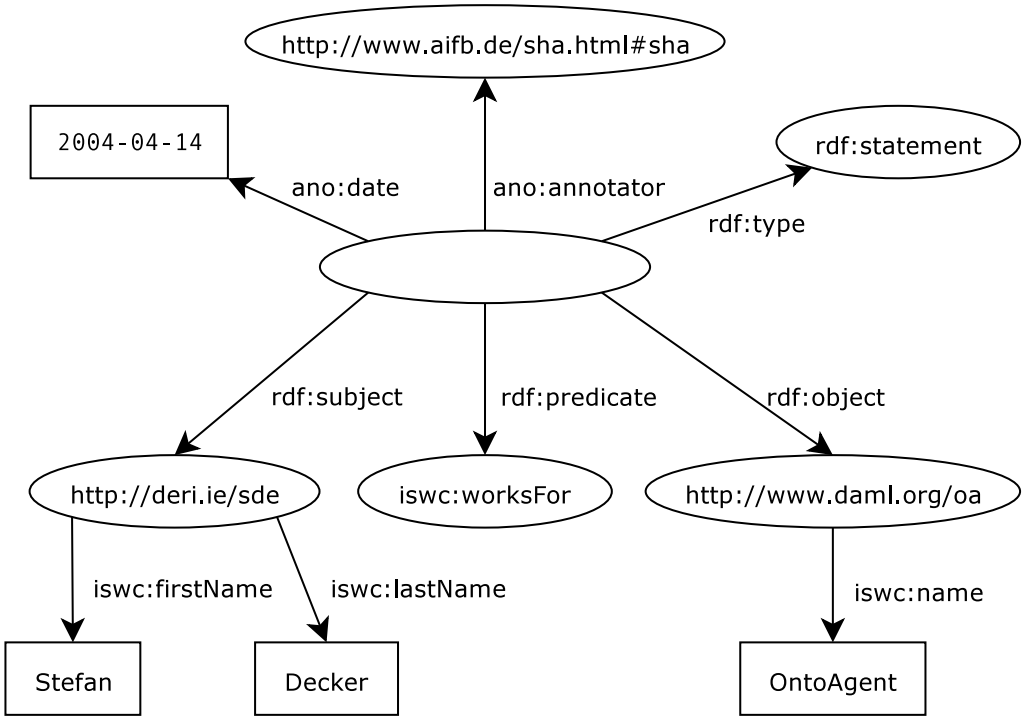

Figure 2.3.: RDF reification

In addition to these core primitives, RDF defines three types of containers to represent collections of resources or literals: one distinguishes (i) bags, that are unordered lists, (ii) sequences, that are ordered lists, and, (iii) alternatives, that are lists from which property can use only one value.

2.4.2. RDF Schema

As shown above, the RDF data model is a simple model for describing interrelationships among resources in terms of named properties and values [Brickley and Guha, 2004]. The RDF data model, however, provides no mechanism for declaring these properties, nor does it provide any mechanism for defining the relationships between these properties and other resources. That is the role of *RDF Schema* (RDFS) which describes how to use RDF to describe RDF vocabularies thereby giving specific meaning to the RDF statements. RDFS basically allows "classes, attributes (property types), value ranges and cardinality constraints for property types" to be defined [Decker *et al.*, 1999, p.9]. By sharing

RDF schemes, the re-usability of metadata definitions can be supported [Manola and Miller, 2004].

RDF(S) serves as a lightweight semantic layer. RDFS can also be used directly to describe a lightweight ontology [Fensel, 2001, p.60/61]. The result is only lightweight, because RDFS provides rather limited expressive power. This is due to the fact that RDFS lacks a standard for describing logical axioms to model definitions or complex relationships [Staab *et al.*, 2000b, p.5].

Illustrated here is the way in which an ontology can be modeled in RDF(S) by presenting a sample ontology (see Figure 2.4) in the abstract data model.

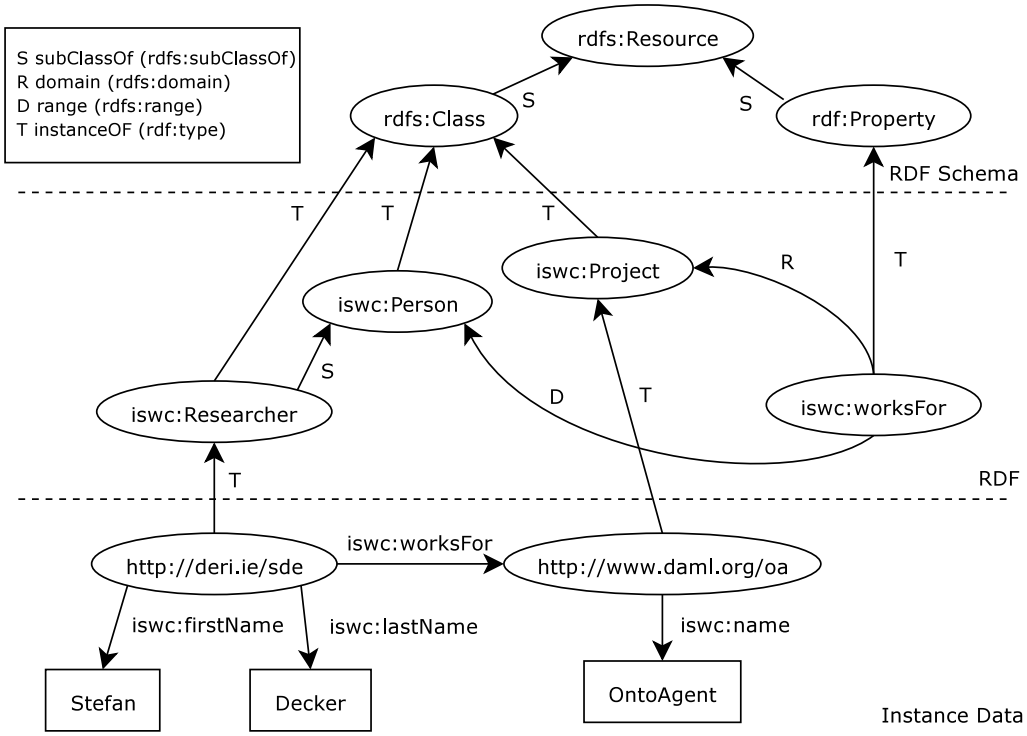

Figure 2.4.: RDF Layering

The most general class is *rdfs:Resource*. It has two subclasses, namely *rdfs:Class* and *rdf:Property*[18] (cf. Figure 2.4). When specifying a domain specific schema for RDF(S), the classes and properties defined in this schema will become instances of these two resources. The resource *rdfs:Class* denotes the set of all classes in an object-oriented sense. That means that classes like *iswc:Person* or *iswc:Project* are instances of the meta-class rdfs:Class. The same holds true for properties, i.e. each property defined in an application specific RDF-Schema is an instance of *rdf:Property*, e.g. *iswc:worksFor*. RDF-Schema defines the special

[18]Note that *Property* belongs to the *rdf* namespace and not to the *rdfs*namespace.

property *rdfs:subClassOf* that defines the subclass relationship between classes. Since *rdfs:subClassOf* is transitive, definitions are inherited by the more specific classes from the more general classes. Resources that are instances of a class are automatically instances of all superclasses of this class. Similar to *rdfs:subClassOf*, which defines a hierarchy of classes, another special type of relation *rdfs:subPropertyOf* defines a hierarchy of properties (e.g. one may express that *leaderOf* is an *rdfs:subPropertyOf* of *worksFor*). RDF-Schema allows the domain and range restrictions to be defined associated with properties. For instance, these restrictions allow the definition that "a person works for a project". As depicted in the middle layer of Figure 2.4 the domain specific classes *iswc:Person*, *iswc:Researcher*, and *iswc:Project* are defined as instances of *rdfs:Class*. In the same way domain specific properties are defined as instances of *rdf:Property*, i.e. *iswc:worksFor*, *iswc:firstName*, and *iswc:lastName*. Additionally, RDF-Schema defines more modeling primitives not shown in the Figure. The property *rdfs:label* allows a human readable form of a name to be defined and the property *rdfs:comment* enables comments.

2.4.3. RDF Syntax

RDF(S)[19] is an XML application, i.e., one can serialize RDF(S) in XML, and thus is a layer above XML in the Semantic Web architecture. Therefore, all metadata represented in RDF(S) can be also represented in XML. The RDF recommendation suggests two standard ways to serialize RDF data in XML: abbreviated syntax and standard syntax. Listing 2.4 shows in abbreviated syntax the XML serialization of an RDF data model and 2.5 shows the XML serialization of the corresponding RDFS vocabulary. Another format in which RDF(S) can be serialized is "Notation 3" (see section 2.4.4).

```
<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:iswc='http://annotation.org/iswc#'>
  <iswc:Person rdf:about="http://www.ontoweb.org/home/stefan">
    <iswc:firstname>Stefan</iswc:firstname>
    <iswc:lastname>Decker</iswc:lastname>
    <iswc:country>Ireland</iswc:country>
  </iswc:Person>
</rdf:RDF>
</xml>
```

Listing 2.4: XML serialization of RDF

[19] The term RDF(S) will be used instead of "both RDF and RDFS"

```
<?xml version='1.0'?>
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#>
<rdfs:Class rdf:ID="Person">
</rdfs:Class>
<rdf:Property ID="firstname">
  <rdfs:domain rdf:resource="#Person">
  <rdfs:range rdf:resource="&rdfs;Literal"/>
<rdf:Property>
<rdf:Property ID="lastname">
  <rdfs:domain rdf:resource="#Person">
  <rdfs:range rdf:resource="&rdfs;Literal"/>
<rdf:Property>
<rdf:Property ID="country">
  <rdfs:domain rdf:resource="#Person">
<rdfs:range rdf:resource="&rdfs;Literal"/>
<rdf:Property>
</rdf:RDF>
</xml>
```

Listing 2.5: XML serialization of RDFS

2.4.4. Notation 3

Notation3, or "N3" is a shorthand alternative serialization of RDF, designed with human-readability in mind, i.e. N3 is much more compact and readable than the XML serialization of RDF. The format is being developed by Tim Berners-Lee, with input from Dan Connolly and others.

According to the Notation 3 Specification[20], it was created as an experiment in optimizing the "expression of data and logic in the same language".

A subset of N3 is N-Triples[21]. It is an extremely constrained language which uses hardly any syntactic sugar at all: it is optimized for reading by scripts, and comparing using text tools. N-Triples contains on formulae, multiple objects, multiline literals, blank bNodes, or QNames. It breaks an RDF graph in separate triples, one on each line. N-Triples generated from the same RDF graph always come out the same, making it an effective way of validating the processing of an RDF document. N-Triples is easy to parse, and is an excellent lowest common denominator for serializations. Therefore it is used by the W3C in the RDF Test Cases document.

Listing 2.6 is the N3 equivalent of Listing 2.4 just as Listing 2.7 is the N3 equivalent of Listing 2.5.

[20] http://www.w3.org/DesignIssues/Notation3
[21] http://www.w3.org/2001/sw/RDFCore/ntriples/

```
:Stefan a :Person;  :firstname "Stefan"; :lastname "Decker";
:country "Ireland".
:Steffen a :Person;  :firstname "Steffen"; :lastname "Staab";
:country "Germany".
```

Listing 2.6: Example for data in N3

```
:Person a rdfs:Class.
:Professor a rdfs:Class; rdfs:subClassOf :Person.
:firstname rdfs:domain :Person; rdfs:range :Literal.
:lastname rdfs:domain :Person; rdfs:range :Literal.
:country rdfs:domain :Person; rdfs:range :Literal.
```

Listing 2.7: Example for schema in N3

The examples demonstrate that Notation 3 in comparison with an XML serialization allows more compact and human readable notation of RDF graphs. Hence, it will be used often in the later chapters of this thesis.

Excursus: N3 and Rules. Additionally to the RDF expressiveness we also have rules in N3. A simple rule might say something like (in some central heating vocabulary) "If the thermostat temperature is high, then the heating system power is zero", or

```
{:thermo :temp :high} => {:heating :power "0"}.
```

The curly brackets here enclose a set of statements, called formula. All formulae are enclosed by the curly brackets. Apart from the fact that the subject and object of the statement are formulae, the thermo example shown above is just a single statement.

The => used here is a special predicate, means implies. This is used to links formulae. It is actually the short hand of the URI log:implies, or: http://www.w3.org/2000/10/swap/log#implies. When two formulas linked with log:implies, it is a rule. Therefore all rules are just a different kind of statements.

Formulas take us out of the things we can represent using the current RDF/XML; these rules are not part of standard RDF syntax.

We could see formula as a collection of statements, for examples:

```
{:thermo :temp :high. :heating :power ''1''.} => {:heating :power "0"}.
```

is another instant of valid N3 rules. In fact, a formula could be more than just a collection of statements. We could have variables within a formula. Variables start with a ? . Examples of variables are:

```
{?x rdfs:subClass ?a, ?a rdfs:subclass b} => {?x rdfs:subClass ?b}
```

The rule says, if x is subclass of a, and if a is subclass of b then x is subclass of b.

2.5. Ontologies

2.5.1. Ontology Definition

There are different definitions in the literature of what an ontology should be, some of which are discussed in [Guarino97]. The most widely accepted definition originates with [Gruber94]: "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. 'Shared' reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group. The reference to 'a domain of interest' indicates that one is not concerned with modeling the whole world, but rather in modeling just the parts that are relevant to the task at hand.

Obviously, this is still far removed from a precise mathematical definition. One reason for this is that the definition should cover all different kinds of ontologies, and should not be related to a particular method of knowledge representation. However, as we want to stress structural aspects here, we present a definition of an ontological model that is used in this thesis and is valid as the smallest common denominator for current ontology languages such as OWL or RDF(S). The core "ingredients" of such an ontology are a set of concepts, a set of properties, and the relationships between the elements of these two sets. In detail, an ontology is defined as follows:

Definition 2.5.1
A core ontology *is a structure*

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

consisting of

- *three disjoint sets C, R and A whose elements are called* concepts , relations *and* attributes *respectively,*
- *a partial order $\leq_C$ on C, called* concept hierarchy *or* taxonomy,
- *a partial order $\leq_R$ on R, called* relation hierarchy.

We furthermore have two functions, domain*: $R \cup A \rightarrow C$ and* range*: $R \rightarrow C$.*

Relations are considered to be binary.

Definition 2.5.2
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.

If $c_1 <_C c_2$ and there is no $c_3 \in C$ with $c_1 <_C c_3 <_C c_2$, then c_1 is a direct subconcept of c_2, *and c_2 is a* direct superconcept of c_1. *We denote this by $c_1 \prec c_2$.* Direct superrelations *and* direct subrelations *are defined analogously.*

Definition 2.5.3
A knowledge base *is a structure*

$$KB := (C_{KB}, R_{KB}, A_{KB}, I, \iota_C, \iota_R, \iota_A)$$

consisting of

- *three sets C_{KB}, R_{KB} and A_{KB}, whose elements are called,* concepts, relations *and* attributes *respectively,*
- *a set I whose elements are called* instances *or* objects,
- *a function $\iota_C \colon C_{KB} \to \mathfrak{P}(I)$ called* concept instantiation,
- *a function $\iota_R \colon R_{KB} \to \mathfrak{P}(I \times I)$, called* relation instantiation.
- *a function $\iota_A \colon A_{KB} \to \mathfrak{P}(I \times STRING)$, called* attribute instantiation. *STRING denotes a the set of all strings.*

We assume that each instance i is uniquely identified by some string $id(i)$, called the identifier *of the instance i. The set of all identifiers is $id(I)$.*

Given a knowledge base as above, we define the class assignments $AC = \{(i, c) | i \in \iota_C(c)\} \subseteq I \times C_{KB}$, *the* relation assignments $AR = \{(i, r, j) | (i, j) \in \iota_R(r)\} \subseteq I \times R_{KB} \times I$ *and the* attribute assignments $AA = \{(i, a, s) | (i, s) \in \iota_A(a)\} \subseteq I \times A_{KB} \times STRING$.

Conversely, given AC, AR and AA one can obtain ι_C, ι_R and ι_A.

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 2.5.4
Let $KB := (C_{KB}, R_{KB}, A_{KB}, I, \iota_C, \iota_R, \iota_A)$ be a knowledge base. The extension $[\![c]\!]_{KB} \subseteq I$ of a concept $c \in C_{KB}$ *is recursively defined by the following rules:*

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

The extension $[\![r]\!]_{KB} \subseteq I \times I$ of a relation $r \in R_{KB}$ *is recursively defined by the following rules:*

- $[\![r]\!]_{KB} \leftarrow \iota_R(r)$
- $[\![r]\!]_{KB} \leftarrow [\![r]\!]_{KB} \cup [\![r']\!]_{KB}$, *for* $r' <_R r$.

If the reference to the knowledge base is clear from the context, we also write $[\![c]\!]$ and $[\![r]\!]$ instead of $[\![c]\!]_{KB}$ and $[\![r]\!]_{KB}$.

The following definition tells us if a knowledge base is consistent with an ontology.

Definition 2.5.5
A knowledge base $KB = (C_{KB}, R_{KB}, A_{KB}, I, \iota_C, \iota_R, \iota_A)$ *is* consistent with an ontology $\mathcal{O}$, *if all of the following hold:*

- $C_{KB} \subseteq C$,
- $R_{KB} \subseteq R$,
- $A_{KB} \subseteq A$,
- $[\![r]\!] \subseteq [\![dom(r)]\!] \times [\![range(r)]\!]$ *for all* $r \in R$.
- $\iota_A(a) \subseteq [\![dom(a)]\!] \times STRING$ *for all* $a \in A$.

Ontologies formalize the intentional aspects. The extensional part is provided by a knowledge base, which contains assertions about instances of the concepts and relations. To define and instantiate a knowledge base is called ontology population. The reader may note, that Semantic Annotation does not just enrich a document with semantic metadata, it also instantiates a knowledge base, viz. it populates an ontology.

2.5.2. Classification

The ontology and knowledge base definitions introduced above provides structures for instantiating ontologies and knowledge bases. These structures enables the development of different types of ontologies. Different classification systems for ontologies have been developed [van Heijst, 1995; Guarino, 1998; Jasper and Uschold, 1999]. A classification system that uses the subject of conceptualization as a main criterion has been introduced by Guarino (*cf.* [Guarino, 1998]). He suggests to develop different kinds of ontologies according to their level of generality as shown in Figure 2.5.

Thus, different kinds of ontologies may be distinguished as follows:

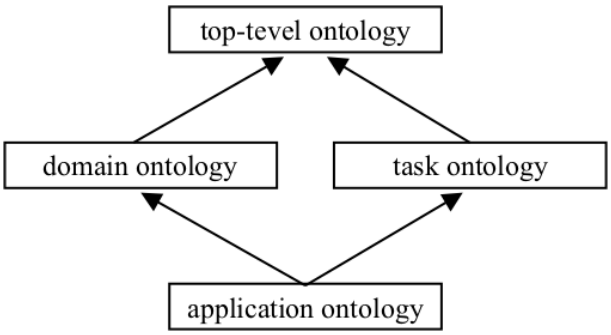

Figure 2.5.: Different kinds of ontologies and their relationships

- **Top-level ontologies** describe very general concepts like space, time, event, which are independent of a particular problem or domain. Such unified top-level ontologies aim at serving large communities of users and applications. Recently, these kinds of ontologies have been also introduced under the name **foundational ontologies**.
- **Domain ontologies** describe the vocabulary related to a specific domain (such as researcher community or tourism), *e.g.* by specializing concepts introduced in a top-level ontology.
- **Task ontologies** describe the vocabulary related to a generic task or activity (e.g. bargain or selling), *e.g.* by specializing concepts introduced in a top-level ontology.
- **Application ontologies** are the most specific ontologies. Concepts in application ontologies often correspond to roles played by domain entities while performing a certain activity, *i.e.* application ontologies are a specialization of domain and task ontologies. They form a base for implementing applications with a concrete domain and scope.

In this work we refer by ontology to domain and in particular to application ontologies.

2.6. Ontology Languages

In the previous section, the general appearance of an ontology, the conceptualization, has been illustrated. To share this conceptualization, a formal knowledge

representation of the ontology and further of the knowledge base, viz. the metadata is needed which is expressed by an ontology (modeling) language.

A couple of representation mechanisms have been developed that allow for the formal representation of ontologies. The most common ontology languages for knowledge representation, also called representation languages, can be differentiated in the following three categories of logics [Fensel, 2001, p.62]:

- Enriched first-order predicate logic languages, e.g. CycL [Lenat and Guha, 1990], Knowledge Interchange Format (KIF) [Genesereth and Fikes, 1992] and Conceptual Graphs (CGs) [Sowa, 1984]
- Frame-based languages, e.g. Ontolingua [Farquhar *et al.*, 1996], Frame Logic [Kifer *et al.*, 1995] , and Simple HTML Ontology Extensions (SHOE) [Heflin and Hendler, 2000a]
- Description Logics, e.g.LOOM [MacGregor, 1991].

The above mentioned ontology languages in their current status are not suitable to perform Web-based tasks, nor to combine the advantages of the different logics categories, nor to perform inferencing services well, and limit the volume of the knowledge base. According to Tim Berners-Lee, "the challenge of the Semantic Web [...] is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported onto the Web"[Berners-Lee *et al.*, 2001]. Therefore, extensive research is being conducted to develop such languages.

DAML+OIL. A number of research groups in the USA and Europe had already identified the need for a more powerful ontology modeling language. This led to a joint initiative to define a richer language, called DAML+OIL. DAML+OIL combines the DARPA[22] Agent Markup Language (DAML) with the Ontology Inference Layer (OIL), the European language, and thus offers "expressive constructs aimed at facilitating agent interaction on the Web"[Noy and McGuinness, 2001]. DAML+OIL in turn was taken as the starting point for the W3C Web Ontology Working Group in defining OWL, the language that is the recommended ontology language of the Semantic Web an to be broadly accepted.

OWL. OWL is syntactically layered on top of RDF. Therefore, the official syntax of OWL is the syntax of RDF. However, OWL extends RDF with additional vocabulary that can be interpreted as OWL ontologies when used to form particular RDF graphs. Consequently, OWL ontologies can be encoded in normal

[22]Defense Advanced Research Project Agency

```
<rdf:RDF>
  <o:Person rdf:about="http://www.ontoweb.org/home/stefan">
    <o:firstname>Stefan</o:firstname>
    <o:lastname>Decker</o:lastname>
    <o:country>Ireland</o:country>
  </o:person>
</rdf:RDF>
```

Listing 2.8: Example for OWL Instances

RDF/XML documents parsed into ordinary RDF graphs. The additional vocabulary defined in OWL renders OWL into a Description Logic language. Several subsets of OWL (c.f. [Antoniou and van Harmelen, 2004, p.70]) were defined in the standard to accommodate various interest groups and allow to safely ignore language features that are not needed for certain applications.

- *OWL Full*: OWL Full contains all constructors of the OWL language and allows the arbitrary combination of those constructors. It also allows those constructors to be combined with RDF and RDF Schema. OWL Full is fully upward compatible (syntactically and semantically) with RDF. Hence, any legal RDF document is also a legal OWL Full document. However, this compatibility leads to "uncontrollable computational properties"[Antoniou and van Harmelen, 2004, p.70] and brings with it the disadvantage that OWL Full is "undecidable, dashing any hope of complete (let alone efficient) reasoning support"[Antoniou and van Harmelen, 2004, p.70].
- *OWL DL*: OWL DL is a subset of OWL. In order to regain computational efficiency it restricts the way in which the constructors of OWL and RDF can be used. As a drawback one loses full RDF compatibility. A legal OWL DL document will still be a legal RDF document but not necessarily vice versa. "An RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document"[Antoniou and van Harmelen, 2004, p.71].
- *OWL Lite* OWL Lite is a subset of OWL DL and thus has a restricted expressivity. Among other things, it excludes disjointness, enumerated classes and a cardinality greater than one. "The advantage of this is a language that is both easier to grasp (for the user) and easier to implement (for tool builders)"[Antoniou and van Harmelen, 2004, p.71].

The Example 2.8 shows, that instances of classes – in OWL called individuals – are declared as in RDF (cf., Listing 2.4). Note that OWL, as RDF(S), can be

```
<?xml version='1.0'?>
<!DOCTYPE uridef[
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema">
]>
<rdf:RDF
  xmlns:owl  ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs ="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd  ="http://www.w3.org/2001/XMLSchema#"
 >
<owl:Class rdf:ID="Person">
</owl:Class>
<owl:DatatypeProperty ID="firstname">
  <rdfs:domain rdf:resource="#Person">
  <rdfs:range rdf:resource="&xsd;string"/>
<owl:DatatypeProperty>
<owl:Property ID="lastname">
  <rdfs:domain rdf:resource="#Person">
  <rdfs:range rdf:resource="&xsd;string"/>
<owl:DatatypeProperty>
<owl:DatatypeProperty ID="country">
  <rdfs:domain rdf:resource="#Person">
  <rdfs:range rdf:resource="&xsd;string"/>
<owl:DatatypeProperty>
```

Listing 2.9: Example for OWL Ontology

serialized in XML and N3. The ontology corresponding to the instance data can be found in Listing 2.9.

3. Semantic Annotation for the Web

This Chapter is about the vision of the Semantic Web (Section 3.1). It shows the Semantic Web Layer Cake based on the metadata languages introduced before. It demonstrates how to apply these technologies in order to provide Semantic Annotation. Therefore, it presents the necessary infrastructure in (Section 3.2) and the underlying processes in (Section 3.3). Subsequently, Section 3.4 defines i) the terminology (Section 3.4.1), ii) the semantic (Section 3.4.2) and iii) the content (Section 3.4.3) of the markup that we aim to create for the Semantic Web.

3.1. The Semantic Web

The Semantic Web aims at machine-processable information. The step from the current Web to the Semantic Web is the step from the manual processing of information to the automatic processing of information. This step is comparable to the step from the manual processing of goods to the machine processing of goods at the beginning of the industrial revolution. Hence, the Semantic Web can be seen as the dawn of the informational revolution.

The Semantic Web enables automated intelligent services such as information brokers, search agents, information filters etc. The Semantic Web, consisting of machine processable information, will be enabled by further levels of interoperability.

Technology and standards need to be defined not only for the syntactic representation of documents (like HTML), but also for the their semantic content. Semantic interoperability is facilitated by recent W3C standardization efforts, notably XML/XML Schema (cf., Section 2.2), RDF/RDF Schema (cf., Section 2.4) and OWL (cf., Section 2.6). The technology stack envisioned by the W3C is depicted in Figure 3.1.

As is apparent, XML as well as an XML Schema are the second layer above URIs and Unicode. The third layer is RDF and RDFS. The next layer is the ontology language. On top of the ontology language, there is a need for a language to express logic, so that information can be inferred and better put into relation. Once there is logic, it makes sense to use it to prove things. The proof layer enables everyone to write logic statements, and an agent can follow these Semantic

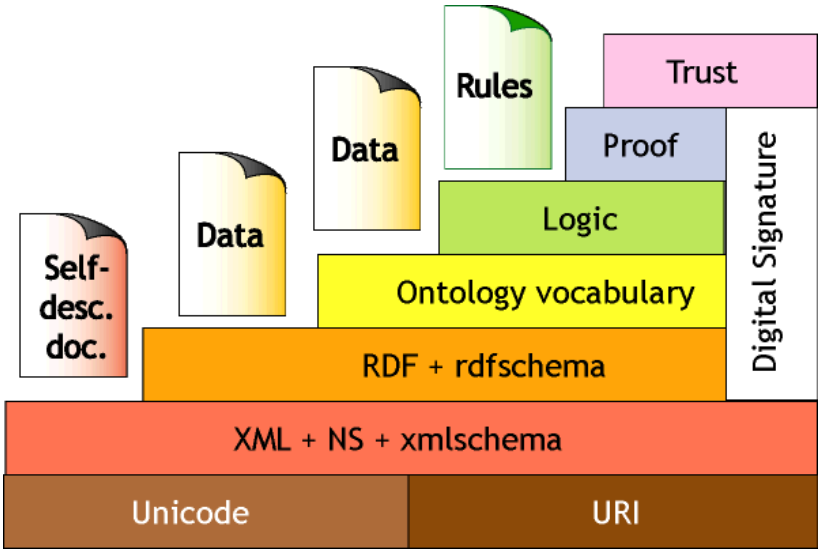

Figure 3.1.: Semantic Web Layer Cake

"links" to construct proofs, so that validity of a statement, especially an inferred statement, can be checked. The proof layer combined with digital signatures will lead to trust. Consequently, ontology and ontology based metadata are the basic ingredients for the Semantic Web layer cake. An important question is therefore how to create and use ontology and ontology based metadata.

3.2. Infrastructure for the Semantic Web – The Information Foodchain

Stefan Decker defines in his thesis [Decker, 2002] with the *information foodchain for the Semantic Web* an infrastructure for the Semantic Web and presents applications for ontology engineering and metadata creation among others.

The food chain (Figure 3.2) starts with the construction of an ontology - preferably using a support tool, an ontology construction tool[1]. The ontology is the foundation for a set of data items. The next part of the information food chain is a tool to support the task of structuring the HTML pages. A Web page annotation tool (cf., Chapter 4) provides the means for browsing an ontology and for selecting appropriate terms of the ontology and mapping them to sections of

[1]For example, the Ontology Engineering environment, OntoEdit (cf. [Sure *et al.*, 2002b])

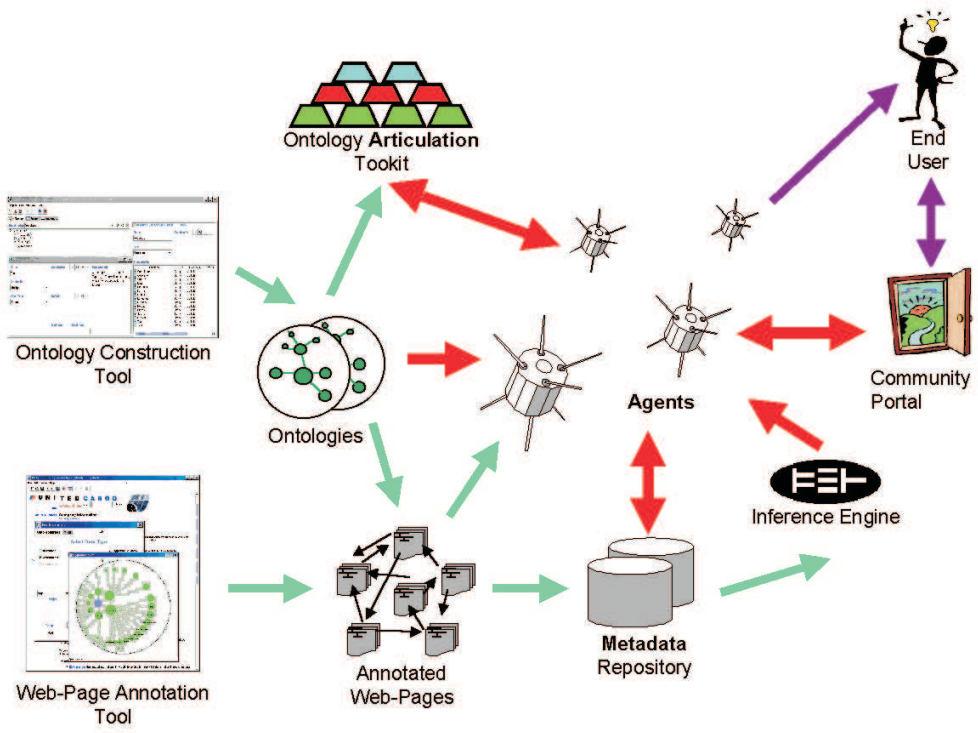

Figure 3.2.: Information Foodchain for the Semantic Web

a Web page. The Web page annotation process creates a set of annotated Web pages, which are available to an automated agent to achieve his task. Of course, the annotation process itself has a human component: although the effort to generate the annotation of a Web page is of an order lower in magnitude than the creation of the Web page itself, there has to be some incentive to expend the extra effort. The incentive for the creation of the annotation (which is metadata for the Web page) is visible on the Web for a community Web portal, which presents a community of interest distributed on the Web to the outside world in a concise manner. The data collected from the annotated Web pages simplifies to a significant extent the task of maintaining a community Web portal because changes are incorporated automatically, without any manual work. An automated agent itself needs several sub-components: an important task of the agent is the integration of data from several distributed information sources. Because of the need to describe the relationships between the data in a declarative way (otherwise the agent has to be programmed for every new task), an agent needs

an inference engine for the evaluation of rules and queries. The inference engine is coupled with a metadata repository - the memory of an agent as to where retrieved information is cached. Furthermore, if an automated agent browses the Web it will usually encounter data formulated in unknown ontologies. Therefore, it needs the facility to relate unknown ontologies to ontologies with which it is already familiar. This facility is an Ontology Articulation Toolkit for information mediation.

3.3. Processes for the Semantic Web – Knowledge Process and Knowledge Meta Process

While the *information foodchain* presents necessary tools and infrastructure this Section presents the underlying processes of ontology and metadata creation.

Two central processes are distinguished in the thesis of York Sure [Sure, 2003] based on the duality of ontology and metadata: the development of an ontology, named knowledge meta process, and then the subsequent creation of a knowledge base, named knowledge process (see Figure 3.3).

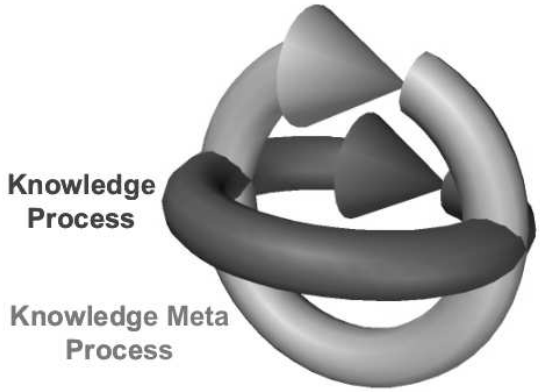

Figure 3.3.: Two orthogonal processes with feedback loops

The *knowledge meta process* comprises all aspects that are necessary for the creation of an ontology as well as its extension and adaption. The *knowledge process* describes in particular the steps for the creation and processing of ontology based metadata.

3.3.1. Knowledge Meta Process

The knowledge meta process is devoted to the modeling of ontologies. The process can be regarded as a form of reverse-engineering, because the structure underlying the resources needs to be derived from the Web resources with the help of domain experts. But this derivation is not exact: the result, the ontology, and also the process, the steps to an ontology, are variable.

Considering the result, since each ontology-designer has a certain application in mind for which he designs his ontology and has another understanding of the considered domain, a multitude of ontologies can be created for one domain. The dependence on an application, however, should be not so strong that it limits the re-usability of an ontology [Noy and McGuinness, 2001, p.4]. With regard to the process "no single correct ontology-design methodology" exists [Noy and McGuinness, 2001, p.4] which describes the steps for designing an ontology. Proposals for methodologies, however, have been developed by various researchers such as the Buchanan-Methodology, the Uschold-Methodology [Uschold1996], and Methontology by López et al. [Lopez1999]. The methodologies have in common approximately the following four steps: specification, conceptualization/refinement, implementation, and evaluation. More steps which can be added are a feasibility study (the four steps mentioned above) and a maintenance phase (following the four steps mentioned above) [Staab *et al.*, 2001c, p.2]. The feasibility study should support the decision if the creation of an ontology is useful in a certain domain of knowledge. The maintenance phase is important so that ontologies keep track of the change in the real world and are evolving with time.

3.3.2. Knowledge Process

In the *knowledge process* (Figure 3.4), which is orthogonal to the knowledge meta process, metadata is created to describe the relevant information of some resources with the use of ontology. The resulting metadata consists of instances of a specific concept connected with properties (attributes and relations) and axioms. The output of the metadata creation process is the knowledge base, viz. a collection of this metadata in a specified formal representation language. In the event that Web resources are described, the knowledge base or a part of it can be embedded in existing Web page description to provide semantic)information for intelligent agents in the WWW. The knowledge base can be extended by inferring "new" facts on the basis of defined axioms. This creation process is the topic of this thesis.

The reader may note that there exist no methodology for the design of metadata as there is for the ontology-design.

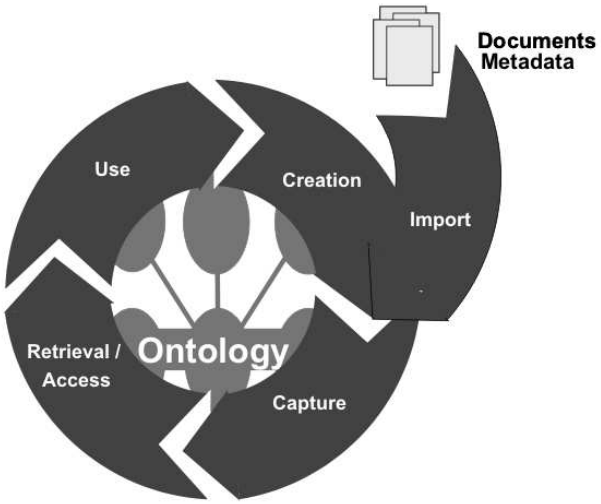

Figure 3.4.: The knowledge process

3.4. Semantic Annotation

In the following we define the terminology, the semantic and the content of metadata creation as envisioned in this thesis.

3.4.1. Terminology

The terminology used in this thesis has been elaborated because many of the terms that are used with regard to metadata creation tools carry several, ambiguous connotations that imply conceptually important decisions.

- **Ontology**: An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest (cf., Section 2.5). In our case, an ontology is defined in RDF(S) or OWL. Hence, an ontology is constituted by statements expressing definitions of OWL classes – RDF(S) resources, respectively – and properties ([OWL Reference, 2004], [Brickley and Guha, 2004]).
- **Annotations**: An annotation in our context is a set of instantiations attached to an HTML document. We distinguish *(i)* instantiations of OWL classes, *(ii)* instantiated properties from one class instance to a datatype instance — henceforth called attribute instance (of the class instance), and

(iii) instantiated properties from one class instance to another class instance — henceforth called relationship instance.

Class instances have unique URIs, e.g. like '`http://www.aifb.uni-karlsruhe.de/WBS/sst/#Steffen`'. They frequently come with attribute instances, such as a human-readable label like 'Steffen'.

- **Metadata**: Metadata are data about data. In our context the annotations are metadata about the HTML documents.
- **Relational Metadata**: We use the term relational metadata to denote the annotations that contain relationship instances.

  Often, the term "annotation" is used to mean something like "private or shared note", "comment" or "Dublin Core metadata". This alternative meaning of annotation may be emulated in our approach by modeling these notes with attribute instances. For instance, a comment note "I like this paper" would be related to the URL of the paper via an attribute instance 'hasComment'.

  In contrast, relational metadata also contain statements like 'Siegfried cooperates with Steffen', *i.e.* relational metadata contain relationships between class instances rather than only textual notes.

Figure 3.5 illustrates our use of the terms "ontology", "annotation" and "relational metadata". It depicts some part of the SWRC[2] (Semantic Web Research Community) ontology. Furthermore it shows two homepages, viz. pages about Siegfried and Steffen (`http://www.aifb.uni-karlsruhe.de/WBS/sha` and `http://www.aifb.uni-karlsruhe.de/WBS/sst`, respectively) with annotations given in an XML serialization of RDF facts. For the two persons there are instances denoted by corresponding URIs (`http://www.aifb.uni-karlsruhe.de/WBS/sst/#Steffen` and `http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried`). The swrc:name of `http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried` is "Siegfried Handschuh". In addition, there is a relationship instance between the two persons, *viz.* they cooperate. This cooperation information 'spans' the two pages.

3.4.2. The Semantics of Semantic Annotation

There are two players in the annotation game, the annotation *provider* and the annotation *consumer*. The key is, as Bechhofer et. al. points out [Bechhofer and Goble, 2001], that consumer and provider *share underlying assumptions* about the annotation. Part of this assumption is the common ontology, but part of it is how the terms of the ontology are to be used.

[2] http://ontobroker.semanticweb.org/ontos/swrc.html

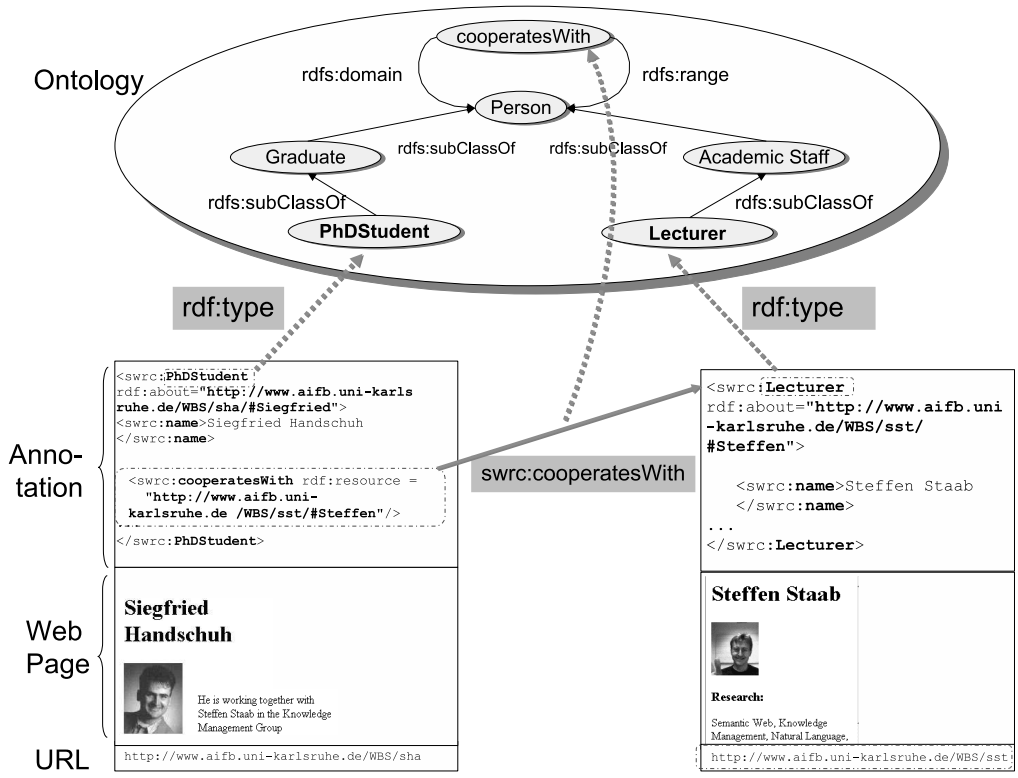

Figure 3.5.: Annotation example.

In the literature about Semantic Annotation it is evident that different assumptions exist about the nature and the scope of Semantic Annotation.

Bechhofer et. al. [Bechhofer and Goble, 2001] differentiate between:

- **Decoration**: Is the idea of creating a kind of user comment about Web pages. This is view that Annotea has adopted for their annotation (cf. [Kahan *et al.*, 2001]).
- **Linking**: To annotate a document with further links. The provision of dynamic linking as annotation is used by the COHSE project (cf. Section 3.1.1 in [Bechhofer and Goble, 2001]). Annotation within COHSE can be seen as a mechanism that allows the user to specify possible link anchors within a document, with the anchor being associated with a conceptual description.

- **Instance Identification**: We are making an assertion that there is some resource in the Web, such that it is an instance of an concept, and the identifier of the instance, viz. the URI identifies the resource. This means the instance about the annotation is being made is clearly accessed by the given URI.
- **Instance Reference**: We are making an assertion that there is some individual in the world, such that it is an instance of the concept, and the identifier of the instance, viz. the URI identifies not the individual itself, but the a reference in a document to the real world individual. This is the semantic of annotation that is used by our annotation framework.
- **Aboutness**: A user expresses that a particular resource (i.e. a Web page) is about a certain concept, but not an instance of an concept.
- **Pertinence**: A user expresses that a particular resource (i.e. a Web page) give further useful information about a concept,

Our viewpoint of annotation is based on *instance reference* (cf., Section 3.4.1), but we can emulate most of the other annotation types with our framework.

3.4.3. Layering of Annotation

As shown in the before, there exists different notions of the semantic of Semantic Annotation. Additionally, [Rinaldi *et al.*, 2003] presents a layering of annotation which reflects different aspects of content to be represented:

- **Structural Annotation**: Used to define the physical structure of the document, it s organization into head and body, into sections, paragraphs and sentences.
- **Linguistic Annotation**: Associated to a short span of text (smaller than a sentence), and identify lexical units. They could be referred to also as *Textual Annotations* or *Lexical Annotation*. This corresponds to the *grammatical structure* in ([Euzenat, 2002]).
- **Semantic Annotation**: Corresponds to our view of Semantic Annotation. Similar to the representation of the *logical structure* ([Euzenat, 2002]) of the document.

As mentioned before our focal point lays on the *Semantic Annotation*. However, we aim to present a generic annotation model that is able to deal with most of the Semantics of Semantic Annotation, as well as with the aspects or layers of Annotation (cf., Section 7.1).

3.5. Summary

The Semantic Web aims at machine-processable information. Fundamental to the vision of the Semantic Web is the use of a formal markup language for annotation of Web resources, based on technologies such as facilitated by recent W3C standardization efforts. One possible infrastructure to realize a Semantic Web is given in the *information foodchain*. This foodchain envisions an Ontology-Construction-Tool for the creation of Ontology and a Web-Page-Annotation-Tool for formal markup. From a process point of view, the *knowledge meta process* comprises all aspects that are necessary for the creation of an ontology and the *knowledge process* describes the metadata creation and processing. Hence, the conception and implementation of an annotation framework that feeds the *information foodchain* and thereby supports the *knowledge process* is the topic of this thesis. Therefore, we have defined the goal of the annotation process, which is the creation of *relational metadata* using the semantic of an *instance reference* and representing the logical structure of a Web page.

Part II.

Metadata for the Semantic Web

"The Web is about links;
the Semantic Web is about the relationships implicit in those links."
— Dan Brickley

4. Annotation and Authoring Framework

This Chapter presents an annotation and authoring framework: CREAM, that allows for the creation of semantic metadata about static and dynamic Web pages, i.e. for Semantic Annotation of the Shallow and the Deep Web. CREAM supports the manual and the semi-automatic annotation of static Web pages, the authoring of new Web pages with the simultaneous creation of metadata, and the deep annotation of Web pages defined dynamically by database queries. In the following we first describe the case studies from which we took a major part of our experiences for guiding the development of CREAM (Section 4.2). Then, we describe some of the requirements in detail that were derived from the case studies (Section 4.3). In Section 4.4 we derive the design of CREAM from the requirements previously elaborated. In Section 4.5, we specify how the meta ontology may modularize the ontology description from the way the ontology is used in CREAM. In Section 4.6, we explain the major modes of interaction with OntoMat, our implementation of CREAM.

References: This chapter is mainly based on [Handschuh *et al.*, 2001], [Handschuh and Staab, 2002] and [Handschuh and Staab, 2003].

4.1. Introduction

The Semantic Web builds on metadata describing the contents of Web pages. In particular, the Semantic Web requires *relational metadata*, *i.e.* metadata that describe how resource descriptions instantiate class definitions and how they are semantically interlinked by properties. We have carried out several case studies that build on this idea of the Semantic Web in order to provide intelligent applications to make knowledge about researchers, about companies and markets, and about research papers accessible by semantic means (cf. Section 4.2). An important cornerstone of the case studies — and many other scenarios in the Semantic Web — is a method and a mechanism that let the user easily and comprehensively contribute relational metadata on which the Semantic Web can be build. For this purpose, we have developed our Semantic Annotation framework, CREAM — CREAting Metadata for the Semantic Web — that is presented here. CREAM is designed to enable the easy and comfortable creation of semantic metadata.

CREAM allows for the *a posteriori* annotation of existing resources. *A posteriori* creation of metadata involves the consideration of an item (e.g. a document) by an agent (possibly, but not necessarily, a human agent) and its description with metadata. *(i)*, the process may include the identification of elements already existing within the item. For instance, one may consider an HTML page and identify its author by a footer appearing *within* the page. *(ii)*, one may classify the item to belong to a category like Business though neither the term itself nor a synonym or hyponym appears in the item.

A posteriori creation of metadata comes with a major drawback. In order to provide metadata about the contents of a Web page, the author must *first* create the content and *second* annotate the content in the additional, a-posteriori, annotation step. To avoid the overhead work, we propose that, *(iii)*, an author has the possibility to easily combine authoring of a Web page *and* the creation of relational metadata describing its content.

Scrutinizing the differences between the *modes of interaction (i)*, *(ii)*, and *(iii)*, we found that the major problems one must deal with for an annotation framework are identical. In fact, we found it preferable to hide the border between annotation (i.e. *(i)* and *(ii)*) and authoring (i.e. *(iii)*) as far as possible within CREAM. Some questions, however, were triggered by the need to modularize the ontology that defines the structure of the relational metadata from its use for annotation or authoring. For this purpose, we here introduce a **meta ontology** that describes how the annotation and authoring modes of OntoMat interfere with classes and properties of the ontology proper. There we modularize the ontology parts needed in the metadata creation process from the ones relevant for the targeted content description.

For the CREAM framework we have also provided a reference implementation, viz. OntoMat, which is freely available for download.[1]

4.2. Case Studies for CREAM

There are many scenarios in which Semantic Annotation may prove beneficial. Below, we describe three case studies that we have performed and that have guided our development of CREAM. The case studies have in common that they require the generation of metadata given a HTML document from which a human could identify relevant metadata entities.

During the studies, we repeatedly found several principal problems. Some of the problems were mostly syntactic, viz. people would easily make *syntactic errors*, *e.g.* closing XML parentheses incorrectly or not at all. In the further course, however, we found that people violated *semantic constraints*, too, *e.g.* they would

[1] http://annotation.semanticweb.org

give a string instead of an object identifier. Also, human annotators would violate *pragmatic constraints*, e.g. have wrong assumptions about the existence of object descriptions in the Semantic Web and, thus, make errors when referring to people.

4.2.1. KA2 Initiative and KA2 Portal

The origin of our work facing the challenge of creating relational metadata dates back to the start of the seminal KA2 initiative [Benjamins *et al.*, 1999], *i.e.* the initiative for providing semantic markup on HTML pages for the knowledge acquisition community and its presentation in a Web portal [Staab *et al.*, 2000a]. The KA2 portal[2] provides a view onto knowledge of the knowledge acquisition community. Besides of semantic retrieval as provided by the original KA2 initiative, it allows comprehensive means for navigating and querying the knowledge base and also includes guidelines for building such a knowledge portal. The potential users provide knowledge, e.g. by annotating their web pages in a decentralized manner. The knowledge is collected at the portal by crawling and presented in a variety of ways.

Metadata for KA2 was initially contributed by manually editing HTML pages providing metadata with an ASCII editor. The syntactic errors that arose from the difficulty of this manual task soon became obvious. However, we still had the wrong intuition that a simple tool that would take care of syntax issues would resolve the problem. Thus, OntoPad was developed in order to facilitate editing [Fensel *et al.*, 1999; Decker, 2002]. However, it was soon found that the semantic and pragmatic concerns are as important as the syntactic issue and could hardly be dealt with by such simple tools that were available then.

4.2.2. TIME2Research Portal

In a second case study, we have used Semantic Annotation in order to provide knowledge about the TIME (telecommunication, IT, multimedia, e-business) markets in the TIME2Research portal [Staab and Maedche, 2001]. The principal idea of the TIME2Research portal is that business analyst review news tickers, business plans and business reports. A considerable part of their work requires the comparison and aggregation of similar or related data, which may be done by semantic queries like "Which companies provide B2B solutions?", when the knowledge is semantically available. Therefore, we have created a knowledge portal for business analysts that let the analysts provide Semantic Annotation for incoming HTML, Word and Excel documents. Then, they were able to browse through the metadata and the documents and they could perform semantic searches looking for individual as well as semantically aggregated information provided by annotation.

[2] http://ka2portal.aifb.uni-karlsruhe.de

For the case study we provided OntoAnnotate[3]. OntoAnnotate gave the human annotator an easy-to-use interface. He was allowed to highlight some piece of HTML text, Word text or an Excel cell and then drag'n'drop it onto a corresponding — possibly instantiated — concept, attribute or relation from the ontology. By the control through ontology guidance and browsing of existing facts most of the semantic and pragmatic problems were dealt with. A substantial part of this paper is about this part of OntoAnnotate that is also realized within CREAM.

In the TIME2Research case study we found that people would need the possibility to easily create documents and their metadata *in one step*, rather than produce a report and create the Semantic Annotation *a posteriori*. The reason is that in a commercial setting people are even more resistant to overhead work than in a closed-community, academic initiative like KA2.

4.2.3. Authors' Annotations of Paper Abstracts at ISWC

The most recent case study involved the mark-up of papers for the International Semantic Web conferences — ISWC-2002[4], ISWC-2003[5], and ISWC-2004[6].

"Eating their own dog food", authors of accepted paper were required to annotate title-pages and abstracts of their papers in order to contribute to the actual creation of the Semantic Web and to produce experience in actually providing Semantic Annotation. The authors could, but were not forced to, use the OntoMat annotation tool and a specifically created ISWC ontology.

The results from the ISWC abstracts annotation study complemented some of the problematic experiences we gained from KA2. Without a tool, even highly trained authors of highly valued papers produced mark-up that obviously was not XML (not to mention RDF) as not all parentheses matched correctly. Others who did not use a tool reused existing RDF code — copying the syntactic errors that the original code contained.

Given the code produced by OntoMat, some third parties actually took advantage of the metadata to build applications on top. M. Frank *et al.* produced a spreadsheet-like summary of some of the core data exploiting their WebScripter approach [Frank *et al.*, 2002b]. C. Fillies and F. Weichhardt used their visualization methods in order to create a graphical depiction of the underlying metadata [Fillies and Weichhardt, 2002].[7]

[3] OntoAnnotate is now a commercial tool available from Ontoprise GmbH.
[4] http://annotation.semanticweb.org/iswc/documents.html
[5] http://annotation.semanticweb.org/iswc2003/
[6] http://annotation.semanticweb.org/iswc2004/
[7] Cf. also `http://www.semtalk.com/pub/sardinia.htm`.

4.3. Requirements for CREAM

Given the problems with syntax, semantics and pragmatics experienced in the case studies described above, we can now list a more precise set of requirements. Thereby, the principal requirements apply for *a-posteriori annotation* as well as for the *integration of Web page authoring with metadata creation* as follows:

- **Consistency**: Semantic structures should adhere to a given ontology in order to allow for better sharing of knowledge. For example, the use of an attribute should be avoided, where the ontology requires a concept instance.
- **Proper Reference**: Identifiers of instances, *e.g.* of persons, institutes or companies, should be unique. For instance, the metadata generated in the KA2 case study contained three different identifiers for our colleague Dieter Fensel. Thus, knowledge about him could not be gathered using a straightforward query.[8]
- **Avoid Redundancy**: The provision of decentralized knowledge should be possible. However, when annotators collaborate, it should be possible for them to identify (parts of) sources that have already been annotated and to reuse previously captured knowledge in order to avoid laborious redundant annotations.
- **Relational Metadata**: Like HTML information, which is spread on the Web, but related by HTML links, knowledge markup may be distributed, but it should be semantically related. Current annotation tools tend to generate template-like metadata, which is hardly connected, if at all. For example, annotation environments often support Dublin Core [dub, 2001; Dublin Core Metadata Template, 2001; Klarity, 2001], providing a means to state, e.g., the name of authors of a document, but not their IDs[9]. Thus, the only possibility to query for all publications of a certain person requires querying an attribute such as fullname — which is very unsatisfactory in the case of names like "John Smith".
- **Maintenance**: Knowledge markup needs to be maintained. An annotation tool should support the maintenance task. In the remainder of the paper we will provide some infrastructure support for the task.

  In particular we provide a baseline document management system to enable the handling of changing documents, facts, and ontologies in the future.

[8] The reader may see similar effects in bibliography databases. E.g., query for James (Jim) Hendler at the — otherwise excellent — DBLP: http://www.informatik.uni-trier.de/~ley/db/.

[9] In the Web context one typically uses the term 'URI' (uniform resource identifier) to speak of 'unique identifier'.

However, this needs to be further explored (e.g., along the lines elaborated for robust linking in [Phelps and Wilensky, 2000] and Ontology Evolution [Stojanovic *et al.*, 2002a]).

- **Ease of Use**: It is obvious that an annotation environment should be easy to use in order to be really useful. However, this objective is not easily achievable, because metadata creation involves intricate navigation of semantic structures, e.g. taxonomies, properties and concepts.
- **Efficiency**: The effort required to produce metadata is a considerable restrictive factor.The more efficiently a tool supports metadata creation, the more metadata users tend to produce. This requirement is related to the ease of use. It also depends on the automation of the metadata creation process, e.g. on the preprocessing of the document.
- **Multiple Ontologies**: HTML documents in the Semantic Web may contain information that is related to different ontologies. Therefore the annotation framework should cater for concurrent annotation with multiple ontologies.

Our framework, CREAM, which is presented here, targets a comprehensive solution for metadata creation during web page authoring and a-posteriori annotation. The objective is pursued by combining advanced mechanisms for inferencing, fact crawling, document management, meta ontology definitions, metadata re-recognition, content generation, and information extraction. These components are explained in the subsequent sections.

4.4. Design of CREAM

4.4.1. CREAM Modules

The requirements and considerations from Sections 1 to 3.4.1 feed into the design rationale of CREAM. The design rationale links the requirements with the CREAM modules. This results in a N:M mapping (neither functional nor injective). An overview of the matrix is given an Table 4.1, page 58.

- **Document Editor**: The document editor may be conceptually — though not practically — divided into a viewing component and the component for generating content:
  - **Viewer**: The document viewer visualizes the document contents. The metadata creator may easily provide new metadata by selecting pieces of text and aligning them with parts of the ontology. The document

viewer should support various formats[10] (HTML, PDF, XML, etc.). For some formats the following component for content generation may not be available.

  - **Content Generation** The editor also allows the conventional authoring of documents. In addition, instances already available may be dragged from a visualization of the content of the annotation inference server and dropped into the document. Thereby, some piece of text and/or a link is produced taking into account the information from the meta ontology (cf. Section 4.5). The newly generated content is already annotated and the meta ontology guides the construction of further information, e.g. further XPointers are attached to instances.

- **Ontology Guidance and Fact Browser**: The framework needs to be guided by the ontology. In order to allow for sharing of knowledge, newly created annotations must be consistent with a community's ontology. If metadata creators instantiate arbitrary classes and properties the semantics of these properties remains void. Of course the framework must be able to adapt to multiple ontologies in order to reflect different foci of the metadata creators. In the case of concurrent annotation with multiple ontologies there is an ontology guidance/fact browser for each ontology.

  Furthermore, the ontology guidance and the browser for facts already given are important in order to steer metadata creators towards the creation of relational metadata. We have done some preliminary experiments and found that subjects have more problems with creating relationship instances than with creating attribute instances (cf. [Staab *et al.*, 2001b]). Without the ontology they would miss even more cues for assigning relationships between class instances.

  Both ontology guidance/fact browser and document editor/viewer should be easy to use: Drag'n'drop helps to avoid syntax errors and typographical errors and a good visualization of the ontology can help to correctly choose the most appropriate class for instances.

- **Crawler:** The creation of relational metadata must take place *within* the Semantic Web. During metadata creation, subjects must be aware of which entities exist already in their part of the Semantic Web. This is only possible if a crawler makes relevant entities immediately available. Thus, metadata creators may look for a proper reference, i.e. decide whether an entity already has a URI (e.g. whether the entity named "Dieter Fensel" or "D. Fensel" has already been identified by some other metadata creators) and only in this way metadata creators may recognize whether properties have already been instantiated (e.g. whether "Dieter Fensel" has already been

[10]The current prototypical implementation of CREAM focuses on HTML/XHTML

linked to his publications). As a consequence of metadata creators' awareness, relational metadata may be created, because class instances become related rather than only flat templates being filled.

A crawler answers the requirements of proper reference and the avoidance of redundancy. For example, a researcher might crawl the homepages of his community before he starts to annotate his homepage in order to reuse and reference existing metadata. However, a crawler reduces but does not solve the problems since it is obviously not possible to crawl the whole Web.

We have built an OWL/RDF Crawler[11], a basic tool that gathers interconnected fragments of OWL and RDF from the Web and builds a local knowledge base from this data In general, OWL/RDF data may appear in Web documents in several ways. We distinguish between *(i)* pure OWL/RDF (files that have an extension like "*.owl" or "*.rdf"), *(ii)* OWL/RDF embedded in HTML and *(iii)* OWL/RDF embedded in XML. Our RDF Crawler relys on the Jena-API[12] which can deal with the different embeddings of RDF described above. One general problem of crawling is the applied filtering mechanism: Baseline document crawlers are typically restricted by a predefined depth value. Assuming that there is an unlimited amount of interrelated information on the Web (hopefully this will soon hold about OWL/RDF data as well), at some point OWL/RDF fact gathering by the OWL/RDF Crawler should stop. We have implemented a baseline approach for filtering: At the very start of the crawling process and at every subsequent step we maintain a queue of all the URIs we want to analyze. We process them in the breadth-first-search fashion, keeping track of those we have already visited. When the search goes too deep, or the crawler has received sufficient quantity of data (measured as number of links visited or the total Web traffic or the amount of OWL/RDF data obtained) it may quit.

- **Annotation Inference Server**: Relational metadata, proper referencing and the avoidance of redundant annotation require querying for instances, i.e. querying whether and, if so, which instances exist. For this purpose as well as for checking consistency, we provide an annotation inference server in our framework. The annotation inference server reasons on crawled and newly created instances and on the ontology. It also serves the ontological guidance and fact browser, because it allows to query for existing classes, instances and properties.

  The annotation inference server supports multiple ontologies. It distinguishes them into different namespaces. CREAM is based on the model,

[11] RDF Crawler is freely available for download at:
http://projects.semwebcentral.org/projects/owlcrawler/.
[12] http://jena.sourceforge.net/

view, controller paradigm. The annotation inference server constitutes the model, whereas the ontology/fact browser constitutes the view. Every ontology in the annotation server may have a corresponding ontology/fact browser as a view.

We use Ontobroker's [Decker *et al.*, 1999] underlying F-Logic [Kifer *et al.*, 1995] based inference engine SilRI [Decker *et al.*, 1998] as annotation inference server. The F-Logic inference engine combines ordering-independent reasoning in a high-level logical language with a well-founded semantics including multiple namespaces [F-Logic Tutorial, 2004]. However, other schemes like the DAML+OIL iFACT reasoner [Horrocks, 1998; Broekstra *et al.*, 2001] or a fact serving peer [Nejdl *et al.*, 2002] might be exploited, too.

- **Document Management**: Two distinct scenarios exist. Firstly, when annotating one's own Web pages, one knows when they change and can install organizational routines that keep track of annotation and changes of annotation (e.g. on the annotation inference server). In the second scenario, one may want to annotate external resources (cf., e.g., [Nejdl *et al.*, 2001]). Here, redundancy of annotation efforts must be avoided; it is not sufficient to ask whether instances exist at the annotation inference server. When an annotator decides to capture knowledge from a Web page, he does not want to query for all single instances that he considers relevant on this page, rather he wants information, whether and how this Web page has been annotated previously.

  Considering the dynamics of HTML pages on the Web, it is desirable to store foreign web pages one has annotated together with their annotations. Foreign documents for which modification is not possible may be remotely annotated by using XPointer (cf. Section 2.3, [Goble *et al.*, 2001]) as an addressing mechanism. When the foreign Web page changes, the old annotations may still be valid or they may become invalid. The annotator must decide based on the old annotations and based on the changes on the Web page.

  A future goal of document management in our framework will be the semi-automatic maintenance of annotations on foreign Web pages. When only few parts of a document change, pattern matching may propose revisions of old annotations. In our current implementation we use a straightforward file-system based document management approach. OntoMat uses the URI to detect the re-encounter of previously annotated documents and highlights annotations in the old document for the user. Then the user may decide to ignore or even delete the old annotations and create new metadata, he may augment existing data, or he may just be satisfied with what has been previously annotated. In order to recognize that a docu-

ment has been annotated before, but now appears under a different URI, OntoMat computes similarity with existing documents by simple information retrieval methods, e.g. comparison of the word vector of a page. If a similarity is discovered during this process, this is indicated to the user, so that he can check for congruency.

- **Metadata Re-recognition & Information Extraction**: Even with sophisticated tools it is laborious to provide Semantic Annotations. A major goal, therefore, is semi-automatic metadata creation taking advantage of information extraction techniques to propose annotations to metadata creators and, thus, to facilitate the metadata creation task. Concerning our environment we envisage the following major techniques:

  1. First, metadata re-recognition compares existing metadata literals with newly typed or existing text. Thus, the mentioning of the name "Siegfried Handschuh" in the document triggers the proposal that URI, `http://www.aifb.uni-karlsruhe.de/WBS/sha/#Siegfried` is co-referenced at this point.
  2. "Wrappers" may be learned from given markup in order to automatically annotate similarly structured pages (cf., e.g., [Kushmerick, 2000]).
  3. Message extraction like systems may be used to recognize named entities, propose co-reference, and extract some relationship from texts (cf., e.g., [MUC7, 1998; Vargas-Vera *et al.*, 2001]).
  4. Linguistic Patterns may be used to categorize instances with regards to a given ontology. This approach combines the idea of using linguistic pattern to identify certain ontological relations as well as using the Web as a big corpus to overcome data sparseness problems. The PANKOW approach (Pattern-based Annotation through Knowledge on the Web) is explained in Section 5.2.

  Parts of this component has been realized by using the Amilcare information extraction system (cf. 5.1.2)[13], but it is not yet available in the download version of OntoMat.

- **Deep Annotation Module** The Deep Annotation Module allows to annotate the hidden Web, viz., Web pages that are generated from a database. A detailed description is given in Section 6.

- **Meta Ontology**: The purpose of the meta ontology is the separation of ontology design and use. It is specifically explained in Section 4.5.

[13] http://www.dcs.shef.ac.uk/~fabio/Amilcare.html

Besides the requirements that constitute single modules, one may identify functions that cross module boundaries:

- **Storage**: CREAM supports three different methods of storage: i) the annotations will be stored inside the document that is in the document management component or ii) they are stored in a separate file, iii) alternatively or simultaneously it is also possible to store them in the annotation inference server.
- **Replication**: We provide a simple replication mechanism by crawling annotations into our annotation inference server. Then inferencing can be used to rule out formal inconsistencies.

4.4.2. Architecture of CREAM

The architecture of CREAM is depicted in Figure 4.1. The design of the CREAM framework pursues the idea to flexibility and openness. Therefore, OntoMat, the implementation of the framework, comprises a plug-in structure, which is flexible with regard to adding or replacing modules. Document viewer/editor and ontology guidance/fact browser together constitute the major part of the graphical user interface. Thanks to the plug-in structure they can be replaced by alternative viewers.

Further capabilities are provided through plugins that establish connections, e.g. one might provide a plug-in for a connection to a commercial document management system.

The core OntoMat already comes with some basic functionalities. For instance, one may work without a plug-in for an annotation inference server, because the core OntoMat provides a simple means of navigating the taxonomy *per se*.

The core OntoMat, which is downloadable, consists of Ontology Guidance and Fact browser, a document viewer/editor, and an internal memory data-structure for the ontology and metadata. However, one only gets the fully-fledged semantic capabilities (e.g. datalog reasoning or subsumption reasoning) when one uses a plug-in connection to a corresponding annotation inference server.

4.5. Meta Ontology

A meta ontology is needed to describe how classes, attributes and relationships from the domain ontology should be used by the CREAM environment. Hence, it describes what role the properties in the domain ontology have in the annotation

Table 4.1.: Design Rationale — Linking Requirements with CREAM Modules.

Requirement / General Problem	Document Editor: Viewer	Document Editor: Content Generation	Replication: Ontology Guidance	Storage: Crawler	Storage: Annotation Inference Server	Document Management	Metadata Recognition	General Information Extraction	Meta Ontology
Consistency		X	X		X				X
Proper Reference		X		X	X		X		X
Avoid Redundancy		X		X	X	X			
Relational Metadata			X	X	X		X		
Maintenance		X				X	X	X	
Ease of use	X		X				X	X	X
Efficiency	X	X	X	X	X	X	X	X	X
Multiple Ontologies			X		X				

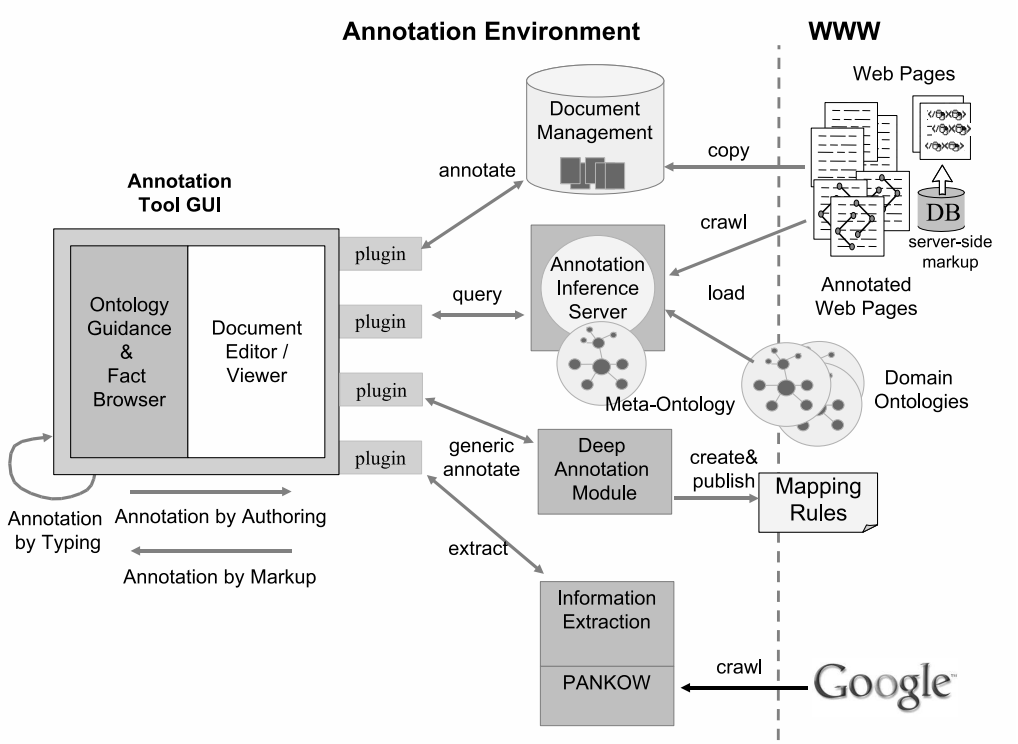

Figure 4.1.: Architecture of CREAM.

system. In particular, we have recognized the urgent need for the meta ontology characterizations detailed in Sections 4.5.1 to 4.5.3.

The meta ontology is given with the annotation system. The annotator must establish the connection of an ontology with the meta ontology. The effort for the annotator is reduced by default settings for this connection, such as the name attribute is by default the label of the instance. The task of connecting the ontology with the meta-ontology is supported by the user interface, e.g. the annotator can change the role of the properties in the Ontology and Fact Browser.

Thus, the ontology describes what the semantic data should look like and the meta ontology connected to the ontology describes how the ontology is used by the annotation environment to actually create semantic data.

This modularization into ontology and meta ontology fulfills the requirement that it should be possible to define a new ontology or to reuse an existing one quite independently from the way it is used to create metadata by Web page authoring

and annotation.

The reader may note that the descriptions of how the meta ontology influences the interaction with the ontology, which are given in this section, depend to some extent on the modes of interaction described in Section 4.6 — and vice versa.

4.5.1. Label

The specification of RDFS [Brickley and Guha, 2004] provides a rdfs:label as a human-readable version of a resource name. Analogously, one wants to assign an instance with a human-readable name even if it instantiates a class from a given ontology that does not use the property rdfs:label *per se*.

For instance, assume that (part of) the RDF(S) ontology definition is as follows:

```
:ssn a rdf:Property;
     rdfs:comment "Social Security Number";
     rdfs:range xml-schema:Integer;
     rdfs:domain :Person.

:fullname a rdf:Property;
     rdfs:comment "Last Name, First Name, Middle Name";
     rdfs:range rdf-schema:Literal;
     rdfs:domain :Person.
```

Then, one might want to state that the property fullname rather than the property ssn takes the role of rdfs:label for class Person. We may link the meta ontology (the relevant piece here is RDFS:LABEL) with the ontology proper by:

```
:label1 a meta_onto:Label;
    meta_onto:about_concept :Person;
    meta_onto:about_attribute :fullname.
```

Now, the authoring and annotation environment may exploit this additional piece of information at (at least) two points of interaction:

1. Instance Generation: When a *new instance* is about to be created and some piece of text has been chosen to represent the name, CREAM creates a new URI and automatically assigns this piece of text to the attribute recorded as rdfs:label, i.e. here fullname.

2. Content Generation: When an instance is selected for generating content of a Web page, the generated text is produced by the rdfs:label attribute. By this method we may produce text that is meaningful to humans with little interaction, because the author need not to specify the attribute that should be used but he may just refer to the instance.

One may note that another person, e.g. from administration, could choose instead:

```
:label1 a meta_onto:Label;
    meta_onto:about_concept :Person;
    meta_onto:about_attribute :ssn.
```

Thus, this person would create Web page content referring to social security numbers when authoring with existing instances and he would create new instances of Person from given social security numbers and not from names.

The reader may note from this scriptsize example that

- The difference between more or less metadata creation effort is often just one click.[14] For example, without the meta ontology description the following steps would have been necessary for instance creation: an instance creation with an appropriate label, filling the attribute "fullname", filling of the attribute "ssn".
- The connection between ontology and meta ontology is not an objective one. Rather their linkage depends on the way an ontology is used in a particular metadata creation scenario.

Statements equivalent to the last text passage hold for the following meta ontology descriptions.

4.5.2. Default Pointing

For many instances that are to be created it is desirable to point to the Web page from which they "originated". Analogously to the way that rdfs:label is used, we use three types of definitions in order to specify the default pointing behavior for class instances, exploiting the XPointer working draft [DeRose *et al.*, 2001].

Consider the meta ontology properties CREAM:UNIQUEDPOINTER, CREAM:AUTODPOINTER, and CREAM:AUTOUNIQUEDPOINTER. If a property is rdfs:subPropertyOf one of the following interactions take place during annotation or authoring:

1. Instance Generation: When a new instance is generated and a property of that instance is of type CREAM:AUTODPOINTER or CREAM:AUTOUNIQUEDPOINTER, an XPointer to the current (part of the) Web page will be automatically added into the corresponding slot of the instance.

[14] We conjecture that the one click difference may distinguish between success and failure of a tool.

2. Content Generation: When an instance is used for generating Web page content, the attribute containing the XPointer is offered for link generation. When the attribute is of type CREAM:UNIQUEDPOINTER or CREAM:AUTO-UNIQUEDPOINTER indicating uniqueness, a link with text corresponding to rdfs:label and HRef corresponding to the XPointer will be automatically generated.

For instance, one may model in the ontology that a Person comes with properties hasHomepage and fullname and in the instantiation of the meta ontology that hasHomepage is a subproperty of cream:uniqueDPointer and fullname a subproperty of rdfs:label. During annotation of people homepages, the label and pointer mechanisms automate, *(i)*, the generation of unique IDs with reasonable labels, *(ii)*, the creation of pointers to people's homepages, and, *(iii)*, the correct linking between people mentioned on the different homepages. Like with rdfs:label, the linkage between meta ontology and ontology proper may depend on the actual usage scenario.

4.5.3. Property Mode

The property mode distinguishes between different roles, different ways in which the property should be treated by the metadata creation environment:

1. **Reference**: In order to describe an object, metadata may simply point to a particular place in a resource, e.g. a piece of text or a piece of multimedia. For instance, one may point to a particular place at `http://www.whitehouse.gov` in order to refer to the current U.S. president. Even when the presidency changes the metadata may remain up-to-date.

   References are particularly apt to point to parts of multimedia, e.g. to a part of a scalable vector graphics.

   The reader may note that every default pointer is a reference, but not vice versa. For example, assume a property hasName, which is a reference – it points to a particular place – to a name on the Web page but not modeled as a default pointer, therefore the pointer is not automatically created or considered as unique.

2. **Quotation**: In order to describe an object, metadata may copy an excerpt out of a resource. In contrast to the mode "reference", a quotation does not change when the corresponding resource changes. A copy of the string "Bill Clinton" as president of the United States in 1999 remains unchanged even if its original source at `http://www.whitehouse.gov` changes or is abandoned.

3. **Unlinked Fact**: An unlinked fact describes an object, but is not in any way stemming from or depending on a resource. Unlinked facts are typical for comments. They are also very apt to be combined with references in order to elucidate the meaning or name of a graphic or piece of multimedia.

   For instance, there may be a reference pointing to the picture "Guernica" (`http://www.grnica.swinternet.co.uk/guernica.jpg`) and attributes that are specified to be unlinked facts. The unlinked fact-attributes may be filled by someone who knows Picasso's paintings, e.g. with specifications like "Guernica" or "Spanish Civil War".

The meaning of the property mode may slightly overlap with the definition of the range of a property, e.g. an unlinked fact is typically only used with an attribute that has a literal as its range. The reason is that a pointer may be used as a URI (e.g. [Goble *et al.*, 2001]) and a URI should typically not appear in a literal (though this is not prohibited). We separate the two aspects, because not every URI is a pointer and it sometimes makes sense to specify the value of a literal by a pointer. Thus, the definition of the range of a property as reference, quotation or unlinked fact may be considered orthogonal to the range of a property being a literal or a resource.

4.5.4. Further Meta Ontology Descriptions

In concluding this section, we ask the reader to note that the list of possibly useful meta ontology descriptions sketched here is by no means conclusive. Rather, we envision (and partially support) the use of meta ontology descriptions for purposes such as:

- Knowledge acquisition from templates: For example we describe in SWOBIS (http://tools.semanticweb.org/) software tools with metadata (cf. Figure 4.2). For each instance, there are a number of attributes required to specify a software tool. The meta ontology allows the definition of attribute instances as being required attribute instances. This information is used to automatically generate a template like interface for OntoMat — one that is similar in its structure to a Dublin Core template. This approach is akin to the way that Protege allows the construction of knowledge acquisition interfaces [Noy *et al.*, 2000].

- Authoring of dynamic ontology and metadata-based Web pages (also cf. OntoWebber [Jin *et al.*, 2001]).

- Provisioning of metametadata, e.g. author, date, time, and location of an annotation. Though this may appear trivial at first sight, this objective easily clashes with several other requirements, *e.g.* ease of use of metadata

generation *and* usage. Eventually, it needs a rather elaborate meta ontology, containing not only static, but also dynamic definitions, *i.e.* rules. For example, to describe that a person A created an instance X, a second person B created an instance Y and a third person C created a relationships instance between X and Y, it is necessary to reify the relationship instance. In order to directly use the relationship instance, it should be translated by a rule into an unreified relationship instance.

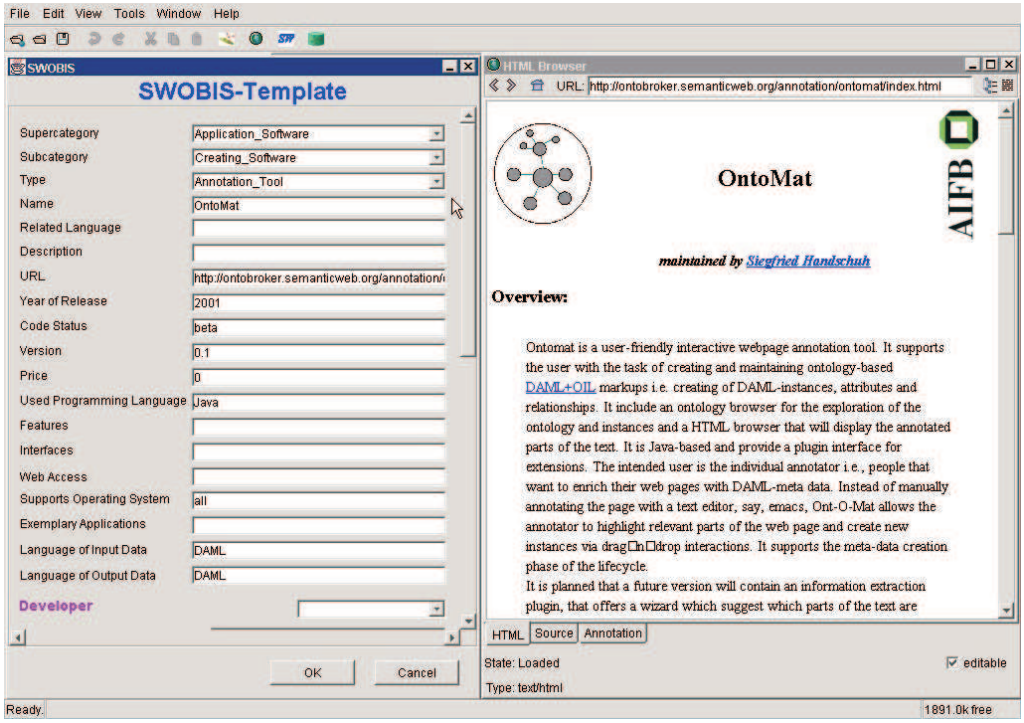

Figure 4.2.: SWOBIS template.

4.6. Modes of Interaction

The metadata creation process in OntoMat is actually supported by three types of interaction with the tool (also cf. Figure 4.1):

1. Annotation by Typing Statements: This involves working almost exclusively within the ontology guidance/fact browser.

2. Annotation by Markup: This mostly involves the reuse of data from the document editor/viewer in the ontology guidance/fact browser.

3. Annotation by Authoring Web Pages: This mostly involves the reuse of data from the fact browser in the document editor.

In order to clarify the different role of the three types of interaction, we describe here how they differ for generating three types of metadata:

1. Generating instances of classes
2. Generating attribute instances
3. Generation relationship instances

4.6.1. Annotation by Typing

Annotation by typing is almost purely based on the ontology guidance/fact browser (cf. Section 4.4) and the generated templates (cf. Section 4.5.4). Basically, the more experienced user navigates the ontology and browses the facts, the less experienced user should use templates instead. The user generates metadata (class instances, attribute instances, relationship instances) that are completely independent from the Web page currently being viewed.

The specification of the rdfs:label property allows to the creation (or re-discovery) of instances by typing where the URI is given and the rdfs:label property is filled with the text. The specification of a default pointer by the meta ontology may associate newly created instances with the currently marked passage in the text.

In addition, the user may drag-and-drop around instances that are already in the knowledge base in order to create new relationship instances (cf. arrow #0 in Figure 4.3).

4.6.2. Annotation by Markup

The basic idea of annotation by markup is the usage of marked-up content in the document editor/viewer for instance generation.

1. Generating class instances: When the user drags a marked up piece of content onto a particular concept from the ontology, a new class instance is generated. If the class definition comes with a meta ontology description of a rdfs:label a new URI is generated, and the corresponding property is assigned the marked up text (cf. arrow #1 in Figure 4.3).

   For instance, marking "Siegfried Handschuh" and dropping this piece of text on the concept PhDStudent creates a new URI, instantiates this URI as belonging to PhDStudent and assigns "Siegfried Handschuh" to the swrc:name slot of the new URI. In addition, default pointers may be provided.

2. Generating attribute instance: In order to generate an attribute instance the user simply drops the marked up content into the corresponding table entry (cf. arrow #2 in Figure 4.3). Depending on whether the attribute is specified as reference or quotation the corresponding XPointer or the content itself is filled into the attribute.

3. Generating relationship instance: In order to generate a relationship instance the user simply drops the marked up content onto the relation of a pre-selected instance (cf. arrow #3 in Figure 4.3). Like in "class instance generation" a new instance is generated and connected with the pre-selected instance.

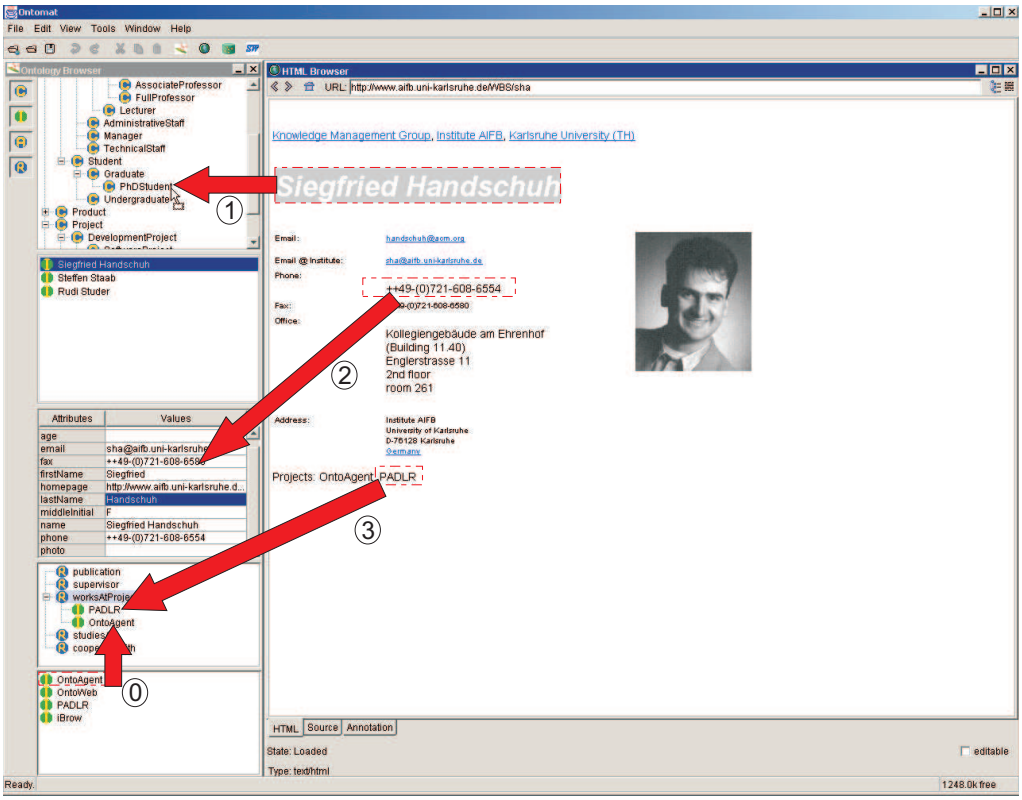

Figure 4.3.: Annotation example.

4.6.3. Annotation by Authoring

The third major process is authoring Web pages and metadata together. There are two modi for authoring: *(i)*, authoring by using ontology guidance and fact

browser for content generation and, *(ii)*, authoring with the help of metadata re-recognition or — more general — information extraction. As far as authoring is concerned, we have only implemented *(i)* so far. However, we wish to point out that even very simple information extraction mechanisms, i.e. metadata re-recognition (cf. Section 4.4) may help the author to produce consistent metadata.

Authoring with Content Generation By inverting the process of markup (cf. Figure 4.1), we may reuse existing instance description, like labels or other attributes:

1. Class instances: Dropping class instances from the fact browser into the document creates text according to their labels and — if possible — links (cf. arrow #1 in Figure 4.4).
2. Attribute instances: Dropping attribute instances from the fact browser into the document (cf. arrow #2 in Figure 4.4) generates the corresponding text (for quotations or unlinked facts) or even linked text (for references).
3. Relationship instances: Dropping relationship instances from the fact browser into the document generates simple "sentences". For instance, the dropping of the relationship COOPERATESWITH between the instances corresponding to Rudi and Steffen triggers the creation of a small piece of text (cf. arrow #3 in Figure 4.4). The text corresponds to the instance labels plus the label of the relationship (if available), e.g. "Rudi Studer cooperates with Steffen Staab". Typically, this piece of text will require further editing.

Further mechanisms, like the creation of lists or tables from selected concepts (e.g. all Persons), still need to be explored.

4.7. Conclusion

CREAM is a comprehensive framework for creating semantic metadata, relational metadata in particular — the foundation of the future Semantic Web. CREAM builds on comprehensive experience we have collected in several case studies on creating metadata for the Semantic Web. CREAM supports three modes of interaction for *a posteriori* annotation and for creating metadata while authoring a Web page. In order to avoid problems with syntax, semantics and pragmatics, CREAM employs a rich set of modules including inference services, crawler, document management system, ontology guidance/fact browser, and document editors/viewers. Process issues of the annotation/authoring task are modularized from content descriptions by a meta ontology.

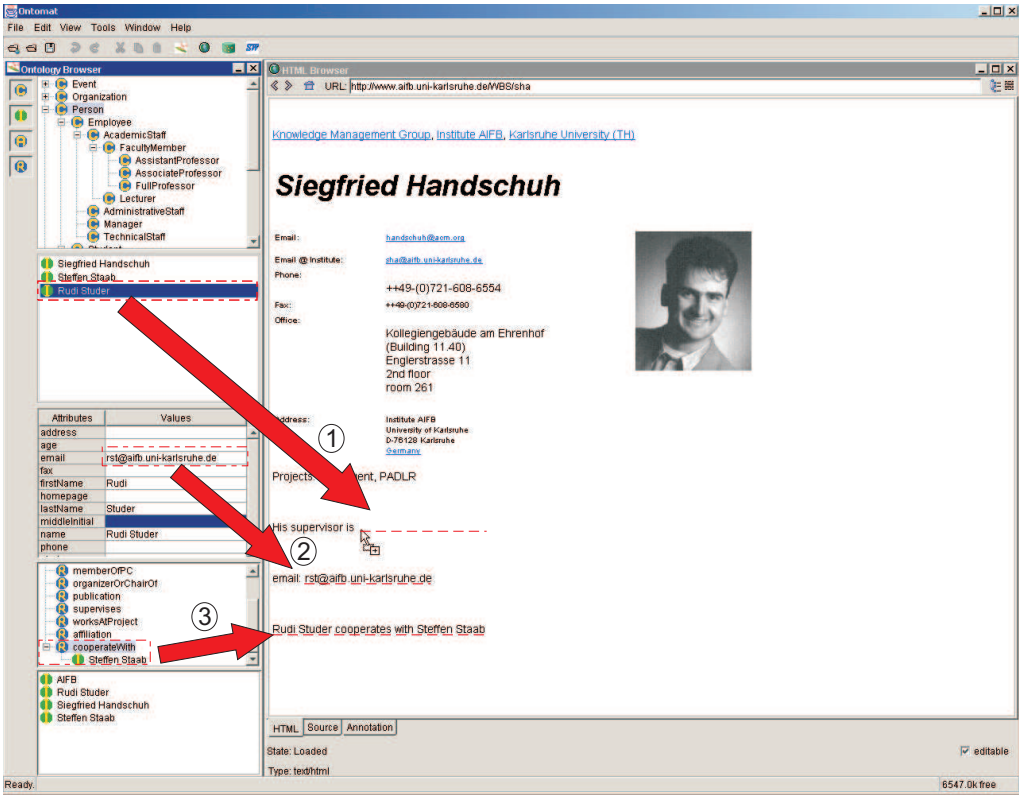

Figure 4.4.: Annotation by Authoring example.

OntoMat is the reference implementation of the CREAM framework. It is Java-based and provides a plug-in interface for extensions for further applications.

5. Semi-Automatic Annotation

This Chapter describes techniques developed for semi-automatic annotation. These techniques are realized extensions, viz. plugins, of our annotation framework *CREAM*. The first extension – *S-CREAM* (Semi-automatic CREAtion of Metadata) – allows for creation of metadata and is trainable for a specific domain (Section 5.1) The second extension – *PANKOW* (Pattern-based Annotation through Knowledge on the Web), focuses on a method that combines the use of linguistic patterns to identify instances with the use of the WWW as a large corpus via a search engine (Section 5.2).

The remainder of of Section 5.1 presents, firstly, the information extraction scenario (Section 5.1.1). Secondly, Amilcare – the information extraction component(5.1.2) is described. Thirdly, the focus is on the problems of integration (Section 5.1.3–5.1.4). Finally a concluding discussion follows in Section 5.1.5. The structure of this Section 5.2 is as follows: Section 5.2.1 describes the principal procedure of PANKOW. Section 5.2.2 details the core algorithmic approach to categorizing instances from text. Then, we present the integration into CREAM/OntoMat (Section 5.2.3).

References: Section 5.1 is mainly based on [Handschuh *et al.*, 2002] and Section 5.2 is mainly based on [Cimiano *et al.*, 2004].

5.1. Information Extraction

The Semantic Web builds on metadata describing the contents of Web pages. In particular, the Semantic Web requires relational metadata, i.e. metadata that describe how resource descriptions instantiate class definitions and how they are semantically interlinked by properties. In order to support the construction of relational metadata, an annotation and authoring framework (CREAM) and a tool (OntoMat) that implements this framework is provided in Chapter 4.

Nevertheless, providing plenty of relational metadata by annotation, i.e. conceptual mark-up of text passages, remained a laborious task. In the architecture Section 4.4.1 the idea that wrappers and information extraction components could be used to facilitate the work was presented, which is described in this section as fully-fledged integration that deals with all the conceptual difficulties. Therefore, the CREAM framework is extended to S-CREAM (Semi-automatic

CREAtion of Metadata), an annotation framework that integrates a learnable information extraction component (viz. Amilcare [Ciravegna, 2001a]).

Amilcare is a system that is able to learn its information extraction rules from manually marked-up input. S-CREAM aligns conceptual markup, which defines relational metadata, (such as provided through OntoMat) with semantic and indicative tagging (such as produced by Amilcare).

There two major type of problems that have been solved for this purpose:

1. When comparing the desired relational metadata from manual markup and the semantic tagging provided by information extraction systems, one recognizes that the output of this type of systems is underspecified for the purpose of the Semantic Web. In particular, the nesting of relationships between different types of concept instances is undefined and, hence, more comprehensive graph structures may not be produced (further elaboration in Section 5.1.3). To overcome this problem, a new processing component, viz. a lightweight module for discourse representation (Section 5.1.4) is introduced.

2. Semantic tags do not correspond one-to-one to the conceptual description (Section 5.1.1 and 5.1.4).

   - Semantic tags may have to be turned into various conceptual markup, e.g., as concept instances, attribute instances, or relationship instances.
   - For successful learning, Amilcare sometimes needs further indicative tags (e.g., syntactic tags) that do not correspond to any entity in a given ontology, but that may only be exploited within the learning cycle.

5.1.1. Process of ontology-based Information Extraction

This section gives a general overview of the information extraction process. The information extraction process can be divided into a setup step, a training phase, with three steps, and into a annotation phase, with two steps, (depicted in Figure 5.1):

Step 1: Project Definition

The first step is the project definition. A domain ontology is the basis for the annotation of different types of documents. Likewise a certain kind of document can be annotated in reference to different ontologies. Therefore, a project defines the combination of a domain ontology (e.g. the GETESS ontology about tourism)

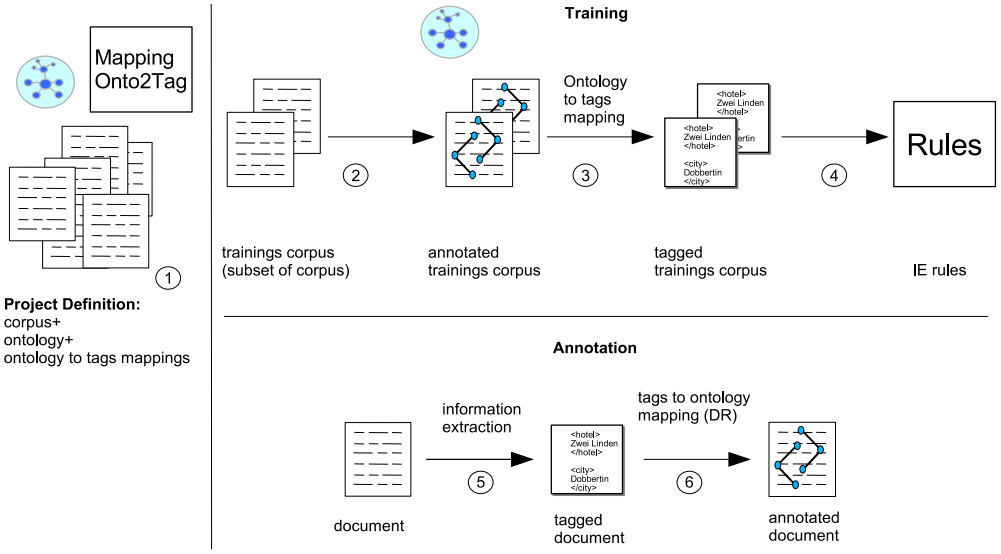

Figure 5.1.: The Process of Information Extraction

with a corpus. In this case a corpus is a collection of similar documents, for example hotel homepages (http://all-in-all.de). In addition, the mapping of the Ontology to the Amilcare tags has to be defined, i.e. as follows:

- Concepts: Concepts are mapped to tags using the name of the concept, e.g. the concept with the name "Hotel" results in a <hotel> tag.
- Hierarchy: The concepts of the ontology are hierarchically structured. OntoMat offers to map a concept to multiple tags in order to emulate the different levels of conceptualization, e.g. the concept "Hotel" may be mapped to the tags <company>, <accommodation>, and <hotel>.
- Attributes: The mapping of attributes to tags is a tradeoff between a specific and a general naming. The specific naming simplifies the mapping back to the ontology, but it results in more complex extraction rules. These rules are less general and less robust. For example, a specific naming of the attribute "phone" would result in tags like <hotel_phone>, <room_phone>, and <person_phone> in comparison to the general tag <phone>. Therefore, the user must decide for each attribute, whether the naming is adequate, as it influences the learning results.

Step 2: Annotation

The user can create the training corpus with OntoMat. The user has to choose a proper sample of the corpus and annotate these documents. The document is annotated by OntoMat with RDF facts. These facts are linked by an XPointer description to the annotated text part.

Alternatively, if enough annotated documents already exist in the Web, the user can starts a crawl of Web pages with OntoMat and collects the documents. The crawl can be limited in this case to documents which are annotated with the desired ontology. If necessary, the ontology sub-set and the mapping to the Amilcare tags must be re-adjusted according to the annotations found in the crawled documents. Subsequently, the desired type of document must still be checked manually.

Step 3: Ontology to Amilcare tags mapping

Because Amilcare needs XML tagged files as a corpus, the RDF annotations are transformed into corresponding XML tags according to the mapping done in the project definition. Only these tags are used to train. Other tags like HTML tags will be used as contextual information.

Step 4: Learning

The learning phase is executed by Amilcare, which is embedded as a plugin into OntoMat. Amilcare processes each document of the corpus and generates extraction rules as described in Section 5.1.2. After the training Amilcare stores the annotation rules in a certain file which belongs to the project.

Step 5: Shallow Information Extraction

Now it is possible to use the induced rules for semi-automatic annotation. Based on the rules the Amilcare plugin produces XML annotation results (cf. A1 in Figure 5.3).

Step 6: Semi-automatic Annotation

As shown in Figure 5.3, a mapping (A2) is done from the flat markup to the conceptual markup in order to create new RDF facts (A3). This mapping is undertaken by the discourse representation (cf. Section 5.1.4). This mapping results in several automatically generated proposals for the RDF annotation of the document. The user can interact with these annotation proposals using three

different automation methods: (i) highlighting the annotation candidates or (ii) interactive suggestion for each annotation or (iii) a first full automatic annotation of the document with subsequent refinement.

- Highlighting mode: The user opens a document he would like to annotate in the OntoMat document editor. Then the highlighting mode marks all annotation candidates with a colored underline. The user can decide if he wishes to use this hint for an annotation or not.
- Interactive mode: This mode is also designed for individual document processing. The interactive suggestion is a step by step process. Each possible annotation candidate will be suggested to the user and he can refuse, accept or change the suggestion in a dialog window.
- Automatic mode: The full automatic approach is useful, if there are a number of documents that need to be annotated, so that the task can be performed in a batch mode. All selected documents are annotated automatically. The user can later refine all the annotations in the document editor.

The result is an RDF-annotated document.

5.1.2. Amilcare

Amilcare is a tool for adaptive Information Extraction from text (IE) designed for supporting active annotation of documents for Knowledge Management (KM). It performs IE by enriching texts with XML annotations, i.e. the system marks the extracted information with XML annotations. The only knowledge required for porting Amilcare to new applications or domains is the ability of manually annotating the information to be extracted in a training corpus. No knowledge of Human Language technology is necessary. Adaptation starts with the definition of a tagset for annotation.

Then users have to manually annotate a corpus for training the learner. As will be later explained in detail, OntoMat may be also used as the annotation interface to annotate texts in a user friendly manner. In this case, OntoMat provides user annotations as XML tags to train the learner. Amilcare's learner induces rules that are able to reproduce the text annotation.

Amilcare can operate in two modes: **training**, used to adapt it to a new application, and **extraction**, used to actually annotate texts. In both modes, Amilcare first of all preprocesses texts using Annie, the shallow IE system included in the Gate package ([Maynard *et al.*, 2002], www.gate.ac.uk). Annie performs text tokenization (segmenting texts into words), sentence splitting (identifying

sentences) part of speech tagging (lexical disambiguation), gazetteer lookup (dictionary lookup) and named entity recognition (recognition of people and organization names, dates, etc.).

When operating in training mode, Amilcare induces rules for information extraction. The learner is based on $(LP)^2$, a covering algorithm for supervised learning of IE rules based on Lazy-NLP [Ciravegna, 2001a] [Ciravegna, 2001c]. This is a wrapper induction methodology [Kushmerick, 1997] that, unlike other wrapper induction approaches, uses linguistic information in the rule generalization process. The learner starts inducing wrapper-like rules that make no use of linguistic information, where rules are sets of conjunctive conditions on adjacent words. Then the linguistic information provided by Annie is used in order to generalize rules: conditions on words are substituted with conditions on the linguistic information (e.g. condition matching either the lexical category, or the class provided by the gazetteer, etc. [Ciravegna, 2001c]). All the generalizations are tested in parallel by using a variant of the AQ algorithm [Mickalski *et al.*, 1986] and the best k generalizations are kept for IE. The idea is that the linguistic-based generalization is used only when the use of NLP information is reliable or effective. The measure of reliability here is not linguistic correctness (immeasurable by incompetent users), but effectiveness in extracting information using linguistic information as opposed to using shallower approaches. Lazy NLP-based learners learn which is the best strategy for each information/context separately. For example they may decide that using the result of a part of speech tagger is the best strategy for recognizing the location in holiday advertisements, but not to spot the hotel address. This strategy is quite effective for analyzing documents with mixed genres – quite a common situation in Web documents[Ciravegna, 2001b].

The learner induces two types of rules: tagging rules and correction rules. A tagging rule is composed of a left hand side, containing a pattern of conditions on a connected sequence of words, and a right hand side that is an action inserting an XML tag in the texts. Each rule inserts a single XML tag, e.g. </hotel>. This makes the approach different from many adaptive IE algorithms, whose rules recognize whole pieces of information (i.e. they insert both <hotel> and </hotel>, or even multi slots. Correction rules shift misplaced annotations (inserted by tagging rules) to the correct position. They are learnt from the mistakes made in attempting to re-annotate the training corpus using the induced tagging rules. Correction rules are identical to tagging rules, but (1) their patterns also match the tags inserted by the tagging rules and (2) their actions shift misplaced tags rather than adding new ones. The output of the training phase is a collection of rules for IE that are associated with the specific scenario.

When working in extraction mode, Amilcare receives as input a (collection of) text(s) with the associated scenario (including the rules induced during the training phase). It preprocesses the text(s) by using Annie, subsequently applies it

rules, and returns the original text with the added annotations. The Gate annotation schema is used for annotation [Maynard *et al.*, 2002].

Amilcare is designed to accommodate the needs of different user types. While naive users can build new applications without delving into the complexity of Human Language Technology, IE experts are provided with a number of facilities for tuning the final application. Induced rules can be inspected, monitored and edited to obtain some additional accuracy, if needed. The interface also allows balancing precision (P) and recall (R). The system is run on an annotated unseen corpus and users are presented with statistics on accuracy, together with details on correct matches and mistakes (using the MUCscorer [Douthat, 1998] and an internal tool). Retuning the P&R balance does not generally require major retraining. Facilities for inspecting the effect of different P&R balances are provided.

5.1.3. Synthesizing S-CREAM

To synthesize S-CREAM from the existing frameworks CREAM and Amilcare, let us consider their core processes in terms of input and output, as well as the process of the as yet undefined S-CREAM. Figure 5.3 surveys the three processes.

The **first process** is indicated by a circled A1. It is information extraction, e.g. provided by Amilcare [Ciravegna, 2001a], which digests a document and produces either an XML tagged document or a list of XML tagged text snippets (cf. Table 5.1(a)).

The **second process** is indicated by a circled M. It is manual annotation and authoring of metadata, which turns a document into relational metadata that corresponds to the given ontology (as described in detail in 4.6). For instance, an annotator may use OntoMat to describe how the relationships listed in Table 5.1(b) appear on the homepage of hotel "Zwei Linden" (cf. Figure 5.2)

The obvious questions that emerge at this point are: Is the result of Table 5.1(a) equivalent to the one in Table 5.1(b)? How can Table 5.1(a) be turned into the result of Table 5.1(b)? The latter is a requirement for the Semantic Web.

The "Semantic Web answer" to this is: The difference between Table 5.1(a) and Table 5.1(b) is analogous to the difference between a very particular serialization of data in XML and a RDF structure. This means that, assuming a very particular serialization of information on Web pages, the Amilcare tags can be specified so precisely[1] that indeed Table 5.1(a) can be rather easily mapped into Table 5.1(b). The only requirement may be a very precise specification of tags, e.g. "43,46" may need to be tagged as <lowerprice-of-doublebedroom-of-hotel>43,46</lowerprice-of-doubleroom-of-hotel> in order to cope with its relation to a double room of a hotel.

[1] We abstract here from the problem of correctly tagging a piece of text.

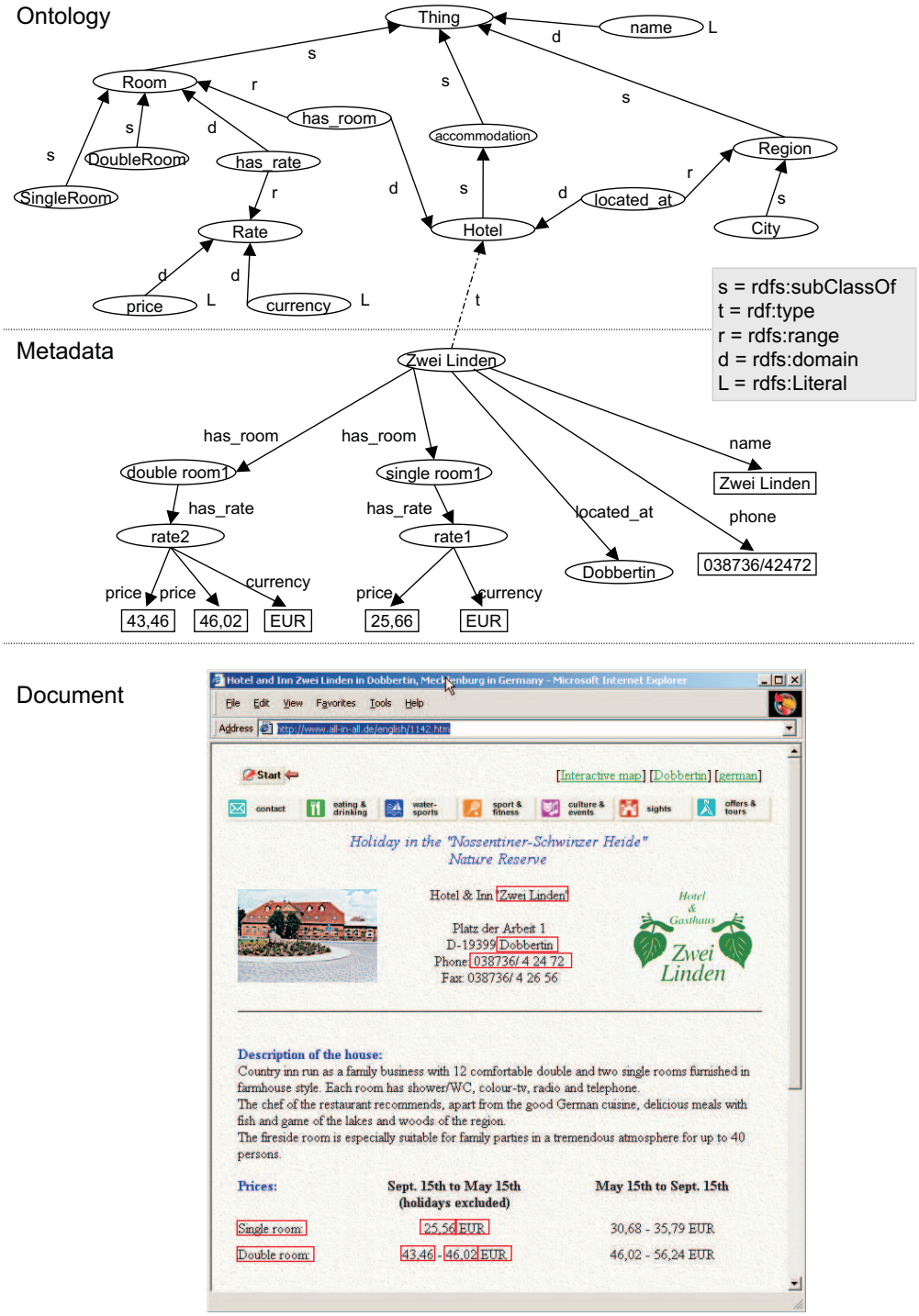

Figure 5.2.: Annotation example

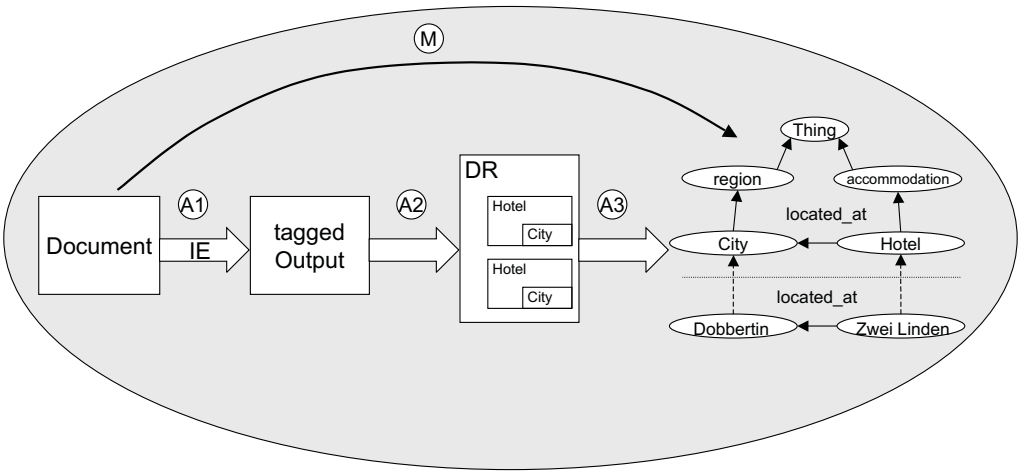

Figure 5.3.: Two Ways to the Target: Manual and Automatic Annotation

<hotel>Zwei Linden</hotel>	:Zwei_Linden a :Hotel.
	:Zwei_Linden :name "Zwei Linden".
	:Zwei_Linden :located_at :Dobbertin.
<city>Dobbertin</city>	:Dobbertin a :City.
	:Dobbertin :name "Dobbertin".
	:Zwei_Linden :has_room :Single_room_1.
<singleroom>Single room</singleroom>	:Single_room_1 a :Single_Room.
	:Single_room_1 :has_rate :Rate2.
	:Rate2 a :Rate.
<price>25,66</price>	:Rate2 :price "25,66".
<currency>EUR</currency>	:Rate2 :currency "EUR".
	:Zwei_Linden :has_room :Double_room_3.
<doubleroom>Double room</doubleroom>	:Double_room_3 a :Double_Room.
	:Double_room_3 :has_rate :Rate4.
	:Rate4 a :Rate.
<lowerprice>43,46</lowerprice>	:Rate4 :price "43,46".
<upperprice>46,02</upperprice>	:Rate4 :price "46,02".
<currency>EUR</currency>	:Rate4 :currency "EUR".
...	...
(a) Amilcare	(b) OntoMat

Table 5.1.: Comparison of Output: Amilcare versus manual OntoMat

The "Natural Language Analysis answer" to the above questions is: Learnable information extraction approaches like Amilcare do not have an explicit discourse model for relating tagged entities — at least for now. Their implicit discourse model is that each tag corresponds to a place in a template[2] and every document (or document analogon) corresponds to exactly one template. This is fine as long as the discourse structures in the text are simple enough to be mapped into the template and from the template into the target RDF structure.

In practice, however, the assumption that the underlying graph structures/discourse structures are quite similar, often does not hold. Then the direct mapping from XML tagged output to target RDF structure becomes awkward and difficult to accomplish.

The **third process** given in Figure 5.3 is indicated by the composition of A1, A2 and A3. It bridges from the tagged output of the information extraction system to the target graph structures via an explicit discourse representation. Our discourse representation is based on a very lightweight version of Centering [Grosz and Sidner, 1986; Strube and Hahn, 1999] and explained in the next section.

5.1.4. Discourse Representation (DR)

The principal task of discourse representation is to describe coherence between different sentences. The core idea is that, during the interpretation of a text (or, more generally, a document), there is always a logical description (e.g., an RDF(S) graph) of the content that has been read so far. The current sentence updates this logical description by:

1. **Introducing new discourse referents**: i.e. introducing new entities, e.g., finding the term 'Hotel & Inn "Zwei Linden" ' to denote a new object.
2. **Resolving anaphora**: i.e. describing denotational equivalence between different entities in the text, e.g. 'Hotel & Inn "Zwei Linden" ' and 'Country inn' refers to the same object.
3. **Establishing new logical relationships**: i.e. relating the two objects referred to by 'Hotel & Inn "Zwei Linden" ' and 'Dobbertin' via locatedAt.

The problem with information extraction output is that it is not clear what constitutes a new discourse entity. Though information extraction may provide some typing (e.g. <city>Dobbertin</city>), it does not describe whether this constitutes an attribute value (of another entity) or an entity of its own. Neither

[2] A template is like a single tuple in an unnormalized relational database table, where all or several entries may have null values.

do information extraction systems like Amilcare treat coherence between different pieces of tagged text.

Grosz & Sidner [Grosz and Sidner, 1986] devised *centering* as a theory of text structures that separate text into segments that are coherent to each other. The principal idea of the centering model is to express fixed constraints as well as "soft" rules which guide the reference resolution process. The fixed constraints denote what objects, if any, are available for resolving anaphora and establishing new logical inter-sentential relationships, while soft rules give a preference ordering to these possible antecedents. The main data structure of the centering model is a list of *backward-looking centers*, $C_b(U_k)$ for each utterance U_k. The backward-looking centers $C_b(U_k)$ constitutes a ranked list of what is available and what is preferred for *resolving anaphora* and for *establishing new logical relationships* with previous sentences.

The centering model allows for relating a given entity in utterance U_k to one of the backward-looking centers, $C_b(U_{k-1})$. For instance, when reading "The chef of the restaurant" in Figure 5.2 the centering model allows relationships with "Country inn", but not with "Dobbertin".

The drawback of the centering model is that, firstly, it has only been devised for full text and not for semi-structured text such as appears in Figure 5.2 and, secondly, it often needs more syntactic information than shallow information extraction can provide.

Therefore, we use only an *extremely lightweight*, "degraded" version of *centering*, where we formulate the rules on an *ad hoc* basis as needed by the annotation task. The underlying ideas of the degrading are that S-CREAM is intended to work in restricted, albeit adaptable, domains. It is not even necessary to have a complete model, because we analyze only a very small part of the text. For instance, we analyze only the part about hotels with rooms, prices, addresses and hotel facilities.

We specify the discourse model by logical rules, the effects of which we illustrate in the following paragraphs. Here, we use the same inferencing mechanisms that we have already exploited for supporting annotation (cf., Section 4.4.1).

As our baseline model, we assume the "single template strategy", viz. only one type of tag, e.g. <hotel>, is determined to really introduce a new discourse referent. Every other pair of tag name and tag value is attached to this entity as an attribute filled by the tag value. For example "Zwei Linden" is recognized as an instance of Hotel, every other entity (like "Dobbertin", etc.) is attached to this instance resulting in a very shallow discourse representation by logical facts illustrated in Table 5.2(a).[3] This is probably the most shallow discourse representation possible, because it does not include ordering constraints or other

[3]Results have been selected to be comparable with Table 5.1.

dr:Zwei_Linden a :Hotel.	:Zwei_Linden a :Hotel.
dr:Zwei_Linden dr:name "Zwei Linden".	:Zwei_Linden name "Zwei Linden".
dr:Zwei_Linden dr:city "Dobbertin".	:Zwei Linden located_at :Dobbertin.
	:Dobbertin a :City.
	:Dobbertin name "Dobbertin".
dr:Zwei_Linden dr:single_room "single room".	:Zwei_Linden :has_room :Single_room1.
	:Single_room1 a :Single_Room.
dr:Zwei_Linden dr:price "25,66".	
dr:Zwei_Linden dr:currency "EUR".	
dr:Zwei_Linden dr:double_room "double room".	:Zwei_Linden :has_room :Double_room1.
	:Double_room1 a :Double_Room.
dr:Zwei_Linden dr:price "43,46".	
dr:Zwei_Linden dr:price "46,02".	
dr:Zwei_Linden dr:currency "EUR".	

(a) Discourse Representation (b) Target Graph Structure

Table 5.2.: Template Strategy

soft constraints. However, it is already adequate to map some of the relations in the discourse namespace ("dr:") to relations in the target space, thus resulting in Table 5.2(b). However, given this restricted tag set, not every relation can be detected.

For more complex models, we may also include ordering information (e.g. simply by augmenting the discourse representation tuples given in Table 5.2 by numbers; this may be modeled by using lists in N3 (see Section 2.4.4) and a set of rules that maps the discourse representation into the target structure integrating

- rules to only attach instances where they are allowed to become attached (e.g., see first rule head of Listing 5.2, where prices are only attached where they are allowed).
- rules to attach tag values to the nearest preceding, conceptually possible entity. The nearest preceding entity is determined by exploiting the ordering information, e.g. as given in Table 5.2 (thus, prices for single and double room may be distinguished).
- rules to create a new complex object when two simple ones are adjacent, e.g., to create a rate when it finds adjacent number and currencies (e.g., see second rule head of Listing 5.2).

Further information that could be included is, e.g. adjacency information, etc. Thus, one may produce Table 5.1(a) out of the discourse representation from a numbered Table 5.2.

Listing 5.1 is a simplified example of a N3 rule that creates instances given the information in Table 5.3. The rules says: if the predicate p is in the list of concepts (e.g. city, single_room or double_room) then instantiate the corresponding instance y. As a side effect it generates a proper instance name *instance* and a proper concept name *concept* using the *cream:instanceName* function and the *cream:conceptName* function, respectively.

Listing 5.2 combines the complex object creation with the attachment to the nearest preceding conceptually possible entity. The rule says, if *price* is preceding *currency*, whereas the minimal distance is ensured by the function *cream:minBefore*, than create an instance with an unique instance name (*cream:uniqueInstanceName*) of type Rate and fill its attributes with the price and the currency. Further, the rule says, if a single room entity *room* is preceeding the currency in the discourse structure than attach the corresponding rate object to the instance ui_room of that room.

The strategy that we follow here is to keep simple things simple and complex tasks possible. The experienced user will be able to handcraft logical rules in order to refine the discourse model to his needs. The standard user, will only exploit the simple template strategy. When the resulting graph structures are simple enough to allow for the latter strategy and a simple mapping, the mapping can also be defined by directly aligning relevant concepts and relations by drag and drop, whilst, in the general, one must write logical rules.

dr:Zwei_Linden a :Hotel.
dr:Zwei_Linden dr:city (2 "Dobbertin").
dr:Zwei_Linden dr:single_room (3 "single room").
dr:Zwei_Linden dr:price (4 "25,66").
dr:Zwei_Linden dr:currency (5 "EUR").
dr:Zwei_Linden dr:double_room (6 "double room").
dr:Zwei_Linden dr:price (7 "43,46").
dr:Zwei_Linden dr:price (8 "46,02").
dr:Zwei_Linden dr:currency (9 "EUR").

Table 5.3.: Ordering Strategy

5.1.5. Conclusion

S-CREAM bridges the gap between the output of a information extraction system, viz. Amilcare, and the annotation one would yield when doing manual

```
{?hotel ?p  (?n ?y).
 ?p list:in (dr:city dr:single_room dr:double_room).
 ?y cream:instanceName ?instance.
 ?p cream:conceptName ?concept
} =>
{?instance a :concept}.
```

Listing 5.1: Example: rule to create instances

```
{
   ?hotel dr:single_room (?n0, ?room).
   ?hotel dr:price (?n1 ?price).
   ?hotel dr:currency (?n2 ?currency).
   ?n1 cream:minBefore ?n2.
   "Rate" cream:uniqueInstanceName ?ui_rate.
   ?n0 cream:minBefore ?n2.
   ?room cream:uniqueInstanceName ?ui_room.
} =>
{
# attach rate instance, at room where it is allowed to be attached
?ur_room :rate ?ui_rate}.
# create complex object rate
?ui_rate a :Rate.
?ui_rate :price ?price. ?ui_rate :currency ?currency.
```

Listing 5.2: Example: create rate and attach it to room

ontology based annotation on the same document. The bridging is done by applying a simple discourse representation, which is specified by logical rules. Using the concept of discourse representation is a first novel, useful and necessary step to generate relational markup for a large set of documents.

While this is a first approach, there are also some unsettled points. Firstly, the process to generate the discourse representation is not automatic but relies on hand crafted rules. In the ideal case, the information extraction system would be ontology aware and therefore learn the discourse structure of the given documents as well. To extend the information extraction system in this way was not possible for us because we could use Amilcare only as a black box component.We envisage therefore further research to specify and implement such an ontology aware information extraction component. Second, to use information extraction one needs a rather large amount of previously annotated documents, which is only useful if one has to annotate a group of similar documents.

To our knowledge nobody else has applied discourse representation for semi-automatic annotation and therefore offered a solution for the bridging problem to date. So, even with the above mentioned manual work we present the most advanced solution for that problem.

We demonstrate the usefulness and efficiency of the approach in an evaluation. In this evaluation we compared the annotation task of S-CREAM in comparison with CREAM, which is described in detail Section 9.

In conclusion S-CREAM is a unique framework that deals with all the conceptual difficulties of a full-fledged information extraction to support semi-automatic annotation.

5.2. The Self-Annotating Web

In the section before we presented an approach that extract with help of an information extraction component knowledge structures from Web pages through the use of information extraction rules. These rules are the result of a learning-cycle based on already annotated pages. While this approach is useful and efficient (as shown in the evaluation in Section 9, it also has some drawbacks:

- Manual definition of an information extraction system is a laborious task requiring a lot of time and expert know-how ([Appelt *et al.*, 1993]); and
- Learning of extraction rules requires a large number of examples for learning the rules.

To overcome this drawbacks, we propose in this Section a different paradigm: the *Self-annotating Web*. The principle idea of the self-annotating Web is that it uses globally available Web data and structures to semantically annotate — or at least facilitate annotation of — local resources. Initial blueprints for this paradigm are found in such works as the following:

- Some researchers use explicit, linguistically motivated natural-language descriptions to propose semantic relationships ([Charniak and Berland, 1999; Googlism, 2003; Hearst, 1992; Maedche and Staab, 2001]).
- Others use the Web to cope with data sparseness problems in tasks that require statistics about possible semantic relationships ([Agirre *et al.*, 2000; Grefenstette, 1999; Keller *et al.*, 2002; Markert *et al.*, 2003]).
- In [Flake *et al.*, 2002; Glover *et al.*, 2002], the Web structure itself is used to determine a focus for harvesting data. Thus, specialized semantic relationships, such as recommendations coming from a particular Web community can be derived.

In taking a first step towards the Semantic Web, we propose an original method called *PANKOW* (Pattern-based Annotation through Knowledge On the Web),

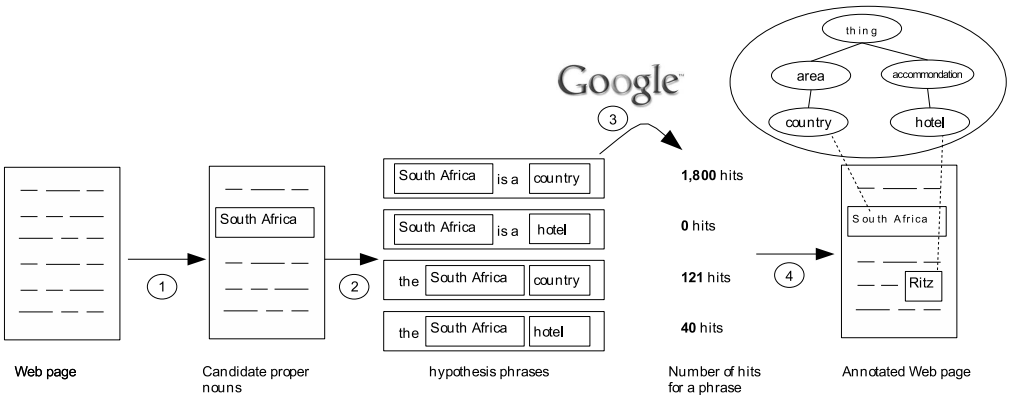

Figure 5.4.: The Process of PANKOW

which employs an unsupervised, pattern-based approach to categorize instances with regard to a given ontology.

The approach is novel, combining the idea of using linguistic patterns to identify certain ontological relations as well as the idea of using the Web as a big corpus to overcome data sparseness problems. It is unsupervised as it does not rely on any training data annotated by hand, and it is pattern-based in the sense that it makes use of linguistically motivated regular expressions to identify instance-concept relations in text. The driving principle behind PANKOW is one of "disambiguation by maximal evidence" in the sense that for a given instance it proposes the concept with the maximal evidence derived from Web statistics. The approach thus bootstraps Semantic Annotations as it queries the Web for relevant explicit natural-language descriptions of appropriate ontological relations.

PANKOW has been conceived for our annotation framework CREAM [Handschuh and Staab, 2002] and has been implemented in OntoMat[4] using queries to the Google Web service API. The automatic annotation produced by PANKOW has been evaluated against Semantic Annotations produced by two independent human subjects.

5.2.1. The Process of PANKOW

This section gives a general overview of the process of PANKOW whereas section 5.2.2 explains the concrete methods and section 5.2.3 the implementation details. The process consists of four steps (depicted in Figure 5.4):

[4] annotation.semanticweb.org/tools/ontomat

Input: A Web page.

In our implementation, we assume that Web pages are handled individually in the CREAM/OntoMat framework ([Handschuh *et al.*, 2001]), though actually batch processing of a whole Web site would be possible.

Step 1: The system scans the Web page for phrases in HTML text that might be categorized as instances of the ontology. Candidate phrases are proper nouns, such as '*Nelson Mandela*', '*South Africa*', or '*Victoria Falls*'). We use a part-of-speech tagger Section 5.2.3) to find such candidate proper nouns.

Thus, we end up with a

Result 1: set of candidate proper nouns

Step 2: The system iterates through the candidate proper nouns. It uses the approach described in Section 5.2.2, introducing all candidate proper nouns and all candidate ontology concepts into linguistic patterns to derive hypothesis phrases. For instance, the candidate proper noun '*South Africa*' and the concepts Country and Hotel are composed into a pattern resulting in hypothesis phrases like '*South Africa is a country*' and '*South Africa is a hotel*'.

Result 2: Set of hypothesis phrases.

Step 3: Then, Google is queried for the hypothesis phrases through its Web service API (Section 5.2.2). The API delivers as its results

Result 3: the number of hits for each hypothesis phrase.

Step 4: The system sums up the query results to a total for each instance-concept pair. Then the system categorizes the candidate proper nouns into their highest ranked concepts (cf. Section 5.2.2). Hence, it annotates a piece of text as describing an instance of that concept. Thus we have

Result 4: an ontologically annotated Web page.

In principle, the query results of step 3 could be investigated further. For instance, it could make sense to constrain the number of hits for hypothesis phrases to the ones that occur in Web pages with topics closely related to the topic of the current Web page, as e.g. measured in terms of cosine distance of the documents. However, without direct access to the Google databases we have considered this step too inefficient for use in automatic annotation and hence ignore it in the following.

5.2.2. Pattern-based Categorization of Candidate Proper Nouns

There is some history of applying linguistic patterns to identify ontological relationships between entities referred to in a text. For instance, Hearst [Hearst, 1992] as well as Charniak and Berland [Charniak and Berland, 1999] make use of such a pattern-based approach to discover taxonomic and part-of relations from text, respectively. Hahn and Schnattinger [Hahn and Schnattinger, 1998] also make use of such patterns and incrementally established background knowledge to predict the correct ontological class for unknown named entities appearing in a text. The core idea of any such pattern-based approach is that one may justify an ontological relationship with reasonable accuracy when one recognizes some specific idiomatic/syntactic/semantic relationships. It is relevant to the pattern-based approach that the specifically addressed idiomatic/syntactic/semantic relationships may be very easily spotted because they can be typically specified through simple and efficiently processable regular expressions.

In the following, we first present the set of patterns; in a second step we describe the procedure to actually search for them; and finally we explain how we use the information conveyed by them for the actual classification of instances.

Patterns for Generating Hypothesis Phrases

In the following we describe the patterns we exploit and give a corresponding example from the data set.

Hearst Patterns

The first four patterns have been used by Hearst to identify *isa*-relationships between the concepts referred to by two terms in the text. However, they can also be used to categorize a candidate proper noun into an ontology.

Since the entities denoted by candidate proper nouns are typically modeled as instances of an ontology, we also describe the problem more conveniently as the instantiation of a concept from a given ontology. Correspondingly, we formulate our patterns using the variable '<INSTANCE>' to refer to a candidate noun phrase, as the name of an ontology instance, and the variable '<CONCEPT>' to refer to the name of a concept from the given ontology.

The patterns reused from Hearst are:

HEARST1: <CONCEPT>s such as <INSTANCE>

HEARST2: such <CONCEPT>s as <INSTANCE>

HEARST3: <CONCEPT>s, (especially|including) <INSTANCE>

HEARST4: <INSTANCE> (and|or) other <CONCEPT>s

The above patterns would match the following expressions (in this order): *hotels such as Ritz*; *such hotels as Hilton*; *presidents, especially George Washington*; and *the Eiffel Tower and other sights in Paris*.

Definites

The next patterns are about definites, i.e. noun phrases introduced by the definite determiner '*the*'. Frequently, definites actually *refer* to some entity previously mentioned in the text. In this sense, a phrase like '*the hotel*' does not stand for itself, but it points as a so-called anaphora to a unique hotel occurring in the preceding text. Nevertheless, it has also been shown that in common texts more than 50% of all definite expressions are *non-referring*, i.e. they exhibit sufficient descriptive content to enable the reader to uniquely determine the entity referred to from the global context ([Poesio and Vieira, 1998]). For example, the definite description '*the Hilton hotel*' has sufficient descriptive power to uniquely pick-out the corresponding real-world entity for most readers. One may deduce that '*Hilton*' is the name of the real-world entity of type Hotel to which the above expression refers.

Consequently, we apply the following two patterns to categorize candidate proper nouns by definite expressions:

DEFINITE1: the <INSTANCE> <CONCEPT>

DEFINITE2: the <CONCEPT> <INSTANCE>

The first and the second pattern would, e.g., match the expressions '*the Hilton hotel*' and '*the hotel Hilton*', respectively.

Apposition and Copula

The following pattern makes use of the fact that certain entities appearing in a text are further described in terms of an apposition as in '*Excelsior, a hotel in the center of Nancy*'. The pattern capturing this intuition looks as follows:

APPOSITION: <INSTANCE>, a <CONCEPT>

Probably the most explicit way of expressing that a certain entity is an instance of a certain concept is by the verb '*to be*', as for example in '*The Excelsior is a hotel in the center of Nancy*'. Here's the general pattern:

COPULA: <INSTANCE> is a <CONCEPT>

Finding Patterns

Having defined these patterns, one could now try to recognize these patterns in a corpus and propose the corresponding instance-concept relationships. However, it is well known that the above patterns are rare and thus one will need a sufficiently big corpus to find a significant number of matches.

Thus, PANKOW resorts to the biggest corpus available: the World Wide Web. In fact, several researchers have shown that using the Web as a corpus is an effective way of addressing the typical data sparseness problem one encounters when working with corpora (compare [Grefenstette, 1999], [Keller *et al.*, 2002], [Markert *et al.*, 2003], [Resnik and Smith, 2003]). Actually, we subscribe to the principal idea by Markert *et al.* [Markert *et al.*, 2003] of exploiting the Google API. As in their approach, rather than actually downloading Web pages for further processing, we just take the number of Web Pages in which a certain pattern appears as an indicator for the strength of the pattern.

Given a candidate proper noun that we want to tag or annotate with the appropriate concept, we instantiate the above patterns with each concept from the given ontology into hypothesis phrases. For each hypothesis phrase, we query the Google API for the number of documents that contain it. The function 'count(i,c,p)' models this query.

$$\text{count} : I \times C \times P \rightarrow \mathbb{N} \tag{5.1}$$

Thereby, i, c, p are typed variables and shorthand for <INSTANCE>, <CONCEPT> and a pattern. Correspondingly, I, C and P stand for the set of all candidate proper nouns, all concepts from a given ontology and all patterns, respectively.

Categorizing Candidate Noun Phrases

We have explored three versions for determining the best categorization.

1. Baseline: The simplest version just adds all the numbers of documents with hits for all hypothesis phrases resulting from one <INSTANCE>/ <CONCEPT> pair.

$$\text{count}_b(i, c) := \sum_{p \in P} \text{count}(i, c, p) \tag{5.2}$$

   This baseline count_b proved to be effective and empirical results presented subsequently will report on this method.

2. Interactive Selection: In an annotation scenario, it is not always necessary to uniquely categorize a candidate proper noun. Rather it is very easy and effective to present to the manual annotator the top ranked <INSTANCE>/<CONCEPT> pairs and let him decide according to the actual context. This is currently implemented in CREAM/ OntoMat based on the validity indicated by count_b (also cf. Section 5.2.3).

In the first two approaches, one may return the set of pairs R_x ($x \in \{b, \vec{w}\}$) where for a given $i \in I$, $c \in C$ maximizes the strength as aggregated from the individual patterns as the result of PANKOW:

$$R_x := \{(i, c_i) | i \in I, c_i := \text{argmax}_{c \in C} \text{count}_x(i, c)\} \tag{5.3}$$

and in the last approach, we return the best n matches for each proper noun resulting in R_x^n ($x \in \{b, \vec{w}\}, n \in \mathbb{N}$)[5]:

$$\begin{aligned} R_x^n := \{(i, c_i) | i \in I : c_{i,j} \in C \wedge \{c_{i,1} \ldots c_{i,|C|}\} = C \\ \text{count}_x(i, c_{i,1}) \geq \text{count}_x(i, c_{i,2}) \geq \ldots \geq \text{count}_x(i, c_{i,|C|}) \wedge \\ c_i = c_{i,1} \vee \ldots \vee c_i = c_{i,n}\} \end{aligned} \tag{5.4}$$

For our evaluation we will use a characterization that does not accept classification of every candidate proper noun, such as R_x does, but only of those that appear strong enough. Thus, we introduce a threshold θ as follows:

$$\begin{aligned} R_{x,\theta} := \{(i, c_i) | i \in I, c_i := \text{argmax}_{c \in C} \text{count}_x(i, c) \wedge \\ \text{count}_x(i, c_i) \geq \theta\} \end{aligned} \tag{5.5}$$

5.2.3. Integration into CREAM

We have integrated PANKOW into the CREAM framework (cf., Chapter 4) extending the CREAM implementation OntoMat by a plugin. The plugin has access to the ontology structure and to the document management system of OntoMat. The plugin utilizes the Google API to access its Web service.

The PANKOW plugin implements the process described in Section 5.2.1 starting with the scanning for the candidate proper nouns by using a POS tagger. We experimented with two POS taggers: One was QTag[6] and the other was Tree-Tagger ([Schmid, 1994]). The advantage of QTag is that it is implemented in Java and therefore better to integrate, whereas, Tree-Tagger produces somewhat better results in our experiments.

[5] Obviously, $R_x^1 = R_x$.

[6] http://web.bham.ac.uk/o.mason/software/tagger/index.html

In addition, we use a heuristic to achieve higher precision for the candidate recognition and therefore to reduce the amount of queries. The heuristic considers the intersection of the POS tagger categorization with the simple capitalized-words approach which consists in interpreting sequences of capitalized words as proper noun candidates[7]. For the capitalized words approach we consider only words that do not follow a period. Given the example of the lonely planet Web page about Nigeria[8], the POS tagger proposes proper nouns such as "Guinea", "Niger", "Cameroon", "Benin", and "Nigeria". For this concrete example, the capitalized words approach proposes basically the same proper nouns as the POS tagger. However, in general the capitalized word heuristic will reduce tagging errors produced by the POS tagger. While our heuristic approach is practical, it has some problems with compound words such as "Côte d'Ivoire" and might need some fine-tuning.

OntoMat supports two modes of interaction with PANKOW: *(i)*, fully automatic annotation and, *(ii)*, interactive semi-automatic annotation. In the fully automatic mode, all categorizations with strength above a user-defined θ, viz. $R_{b,\theta}$, are used to annotate the Web content. In the interactive mode, the system proposes the top five concepts to the user for each instance candidate, i.e. R_b^5. Then, the user can disambiguate and resolve ambiguities.

The screenshot in Figure 5.5 shows the user interface. In the lower left corner of the screenshot you can see the progress dialog for the Google queries. The dialog shows the extracted candidate proper nouns and logs the query results for the hypothesis phrases. Also shown is the interactive dialog for disambiguation, e.g. the choice to assign "Niger" as an instance to one of the concepts "river", "state", "coast", "country" or "region". The number in the brackets behind each concept name gives the number of Web hits.

5.2.4. Conclusion

We have described PANKOW, a novel approach towards the *Self-annotating Web*. It overcomes the burden of laborious manual annotation and it does not require the manual definition of an information extraction system or its training based on manually provided examples. It uses the implicit wisdom contained in the Web to propose annotations derived from counting Google hits of instantiated linguistic patterns. The results produced are comparable to state-of-the-art systems, whereas our approach is comparatively simple, effortless and intuitive to use to annotate the Web.

While we consider PANKOW as a valid step towards the self-annotating Web,

[7] This heuristic works especially well for English, where typically only proper nouns appear capitalized.

[8] http://www.lonelyplanet.com/destinations/africa/nigeria/environment.htm

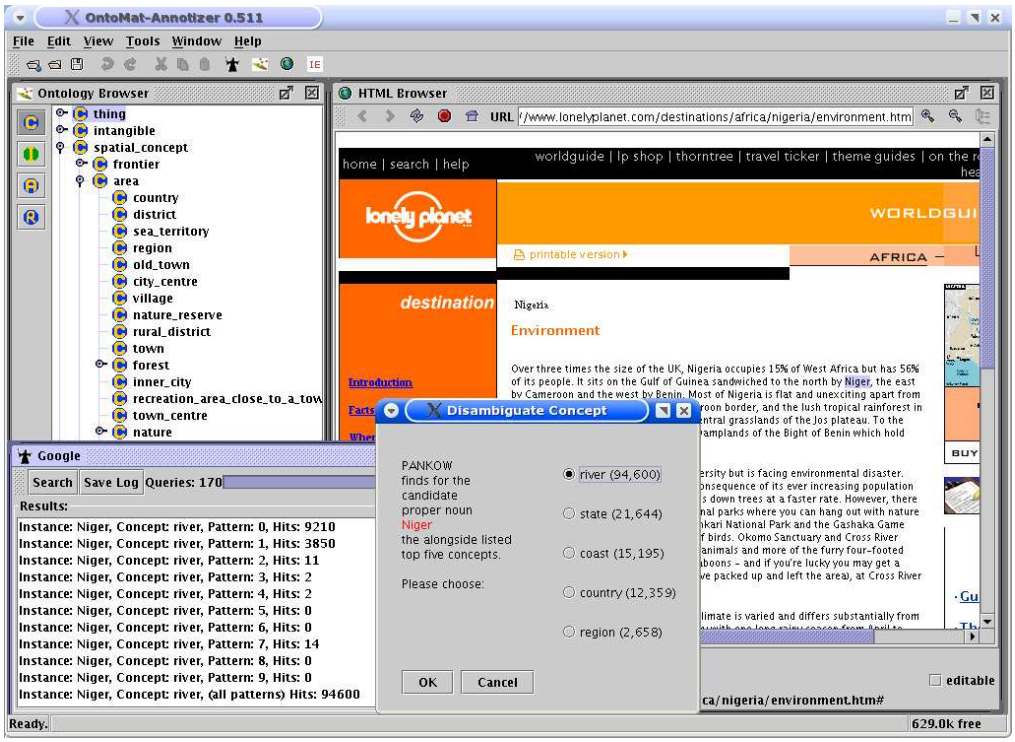

Figure 5.5.: Screenshot of CREAM with PANKOW plugin in interactive mode

we are well-aware that effectiveness, efficiency, and range of PANKOW needs to and can be improved.

With regard to effectiveness, we have mentioned initially that [Flake *et al.*, 2002; Glover *et al.*, 2002] used the Web structure itself to determine a focus for harvesting data. In this line, by determining such a focus we could have a more domain-specific disambiguation than in our current approach. Such a focus could for example be determined by crawling only similar documents from the Web as for example in the approach of Agirre et al. ([Agirre *et al.*, 2000]). For instance, our annotators tagged 'Niger' as a country, while PANKOW found the other meaning that 'Niger' has, viz. it also refers to the river 'Niger'.

With regard to efficiency, we thank Google for their help and support with their Web service API. However, the self-annotating Web is not possible with the kind of implementation that we provided. The reason is that we have issued an extremely large number of queries against the Google API — to an extent that would not scale towards thousands or even millions of Web pages in a reasonable

time-scale. However, we envision an optimized indexing scheme (e.g., normalizing the various forms of '*to be*' in order to recognize '*George Washington was a man*') and API that would reduce this number to acceptable load levels. Also, an interesting direction for further research would be to learn the weights of the different patterns by machine-learning techniques. Furthermore, in order to reduce the amount of queries sent to the Google Web service API, a more intelligent strategy should be devised, which takes into account the ontological hierarchy between concepts.

With regard to range, we have only covered the relationship between instances and their concepts, but not other relationships between instances, such as *is_located_in(Eiffel Tower,Paris)*. Our first step in this direction will be the tighter integration of PANKOW with Amilcare [Ciravegna, 2001a], such that instance data from PANKOW will be used to train Amilcare as has been done for Armadillo [Ciravegna *et al.*, 2003]. Overall, however, this remains extremely challenging work for the future.

6. Deep Annotation

This Chapter portrays an extension of the CREAM framework for metadata creation when Web pages are generated from a database *and* the database owner is cooperatively participating in the Semantic Web. In order to create metadata, the framework combines the presentation layer with the data description layer — in contrast to "conventional" annotation, which remains at the presentation layer. Therefore, we refer to the framework as *deep annotation*. Firstly, we elaborate on the use cases of deep annotation in order to illustrate its possible scope (Section 6.2). We continue with a description of the overall process in Section 6.3. Section 6.4 details the architecture that supports the process, where we find three major requirements that must be provided:

1. A server-side Web page markup that defines the relationship between the database and the Web page content (cf. Section 6.5)
2. An annotation tool to actually let the user utilize information proper, information structures and information context for creating mappings (cf. Section 6.6).
3. Components that let the user investigate the constructed mappings (cf. Section 6.7), and query the serving database.

References: This chapter is mainly based on [Handschuh *et al.*, 2003].

6.1. Introduction

One of the core challenges of the Semantic Web is the creation of metadata by mass collaboration, i.e. by combining semantic content created by a large number of people. To attain this objective we presented in Chapter 4 a basic framework that deals with the manual and/or the semi-automatic creation of metadata from existing information.

These approaches, however, as well as older ones that provide metadata, e.g. for search on digital libraries, build on the assumption that the information sources under consideration are *static*, e.g. given as static HTML pages or given as books in a library, etc. (cf., [A,B] in Table 6.1).

Nowadays, however, a large percentage of Web pages are not static documents. On the contrary, the majority of Web pages are dynamic.[1] For dynamic Web pages (e.g. ones that are generated from the database that contains a catalogue of books) it does not seem to be useful to manually annotate every single page. Rather one wants to "annotate the database" in order to reuse it for one's own Semantic Web purposes.

For this objective, approaches have been conceived that allow for the construction of wrappers by explicit definition of HTML or XML queries [Sahuguet and Azavant, 2001] or by learning such definitions from examples [Kushmerick, 2000; Ciravegna, 2001a].

Thus, it has been possible to manually create metadata for a set of structurally similar Web pages. The wrapper approaches come with the advantage that they do not require cooperation by the owner of the database. However, their shortcoming is that the correct scraping of metadata is dependent to a large extent on data layout rather than on the structures underlying the data (cf., [C] in Table 6.1).

While for many Web sites, the assumption of non-cooperativity may remain valid, we assume that many Web sites will in fact participate in the Semantic Web and will support the sharing of information. Such Web sites may present their information as HTML pages for viewing by the user, but they may also be willing to describe the structure of their information on the very same web pages. Thus, they give their users the possibility to utilize

1. information proper,
2. information structures, and
3. information context.

A user may then exploit these three in order to create mappings into his own information structures (e.g., his ontology) — which may be a lot easier than if the information a user receives is restricted to information structures [Noy and Musen, 2000] and/or information proper alone [Doan *et al.*, 2002].

We define "deep annotation" [D] as an annotation process that utilizes information proper, information structures and information context in order to derive mappings between information structures. The mappings may then be exploited by the same or another user to query the database underlying a Web site in

[1] It is not possible to give a percentage of dynamic to static Web pages in general, because a single Web site may use a simple algorithm to produce an infinite number of, probably not very interesting, Web pages. Estimations, however, based on Web pages actually crawled by existing search engines estimate that dynamic Web pages outnumber static ones by 100 to 1.

order to retrieve semantic data — combining the capabilities of conventional annotation and databases.

Table 6.1 summarizes the different settings just laid out.

Web site	cooperative owner	uncooperative owner
static	Embedded or remote metadata by conventional annotation [A]	Remote metadata by conventional annotation [B]
dynamic	*Deep annotation* with server- or client-side mapping rules [D]	Wrapper construction, remote metadata [C]

Table 6.1.: Principal situation

Thereby, Table 6.1 further distinguishes between two scenarios regarding static Web sites. In the one scenario [B], the annotator is not allowed to change static information, but he can create the metadata and remotely retain it from the source it belongs to (e.g., by XPointer). In the other scenario [A], he is free to choose between embedding the metadata created in the annotation process into the information proper (e.g., via the <meta> tag of a HTML page) or keeping it remote.[2] For deep annotation the two choices boil down to storing the created mappings either within the database of the server or externally.

6.2. Use Cases for Deep Annotation

Deep annotation is relevant for a large and fast-growing number of web sites which have cooperation as their aim, for instance:

Scientific databases. They are frequently built to foster cooperation by researchers. Medline, Swissprot, or EMBL are just a few examples that can be found among the Web. In the bioinformatics community alone it is currently estimated that 500+ large databases are freely accessible.

Such databases are frequently hard to understand and it is often difficult to evaluate whether a database table named "species" is equivalent to a table named "organism" in another database. Exploiting the information proper found in concrete tuples may help. But whether the "leech" considered as entry to an "organism" is actually the animal or the plant may be much easier to tell from the context in which it is presented than from the concrete database entry, which may resolve to "plant" or "animal" only via several joins.[3]

Supply Chains. Car manufacturers frequently "outsource" the problem of providing mappings to their databases for tenders and orders by providing a portal

[2] Cf. Chapter 4 on those two possibilities.

[3] Concrete examples are typically not as easy to understand as the leech example!

with only HTML pages to their suppliers. Suppliers then either need to retype information or to use wrappers in order to replicate information. When the manufacturers provided information structures on their portal, a one-time deep annotation process could easily create the mapping for new suppliers.

Syndication. Besides direct access to HTML pages of news stories or market research reports, etc., commercial information providers frequently provide syndication services. The integration of such syndication services into the portal of a customer is typically an expensive manual programming effort that could be reduced by a deep annotation process which defines the content mappings.

For the remainder of the paper we will focus on the following use case:

Community Web Portal (cf., [Staab *et al.*, 2000a]). It serves the information needs of a community on the Web with possibilities for contributing and accessing information by community members. A recent example that is also based on Semantic Web technology is[4] [Studer *et al.*, 2002]. The interesting aspect of such portals lies in the sharing of information and some of them are even designed to deliver semantic information back to their community as well as to the outside world.[5]

The primary objective of a community setting up a portal will remain to offer access for users. However, given the appropriate tools they could easily produce information content, information structures and information context to their members for deep annotation. The way in which this process operates is described in the following.

6.3. The Process of Deep Annotation

The process of deep annotation consists of the following four steps (depicted in Figure 6.1):

Input: A Web site[6] driven by an underlying relational database.

Step 1: The database owner produces server-side Web page markup according to the information structures of the database (described in detail in Section 6.5).

Result: Web site with server-side markup.

Step 2: The annotator produces client-side annotations conforming to the client ontology and the server-side markup (Section 6.6).

[4] http://www.ontoweb.org
[5] Cf., e.g., [Stojanovic *et al.*, 2001] for an example producing RDF from database content.
[6] Cf. Section 6.8 on other information sources.

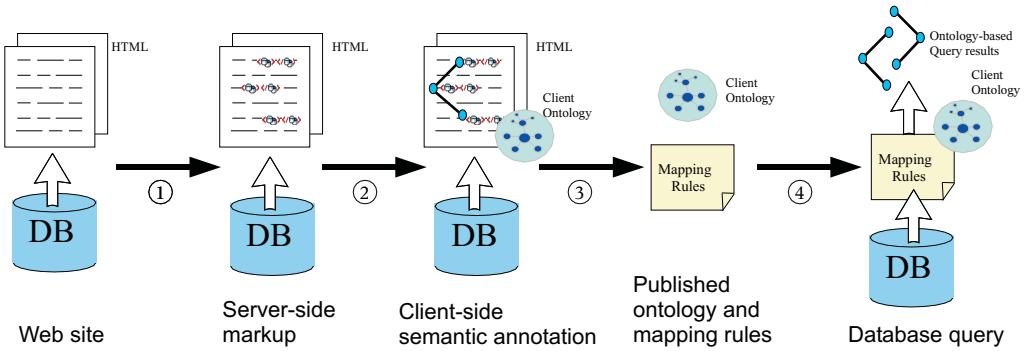

Figure 6.1.: The Process of Deep Annotation

Result: Mapping rules between database schema and client ontology

Step 3: The annotator publishes the client ontology (if not previously completed) and the mapping rules derived from annotations (Section 6.7).

Result: The annotator's ontology and mapping rules are available on the Web

Step 4: The querying party loads the second party's ontology and mapping rules and uses them to query the database via the Web service API (Section 6.7.1 and 6.7.2).

Result: Results retrieved from database by querying party

Obviously, in this process one single person may be the database owner and/or the annotator and/or the querying party.

To align this with our running example of the community Web portal, the annotator might annotate an organization entry from ontoweb.org according to his own ontology. Subsequently, he may use the ontology and mapping to instantiate his own syndication services by regularly querying for all recent entries the titles of which match his list of topics.

6.4. Architecture

Our architecture for deep annotation consists of three major pillars corresponding to the three different roles (database owner, annotator, querying party) as described in the process.

Database and Web Site Provider. At the Web site, we assume that there is an underlying database (cf. Figure 6.2) and a server-side scripting environment, like Zope, JSP or ASP, used to create dynamic Web pages. Furthermore, the Web site may also provide a Web service API to third parties who want to query the database directly.

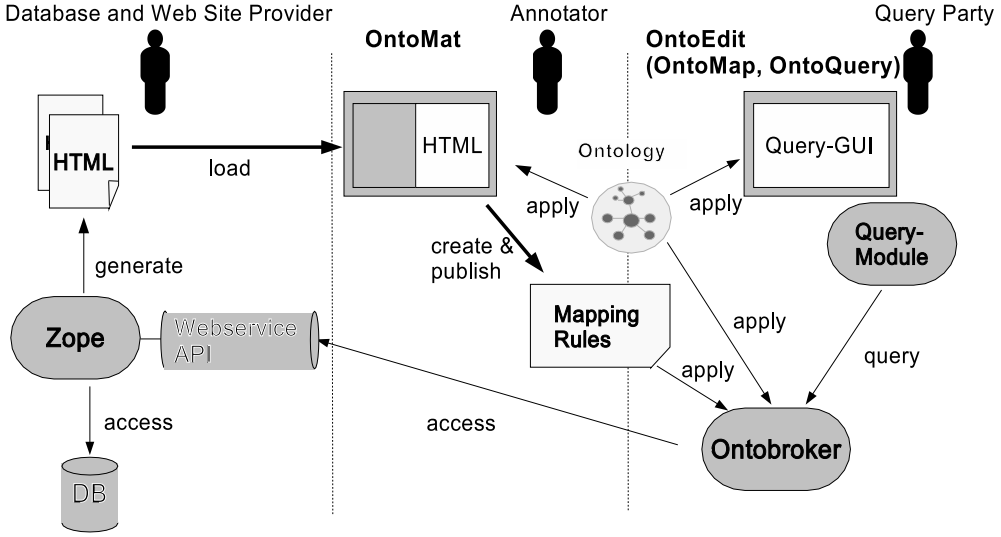

Figure 6.2.: An Architecture for Deep Annotation

Annotator. The annotator uses an extended version of the OntoMat-Annotizer in order to manually create relational metadata, which correspond to a given client ontology, for some Web pages. The extended OntoMat-Annotizer takes into account problems that may arise from generic annotations required by deep annotation (see Section 6.6). With the help of OntoMat-Annotizer, we create mapping rules from such annotations that are later exploited by an inference engine.

Querying Party. The querying party uses a corresponding tool to visualize the client ontology, to compile a query from the client ontology and to investigate the mapping. In our case, we use OntoEdit [Sure *et al.*, 2002a] for those three purposes. In particular, OntoEdit also allows for the investigation, debugging and change of given mapping rules. To that extent, OntoEdit integrates and exploits the Ontobroker [Fensel *et al.*, 1999] inference engine (see Figure 6.2).

6.5. Server-Side Web Page Markup

The goal of the mapping process is to allow interested parties to gain access to the source data. Hence, the content of the underlying database is not materialized, as proposed in [Stojanovic *et al.*, 2002c]. Instead, we provide pointers to the underlying data sources in the annotations, e.g. we specify which database columns provide the data for certain attributes of instances. Thus, the capabilities of conventional annotation and databases are combined.

6.5.1. Requirements

All required information has to be published, so that an interested party can use this information to retrieve the data from the underlying database. This information has to specify *(i)* which database is used as a data source and how this database can be accessed *(ii)* which query is used to retrieve data from the database and *(iii)* which elements of the query result are used to create the dynamic web page. Those three components are detailed in the remainder of this section.

6.5.2. Database Representation

The database representation is specified using a dedicated deep annotation ontology, which is instantiated to describe the physical structure of the part of the database which may facilitate the understanding of the query results. Thereby, the structure of all tables/views involved in a query can be published. For example the following representation is part of the HTML head of the Web page presented in Figure 6.3.

```
<!--

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:da="http://annotation.semanticweb.org#deepanno">
  <da:DB rdf:ID="OntoSQL">
    <da:accessService
        rdf:resource="www.ontoweb.org/database_access.wsdl"/>
  </da:DB>
  <da:Table rdf:ID="Person">
    <da:name>Person</da:sqlName>
    <da:inDatabase rdf:resource="#OntoSQL" />
    <da:hasColumns rdf:parseType="Collection">
        <da:PrimaryKey rdf:ID="Person.ID"
            da:name="ID" da:type="int" />
        <da:Column da:name="FIRSTNAME" da:type="varchar"/>
        <da:Column da:name="LASTNAME" da:type="varchar"/>
    </da:hasColumns>
  </da:Table>
```

```
  <da:Table rdf:ID="Organization">
    <da:name>Organization</da:name>
    <da:inDatabase rdf:resource="#OntoSQL" />
    <da:hasColumns rdf:parseType="Collection" />
        <da:PrimaryKey rdf:ID="Organization.ID"
            da:name="ID" da:type="int" />
        <da:Column da:name="ORGNAME" da:type="varchar"/>
        <da:Column da:name="LOCATION" da:type="varchar"/>
        ...
    </da:hasColumns>
  </da:Table>
  <da:Table rdf:ID="PersonOrg">
    <da:name>Person_Org<da:name>
    <da:inDatabase rdf:resource="#OntoSQL" />
    <da:hasColumns rdf:parseType="Collection" />
        <da:PrimaryKey da:name="PERSONID" da:type="int">
            <references rdf:resource="#Person.ID"/>
        </da:PrimaryKey>
        <da:PrimaryKey da:name="ORGID" da:type="int">
            <references rdf:resource="#Organization.ID"/>
        </da:PrimaryKey>
    </da:hasColumns>
  </da:Table>
</rdf:RDF>
-->
```

The property *accessService* of the <DB> class represents the link to a service which allows anonymous database access; consequently additional security measures can be implemented in the service. Usually, anonymous users should only have read-access to public information. As we rely on a Web service to host the database access we avoid protocol issues (database connections are usually made via sockets on proprietary ports).

6.5.3. Query Representation

Additionally, the query itself, which is used to retrieve the data from a particular source is placed in the header of the page. It contains the intended SQL-query and is associated with a name as a means to distinguish between queries and operates on a particular data source.

```
<!--

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:da="http://annotation.semanticweb.org#deepanno">
    <da:Query rdf:ID="Q1">
      <da:source rdf:resource="#OntoSQL" />
      <da:hasResultColumns rdf:parseType="Collection">
        <ColumnGroup rdf:about="#g1" />
        <ColumnGroup rdf:about="#g2" />
      </da:hasResultColumns>
      <da:sql>
```

```
      SELECT Person.*, Person_Org.Orgid, Organization.*
      FROM   Person, Organization, Projekt_Org
      WHERE  Person.ID = Projekt_Org.PERSONID
             AND Organization.ID = Projekt_Org.ORGID
      </da:sql>
    </da:Query>
    <da:Columngroup rdf:ID="#g1">
        <da:prefix
            rdf:resource="http://www.ontoweb.org/person/">
        <da:hasColumns rdf:parseType="Collection">
            <Identifier da:name="Id" />
            <Column da:name="Firstname" />
            <Column da:name="Lastname" />
        </da:hasColumns>
    </da:Columngroup>
    <da:Columngroup rdf:ID="#g2">
        <da:prefix
            rdf:resource="http://www.ontoweb.org/org/">
        <da:hasColumns rdf:parseType="Collection">
            <Identifier da:name="OrganizationId" />
            <Column da:name="Orgname" />
            <Column da:name="Location" />
        </da:hasColumns>
    </da:Columngroup>
  </rdf:RDF>
-->
```

The structure of the query result must be published by means of column groups. Each column group must have at least one identifier, which is used in the annotation process to distinguish between individual instances and detect their equivalence. Since database keys are only local to the respective table, but the Semantic Web has a global space of identifiers, appropriate prefixes have to be established. The prefix also ensures that the equality of instance data generated from multiple queries can be detected, if the Web application maintainer chooses the same prefix for each occurrence of that id in a query. Eventually, database keys are translated to instance identifiers (cf. Section 6.7.2) via the following pattern:

$$< prefix > [key_i - name = key_i - value]$$

For example: http://www.ontoweb.org/person/id=1

6.5.4. Result Representation

Whenever parts of the query results are used in the dynamically generated Web page, the generated content is surrounded by a tag, which carries information about which column of the resulting tuple delivered by a query represents the used value. In order to maintain compatibility with HTML, we used the tag as an information carrier. The actual information is represented in attributes of :

```
<table>
```

```
<tr>
<td>
<span qresult="q1" column="Orgname">AIFB</span>
</td>
<td>
<span qresult="q1" column="Location">Karlsruhe</span>
</td>
...
<td>
<span qresult="q1" column="Firstname">Steffen</span>
</td>
...
</tr>
</table>
```

Such span tags are then interpreted by the annotation tool and are used in the mapping process.

6.6. Annotation

An annotation in our context is a set of instantiations related to an ontology and referring to an HTML document. We distinguish *(i)* instantiations of OWL classes, *(ii)* instantiated properties from one class instance to a datatype instance — henceforth called attribute instance (of the class instance), and *(iii)* instantiated properties from one class instance to another class instance — henceforth called relationship instance.

In addition, for deep annotation one must distinguish between a *generic annotation* and a *literal annotation*. In a *literal annotation*, the piece of text may stand for itself. In a *generic annotation*, a piece of text that corresponds to a database field and that is annotated is only considered to be a place holder i.e. a variable must be generated for such an annotation and the variable may have multiple relationships allowing for the description of general mapping rules. For example, a concept Institute in the client ontology may correspond to one generic annotation for the Organization identifier in the database.

Applying the above terminology, we will refer to generic annotation in detail as *generic class instances*, *generic attribute instances*, and *generic relationship instances*.

6.6.1. Annotation Process

An annotation process of server-side markup (generic annotation) is supported by the user interface as follows:

1. The user opens in the browser a server-side marked up Web page.

2. The server-side markup is handled individually by the browser, e.g. it provides graphical icons on the page wherever a markup is present, so that the user can easily identify values which come from a database.

3. The user can select one of the server-side markups to either create a new *generic instance* and map its database field to a generic attribute, or map a database field to a *generic attribute* of an existing *generic instance*.

4. The database information necessary to query the database in a later step is stored along with the *generic instance*.

The reader may note that *literal annotation* is still performed when the user drags a marked up piece of content, that is not a server-side markup.

6.6.2. Creating Generic Instances of Classes

When the user drags a server-side markup onto a particular concept of the ontology, a new generic class instance is generated (cf. arrow #1 in Figure 6.3). The application displays a dialog for the selection of the instance name and the attributes to which the database value may be mapped. Attributes which resemble the column name are preselected (cf. dialog #1a in Figure 6.3). If the user clicks OK, database concept and instance checks are performed and the new generic instance is created. Generic instances will appear with a database symbol in their icon.

Each generic instance stores the information about the database query and the unique identifier pattern. This information is resolved from the markup. A server-side markup contains the reference to the query, the column, and the value. The identifier pattern is obtained from the reference to the query description and the appropriate column group (cf. Section 6.5.3). The markup used to create the instance defines the identifier pattern for the generic instance. The identifier pattern will be used when instances are generated from the database (cf. Section 6.7.2). For example, the server-side markup "AIFB" is selected and dropped on the concept Institute. The content of the markup is 'AIFB'. This creates a new generic instance with a reference to the query *q1* (cf. Section 6.5.3). The dialog-based choice for the instance name "AIFB" assigns the generic attribute name with the database column "Orgname". This defines the identifier pattern of the generic instance as "http://www.ontoweb.org/org/OrganizationID=$OrganizationID". OrganizationID is the name of the database column in query *q1* that holds the database key.

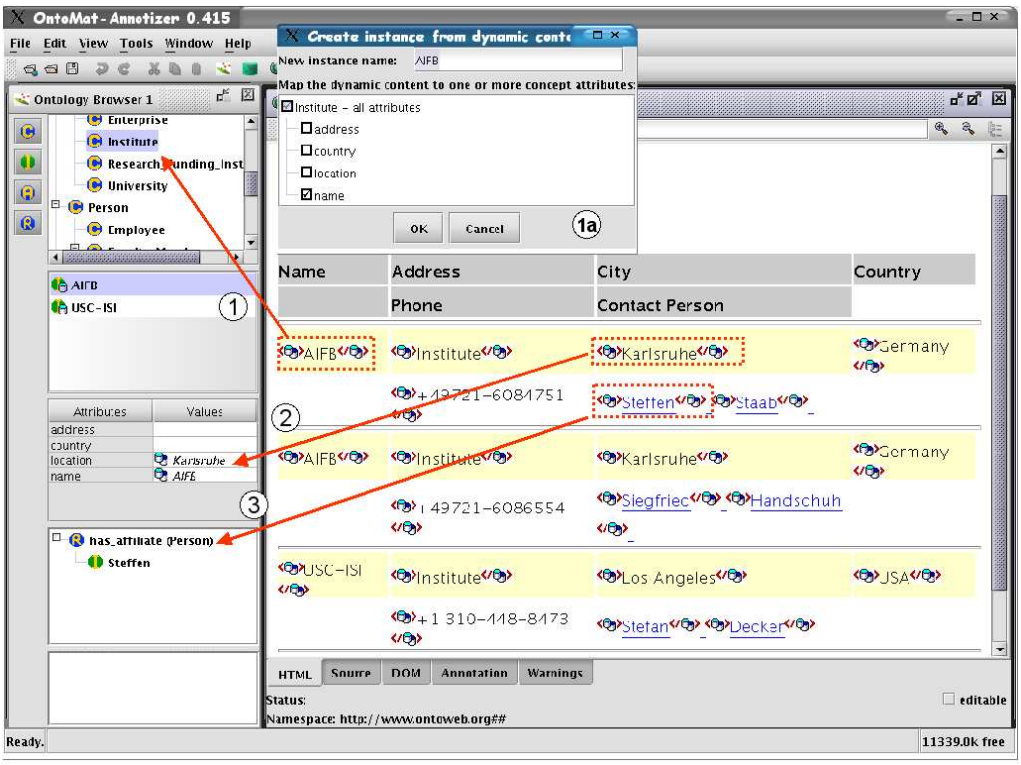

Figure 6.3.: Screenshot of Providing Deep Annotation with OntoMat-Annotizer

6.6.3. Creating Generic Attribute Instances

In order to create a generic attribute instance the user simply drops the server-side markup into the corresponding table entry (cf. arrow #2 in Figure 6.3). Generic attributes which are mapped to database table columns will also show a special icon and their value will appear in italics. Such generic attributes cannot be modified, but their value can be deleted.

When the generic attribute is filled the following steps are performed by the system:

1. Checking database definition integrity.
2. All attributes of the selected generic instance (except the generic attribute to be pasted to) are examined. The following conditions apply to each attribute:

- The attribute is empty or
- The attribute does not hold server-side markup or
- The attribute holds markup, the database name and the query id of the content on the current selection must be the same. This must be checked to ensure that result fields come from the same database and the same query. If this is not checked, non-matching information (e.g. publication titles and countries) could be queried for.

3. The generic attribute contains the information given by the markup, i.e. which column of the resulting tuple delivered by a query represents the value.

6.6.4. Creating Generic Relationship Instances

In order to create a generic relationship instance the user simply drops the selected server-side markup onto the relation of a pre-selected instance (cf. arrow #3 in Figure 6.3). As in Section 6.6.2 a new generic instance is generated. In addition, the new generic instance is connected with the preselected generic instance.

6.7. Mapping and Querying

The results of the annotation are mapping rules between the database and the client ontology. The annotator publishes[7] the client ontology and the mapping rules derived from annotations. This enables third parties (querying party) to access and query the database on the basis of the semantics that is defined in the ontology. The user of this mapping description might be a software agent or a human user.

6.7.1. Investigating Mappings

The querying party uses a corresponding tool to visualize the client ontology, to investigate the mapping and to compile a query from the client ontology. In our case, we used the OntoEdit plugins OntoMap and OntoQuery.

OntoMap visualizes the database query, the structure of the client ontology, and the mapping between them (cf. Figure 6.4). The user can control and change the mapping and also create additional mappings.

[7]Here, we used the Ontobroker OXML format to publish the mapping rules.

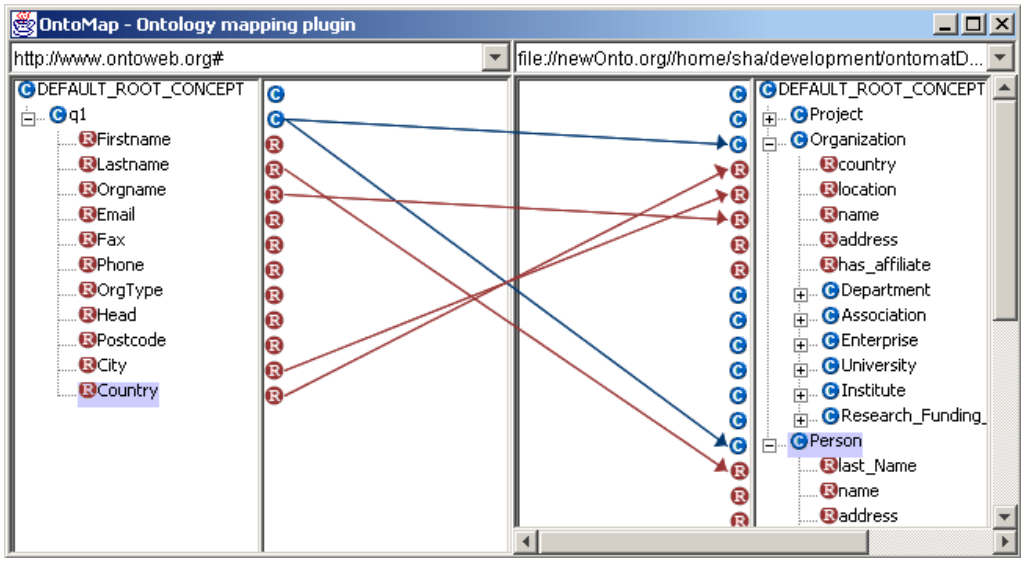

Figure 6.4.: Mapping between Server Database (left window) and Client Ontology (right window)

6.7.2. Querying the Database

OntoQuery is a Query-by-Example user interface. A query is created by clicking on a concept and selecting the relevant attributes and relationships. The underlying Ontobroker system transforms the ontological query into a corresponding SQL query. Ontobroker uses the mapping descriptions, which are internally represented as F-Logic Axioms, to transform the query. The SQL query will be send as an RPC call to the Web service, where it will be answered in the form of a set of records. These records are transformed back to an ontological representation. This task will be executed automatically, so that no interaction with the user is necessary.

For example, a query is created by selecting the concept Person. In the dialog the search can be restricted to instances of Person starting with the letter "S" in the name (cf. Figure 6.5). This ontological query is expressed as an F-Logic query and then evaluated by Ontobroker by using the mapping axioms (cf. Figure 6.6).

The data migration will be executed in two separate steps. In the first step, all the required concept instances are created without considering relationships or attributes. The instances are stored together with their identifier. The identifier is translated from the database keys by using the identifier pattern

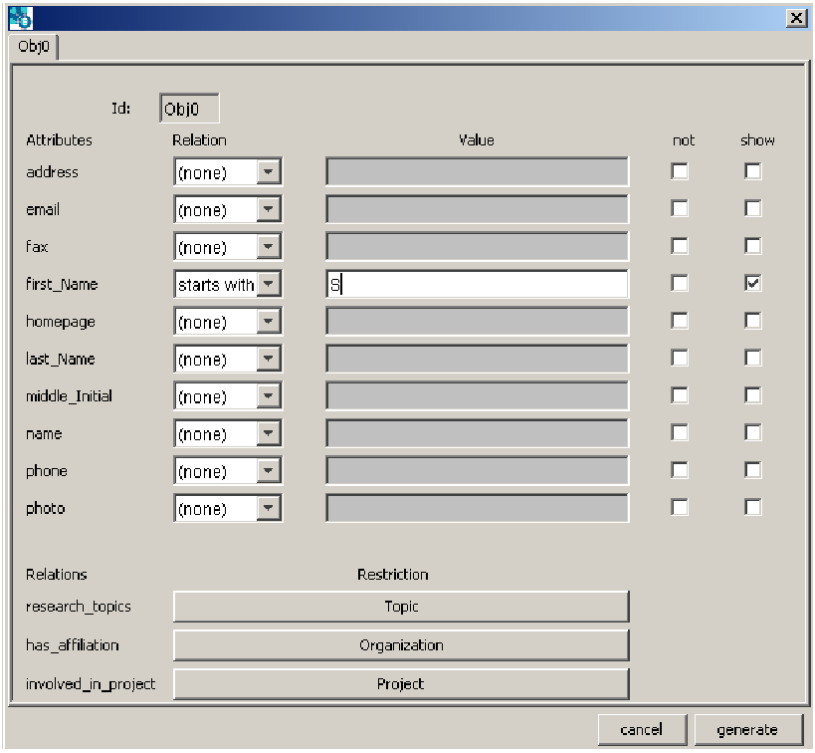

Figure 6.5.: Querying: Persons with first names starting with letter 'S'

(see Section 6.5.2). For example, the instance with the name "AIFB" of the concept Institute, which is a subconcept of Organization, has the identifier: "http://www.ontoweb.org/org/OrganizationID=3.

After the creation of all instances we begin computing the values of the instance relationships and attributes. The way the values are assigned is determined by the mapping rules. Since the values of an attribute or a relationship have to be computed from both the relational database and the ontology, we generate two queries per attribute/relationship, one SQL query and one Ontobroker query. Each query is invoked with an instance key value (corresponding database key in SQL-queries) as a parameter and returns the value of the attribute/relationship.

Note that the database communication takes place through bind variables. The corresponding SQL query is generated, and if this is the first call, it is cached. A second call would try to use the same database cursor if still available, without parsing the respective SQL statement. Otherwise, it would find an unused cursor and retrieve the results. In this way efficient access methods for relations and

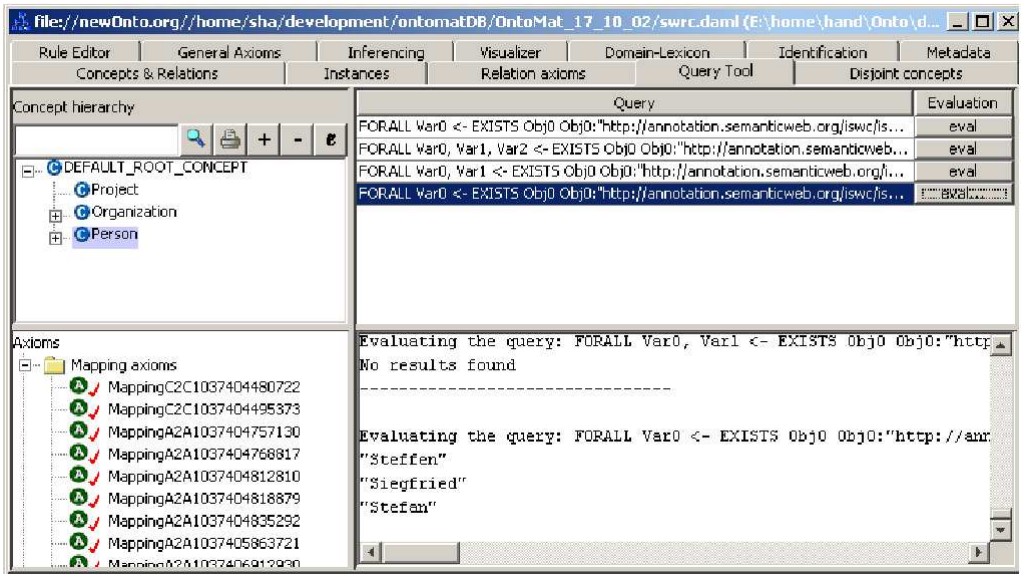

Figure 6.6.: Querying: F-Logic Query

database rules can be maintained throughout the session.

6.8. Conclusion

In this Chapter we have described *deep annotation*, an original framework to provide Semantic Annotation for large sets of data. Deep annotation leaves semantic data where they can be handled best, *viz.* in database systems. Thus, deep annotation provides a means for mapping and re-using dynamic data in the Semantic Web with tools that are comparatively simple and intuitive to use.

To reach this objective we have defined a deep annotation process and architecture. We have incorporated means for server-side markup that allows the user to define semantic mappings by using OntoMat An ontology and mapping editor and an inference engine are then used to investigate and exploit the resulting descriptions. Thus, we have provided a complete framework and its prototype implementation for deep annotation.

For the future, there is a long list of open issues concerning deep annotation — from the more mundane, though important, ones (top) to far-reaching ones (bottom):

1. Granularity: So far we have only considered atomic database fields. For instance, one may find a string "Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002." as a title of a book whereas one might rather be interested in separating this field into title, location and date.

2. Automatic derivation of server-side Web page markup: A content management system like Zope could provide means for automatically deriving server-side Web page markup for deep annotation. Thus, the database provider could be freed from *any* workload, while allowing for participation in the Semantic Web. Some steps into this direction are currently pursued in the KAON CMS, which is based on Zope[8].

3. Other information structures: For now, we have built our deep annotation process on SQL and relational databases. Future schemes could exploit XQuery[9] or an ontology-based query language.

4. Interlinkage: In the future deep annotations may even link to each other, creating a dynamic interconnected Semantic Web that allows to translate between different servers.

5. Opening the possibility to directly query the database, certainly creates problems such as new possibilities for denial of service attacks. In fact, queries, e.g. ones that involve too many joins over large tables, may prove hazardous. Nevertheless, we see this rather as a challenge to be solved by clever schemes for CPU processing time (with the possibility that queries are not answered because the time allotted for one query to one user is up) than for a complete "no go".

We believe that these options make *deep annotation* a rather intriguing scheme on which a considerable part of the Semantic Web might be built.

[8] see http://kaon.aifb.uni-karlsruhe.de/Members/rvo/kaon_portal
[9] http://www.w3.org/TR/xquery/

7. Application

In **this Chapter** we present two applications for our annotation framework on linguistic annotation (Section 7.1) and service annotation (Section 7.2). We show how to exploit the basic framework easily for linguistic markup. Furthermore, it shows how to apply the concept of deep annotation also to annotation and composition of Web services.

The structure of Section 7.1 is as follows: Section 7.1.1 presents the ontology-based framework for linguistic annotation, and Section 7.1.2 shows how the framework can be applied to the annotation of anaphoric relations. Section 7.1.3 presents the features of CREAM that are useful for the application of linguistic annotation. Finally, Section 7.1.4 gives some concluding remarks about using CREAM for linguistic annotation.

The structure of Section 7.2 is as follows: We first describe a simple use case for CREAM-Service (CREAM applied for Web services) (cf. Section 7.2.1), including a detailed WSDL description of a Web service used for the running example. In section 7.2.2, we describe the process that allows Web services to be turned into a service Web and that lets a user surfing the Web with OntoMat-Service-Surfer exploit the very same tool to aggregate and invoke Web services. The first step of this process, i.e. advertising Web services in a form that combines presentation for human and machine agent consumption, is sketched in section 7.2.3. The second step of this process, i.e. using browsing and semantic deep annotation to tie together conceptual descriptions, is detailed in section 7.2.4.

References: Section 7.1 is mainly based on [Cimiano and Handschuh, 2003] and Section 7.2 is mainly based on [Agarwal *et al.*, 2003].

7.1. Linguistic Annotation

Linguistic annotation (cf. Section 3.4.3) is crucial for the development and evaluation of natural language processing (NLP) tools. In particular machine-learning based approaches to part-of-speech tagging, word sense disambiguation, information extraction or anaphora resolution - just to name but a few - rely on corpora annotated with the corresponding phenomenon to be trained and tested. In this Section, we argue that linguistic annotation can to some extent be considered a special case of Semantic Annotation with regard to an ontology. Part-of-Speech

(POS) annotation for example can be seen as the task of choosing the appropriate tag for a word from an ontology of word categories (compare for example the Penn Treebank POS tagset as described in [Marcus *et al.*, 1993]). The annotation of word senses such as used by machine-learning based word sense disambiguation (WSD) tools corresponds to the task of selecting the correct semantic class or concept for a word from an underlying ontology such as WordNet [Resnik, 1997]. Annotation by template filling such as that used to train machine-learning based information extraction (IE) systems as (cf., Section 5.1.2) can be seen as the task of finding and marking all the attributes of a given ontological concept in a text. An ontological concept in this sense can be a launching event, a management succession event or a person together with attributes such as name, affiliation, position, etc. The annotation of anaphoric or bridging relations is actually the task of identifying the semantic relation between two linguistic expressions representing a certain ontological concept.

Most linguistic annotation tools make use of schemes specifying what can actually be annotated. These schemes can in fact be understood as a formal representation of the conceptualization which underlies the annotation task. Ontologies are formal specifications of a conceptualization (cf. Section 2.5) so that it seems the logical next step to formalize annotation schemes as ontologies and make use of our annotation framework (cf. Chapter 4) and the corresponding tool OntoMat for the purpose of linguistic annotation.

7.1.1. The Ontology-based linguistic annotation framework

An ontology is a structure $\mathcal{O} := (C, \leq_C, R, \leq_R, A)$ as defined in Definition 2.5.1 (page 27).

Our framework basically offers three ways of annotating a text with regard to an ontology:

- a linguistic expression appearing in a text can be annotated as an instance of a certain ontological concept $c \in C$
- a linguistic expression in a text can be annotated as an attribute instance of some other linguistic expression previously annotated as a certain concept $c \in C$
- the semantic relation between two linguistic expressions respectively annotated as instances of two concepts $c_1, c_2 \in C$ can be annotated as an instance of relation r.

The advantages of an ontology-based linguistic annotation framework as described above are the following:

- The formalization of the annotation scheme as an ontology as well as the use of standard formalisms such as RDF (cf. Section 2.4) or OWL (cf. Section 2.6 to encode it, allow the scheme to be reused across different annotation tools. This meets the interoperability requirement mentioned in [Ide, 2002].
- The specification of the annotation task, i.e. the annotation scheme, can be performed in an arbitrary ontology development environment and thus becomes completely independent of the annotation tool actually used.
- The ontology-based linguistic annotation model offers the kind of flexibility mentioned in [Ide, 2002] in the sense that it is general enough to be applied to a broad variety of annotation tasks.
- The fact that annotation is performed with respect to an ontological hierarchy offers annotators the possibility of choosing the appropriate level of annotation detail so that they are never forced to overspecify, i.e. to annotate more specifically than they actually feel comfortable with.

In addition, hierarchical linguistic annotation offers further possibilities regarding the computation of the agreement between different annotators as well as the evaluation of a system against a certain annotation. In this sense, instead of measuring only the categorial agreement between annotators with the kappa statistic [Carletta, 1996] or the performance of a system in terms of precision/recall, we could take into account the hierarchical organization of the categories or concepts by making use of measures considering the "hierarchical distance" between two concepts (cf. Section 8.4.2).

Furthermore, the use of an ontology-based and thus more semantic framework for linguistic annotation has two further, very interesting properties. On the one hand, the use of an ontology helps to constrain the possible relations between two concepts, thus reducing the amount of errors in the annotation process. For example when annotating *Coreference*-relations in a text, it seems obvious that an event and an entity will never confer and in fact such an erroneous annotation can be avoided if the underlaying ontological model actually forbids such an annotation (see below). Furthermore, by using axioms for example stating that *Coreference* is reflexive, symmetric and transitive - thus representing an equivalence relation - the evaluation of systems becomes much easier and more straightforward when using an inference machine such as the annotation inference server module of CREAM (cf., Section 7.1.3 and 4.4.1).

If an annotator for example annotates the following coreferences: Coreference(A,B) and Coreference(B,C) a system's answer such as Coreference(A,C) will actually be deemed correct due to the fact that *Coreference* is defined as a transitive relation within the ontology.

7.1.2. Annotating anaphoric relations

Before showing how our framework can be applied to the annotation of anaphoric relations in written texts, the assumptions underlying our model need to be explained. Firstly, we aim at a more semantic annotation of anaphoric relations than for example described in [Müller and Strube, 2001] because we think that such a model can to some extent be subsumed by the one we propose. In fact, we will interpret the term *anaphoric* in a much wider sense in line with [Krahmer and Piwek, 2000] and [van Deemter and Kibble, 2000]. They argue for example that coreference is not a necessary property of anaphora such as proposed in [Müller and Strube, 2001]. So annotating the relation between two expressions as anaphoric will correspond to the most general relation in our hierarchy. In particular, *Identity* or *Coreference* will only be a special type of anaphoric relation (compare Figure 7.2) in our model.

On the other hand, *bridging* will be defined in our framework in line with [Asher and Lascarides, 1999] as "the inference that two objects or events that are introduced in a text are related in a particular way that isn't explicitly stated". Thus *Coreference* or *Identity* can represent an anaphoric relation or more specifically a bridging reference depending on whether or not the identity relation is explicit or not. Consider the following minimal pair:

(7.1) John bought a car yesterday. The car was in a good state.

(7.2) John bought a car yesterday. The vehicle was in a good state.

In example (7.1), the anaphoric relation is explicit due to the matching heads of the NPs *a car* and *The car*. In (7.2) the anaphoric or bridging relation is not explicit as world knowledge that cars are vehicles is needed to resolve the reference. In the semantics-based model for the annotation of anaphoric relations we propose in this Chapter, both examples will in fact be annotated as instances of the *Coreference* or *Identity* relation. Consequently, we will completely omit the concept *bridging reference* in the ontology underpinning the annotation. In fact, we claim that the classification of an anaphora as a bridging reference, direct anaphora, pronominal anaphora, etc. such as pursued in [Müller and Strube, 2001] can be seen as a byproduct of a more semantic classification as proposed here if additional grammatical information provided by the annotators is available. This grammatical information can be added to the concepts depicted in Figure 7.1.1 in the form of attributes specifying the grammatical form of the expression, i.e. whether it is for example a noun, an NP, a pronoun, a verb or a VP, as well as information about its head, gender or tense. The semantic classification proposed here together with the grammatical information modeled as attributes of a concept will then yield a classification as envisioned by [Müller and

Strube, 2001]. For example, if two expressions are annotated as *coreferring*, this semantic relation can be further classified as *nominal anaphora* if the referring expression is a pronoun, as *direct anaphora* if the heads of the expression match or as a *bridging reference* otherwise. On the other hand, all the *Non-Identity* relations modeled in the ontology underlying the annotation task will lead to a classification as a bridging reference (compare Figure 7.2). However, it should be mentioned that we do not aim at such a "grammatical" classification of anaphoric relations. We envision a task as in [Asher and Lascarides, 1999], where bridging reference resolution corresponds to the task of finding the discourse referent serving as antecedent as well as the semantic relation between this discourse referent and the one of the referring expressions.

In our model, an expression can be antecedent for more than one referring expression, an assumption which seems to be commonly shared by many annotation schemes. However, in our model a certain expression can also refer to more than one antecedent. [Poesio and Reyle, 2001] for instance show that the antecedent of a referring expression can in fact be ambiguous in a way that the overall interpretation of the expression or sentence is not affected. Furthermore, [Poesio and Reyle, 2001] argue that it is not clear whether the addressees of an utterance actually are aware of all the possible antecedents for a certain referring expression, if they underspecify the antecedent of a referring expression in case the overall interpretation is not affected or if they just choose one of the possible antecedents without being aware of the other ones. In any case, a model for the annotation of anaphoric or bridging relations should not *a priori* exclude that referring expressions can have more than one antecedent. Consequently, the annotation of the semantic relation between a referring expression and an antecedent can neither take place at the antecedent nor the referring expression such as in [Müller and Strube, 2001], but in a functional way, i.e. at a virtual edge between them.

The ontology underlying our annotation scheme is depicted schematically in Figure 7.1 We distinguish two types of eventualities: events and states, and model the discourse relations described in [Lascarides and Asher, 1991] as semantic relations between them. In addition, we distinguish between three types of (meta-) entities: sets of entities, intensional entities [van Deemter and Kibble, 2000] and (real-world) entities together with the potential relations such as *member_of*, *part_of*, etc. between them as well as to other types: An *entity* for example can play a certain thematic role in some *event* (compare Figure 7.1).

With such a concept hierarchy as well as semantic relations with a precisely defined signature, we can for example overcome annotation problems of intensionality and predication as discussed in [van Deemter and Kibble, 2000]. In order to profit from the benefits of a hierarchical annotation, we also define a hierarchy for the semantic relations (see Figure 7.2). Thus if annotators, for example, feel that there is an anaphoric relation between two linguistic expressions, but can not specify the type of relation, they can choose the most general relation in the

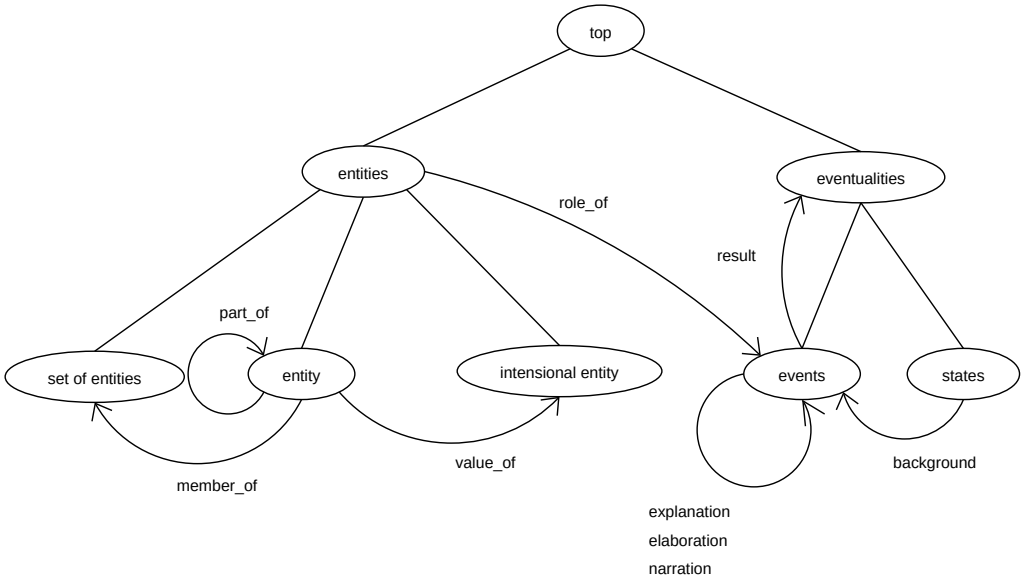

Figure 7.1.: The ontology underlying the annotation scheme

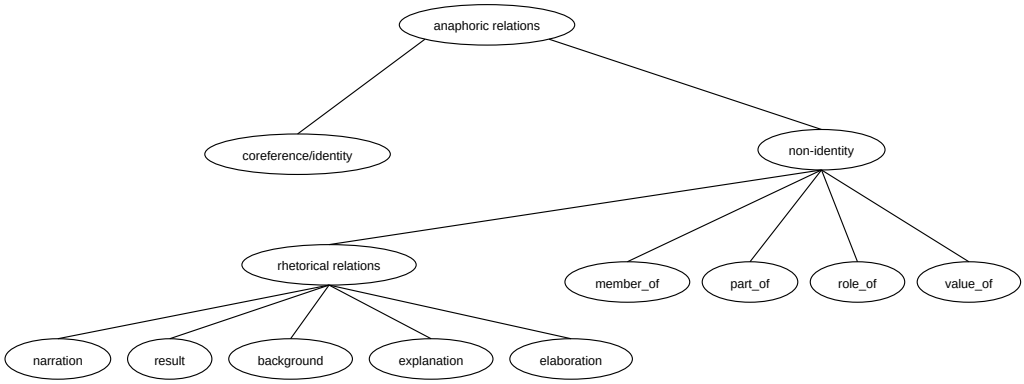

Figure 7.2.: The hierarchical organization of the semantic relations.

hierarchy, i.e. *anaphoric relation*. As mentioned in Section 7.1.1, the idea is that annotators are never forced to overspecify and can annotate at the hierarchical level they feel comfortable with.

7.1.3. CREAM and OntoMat

The CREAM framework and the concrete implementation OntoMat is described in detail in Chapter 4. The framework itself is developed for the creation of ontology-based annotation in the context of the Semantic Web. However, with an appropriate ontology one can also take advantage of the framework and use it for linguistic annotation. In the subsequent section we will briefly emphasize the features that are relevant to this new purpose (for an extensive description of the CREAM modules cf. Section 4.4.1).

CREAM Features

User Interface. OntoMat's *document viewer* visualizes the document contents. The *Ontology and Fact Browser* is the visual interface to the ontology and the annotated facts. Both the *Ontology and Fact Browser* and the *document editor/-viewer* are intuitive to use: Drag'n'drop helps to avoid syntax errors and typographical errors and a good visualization of the ontology helps the annotators to choose correctly the most appropriate class for an instance (compare Figure 7.3).

Annotation. An annotation in our context is a set of instantiations of classes, relationships and attributes. These instances are not directly embedded into the text, but are pointing to appropriate fragments of the document. The link between the annotation and the document is done by using XPointer (cf. Section 2.3) as an addressing mechanism. This has some advantages with regard to the flexibility of annotation as it allows (i) multiple annotation (ii) nested annotation and (iii) overlapping annotation of text segments.

Annotation Inference Server. The annotation inference server reasons based on the instances and on the ontology. In doing so, it also takes into account the axioms modeled within the ontology and can thus be used in the evaluation of a certain system such as described in Section 4.4.1

Storage. CREAM supports different ways of storing the annotation. This flexibility is given by the XPointer technique which allows the separation of the annotation from the document. Hence, the annotations can be stored together with the document. Alternatively or simultaneously it is also possible to store them remotely, either in a separate file or in the annotation inference server.

Annotating anaphoric relations

The ontology described in Section 7.1.2 is available in the form of DAML+OIL[1] classes and properties, in OWL, as pure RDF-Schema and in F-Logic. In the following, we explain briefly how OntoMat can be used for the creation of instances consistent with the ontology in question.

Figure 7.3 shows the screen for navigating the ontology and creating annotations in OntoMat. The right pane displays the document and the left pane shows the ontological structures contained in the ontology, namely classes, attributes and relations. In addition, the left pane shows the current Semantic Annotation knowledge base, i.e. existing class instances, attribute instances and relationship instances created during the Semantic Annotation. First of all, the user browses

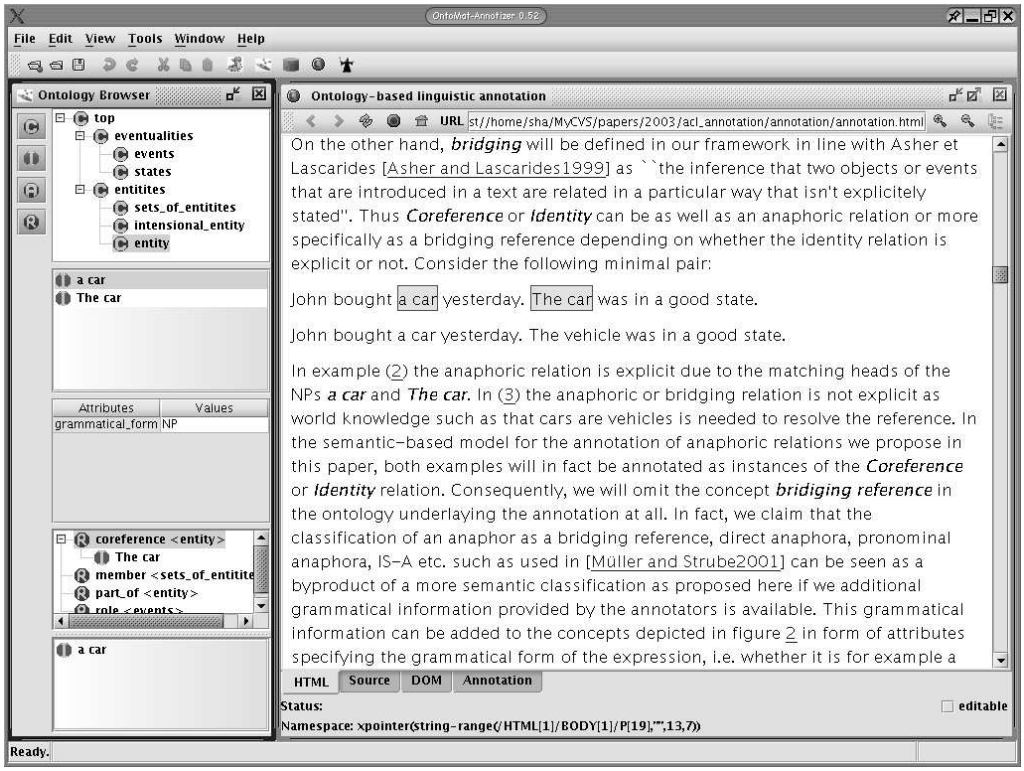

Figure 7.3.: Annotation Tool Screenshot.

a document by entering the URL of the Web document that he would like to annotate. Then he loads the corresponding ontology into the ontology browser. He selects a text fragment by highlighting it. There are two ways in which the

[1] http://annotation.semanticweb.org/ontologies/AnaphOnto.daml

text fragment may be annotated: as an instance or as a relation. In the case of an instance, the user selects in the ontology the class in which the text fragment fits in, e.g. for the expression "a car" in Example 7.1, he would select the class *entity*. By clicking on the class, the annotation is created and thus the text fragment will be displayed as an instance of the selected class in the ontology browser. The relationships between the created instances can be specified, e.g. the entity *The car* can be annotated as coreferring with the preceding entity *a car* as described in Section 7.1.1. For this purpose, when selecting a certain class instance as well as a corresponding semantic relation from the ontology, OntoMat already presents the possible target class instances according to the range restrictions of the chosen relation. Hereby erroneous annotations of relations are avoided (compare Section 7.1.1). Furthermore, literal attributes can be assigned to every created instance by typing them into the related attribute field. The choice of the predefined attributes depends on the class to which the instance belongs. Thus, instances of a certain concept can be annotated with grammatical information about how they are linguistically expressed, i.e. through an NP, a noun, a pronoun, a verb, etc. (compare Section 7.1.2).

7.1.4. Conclusion

We have argued that many linguistic annotation tasks can be seen as a special case of Semantic Annotation with regard to an ontology, and have proposed a novel ontology-based framework for this purpose. Furthermore, we have applied our framework to the annotation of anaphoric relations in written texts. For this purpose we have proposed a relatively complex annotation scheme for anaphoric relations in which we have deliberatively abstracted from important issues such as inter-annotator agreement. In fact, the main contribution of this Section is to show that relatively complex annotation schemes such as the one proposed can be modeled in our ontology-based framework in a straightforward manner. The main benefits of the approach presented here are that the annotation can be performed at different levels of detail with regard to a given taxonomy, and that the possible relations between two different concepts are constrained by the underlying ontology, which could make the annotation less error-prone. In addition we have shown how the modeling of axioms within the ontology can actually make the evaluation of a system more straightforward. The most important advantage is that by specifying the annotation scheme in form of an ontology and adhering to standards such as RDF or OWL, it can be easily exchanged between different parties and can also be developed independently of the annotation tool used, which meets the interoperability requirement mentioned in [Ide, 2002]. In addition, our framework is flexible enough to be applied to various annotation tasks, which is also a requirement mentioned in [Ide, 2002].

7.2. Service Annotation

The Stencil Group defines Web services as: loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard internet protocols. Though this definition captures the broad understanding of what Web services are, it raises the question, what Web services have to do with the web. Even if HTTP is used as a communication protocol and XML/SOAP to carry some syntax this appears to be a rather random decision than a deeply meaningful design.

We believe that it makes sense to actually integrate the strengths of the conventional World Wide Web, viz. lightweight access to information in a highly-distributed setting, with the strengths of Web services, viz. execution of functionality by lightweight protocols in a highly-distributed setting. To seamlessly integrate the two aspects we envision a *service web* that uses XHTML/XML/RDF to transport information and a Web service framework to invoke operations and a framework, CREAM-Service, to bind the two aspects together. CREAM-Service offers an infrastructure, OntoMat-Service-Surfer, that allows

- for seamlessly browsing conventional Web pages, including XHTML advertisements for Web services;
- for direct, manual invocation of an advertised Web service as a one-off use of the service;
- for tying Web service advertisements to each other when browsing them;
- for tying Web service advertisements to one's own conceptualization of the Web space when browsing them; and
- for invoking such aggregated Web services.

For these objectives, we build on existing technologies like RDF (cf., Section 2.4), ontologies (cf., Section 2.5) or WSDL [W3C, 2003]. To integrate the Web and Web services into the service Web, we make specific use of a new type of *Semantic Annotation* (cf. Chapter 4), namely *deep annotation* (cf. Chapter 6).

7.2.1. Use Case

A typical use case supported by CREAM-Service is the following (adapted from a larger scenario in [Maedche and Staab, 2003]): An employee in a small enterprise needs a new laptop. In order to buy one he defines the characteristics of the laptop like processor speed, disk size, etc. Based on the configuration of the

laptop he collects offers from laptop vendors. When he receives an offer he also solicits insurance terms from a third party. Once the most reasonable laptop and the best insurance contract terms are determined, the employee purchases the laptop and closes the service contract.

In our scenario, we assume a laptop vendor and an insurer offering Web services with two operations each, i.e. `getLaptopOffer` / `buyLaptop` and `getInsuranceTerms` / `closeServiceContract`, respectively. The sequence of operations that must be executed by the customer is depicted in Figure 7.4.

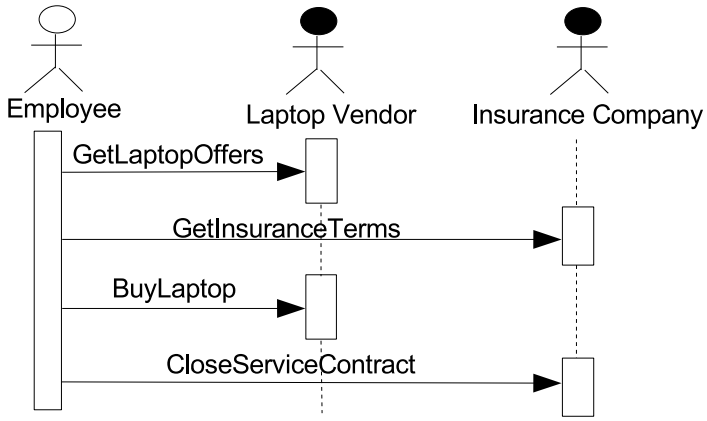

Figure 7.4.: Sequence Diagram for the Use Case

The laptop vendor and the insurer being Web service providers describe their Web services with WSDL documents. In Listing 7.1, we show how a conventional WSDL document of the laptop vendor located at `http://laptop-vendor.de/laptop.wsdl` might look like.[2]

The WSDL document describes:

- *Data type definitions* in the XML element `types`. They are only sketched in Listing 7.1 as they correspond to the laptop vendor's ontology depicted in N3 (cf., Section 2.4.4) in Listing 7.3. Thereby, we assume the definitions given in Listing 7.2. In our running example, the WSDL document of the laptop vendor, we describe the class LAPTOP.
- *Messages* that a service sends and/or receives and that constitute the Web service operations in the XML element `portType`. For instance, our

[2]The single ideosyncrasy we have here is that the WSDL document employs RDFS in order to describe the data structures instead of the more common XML schema — though actually WSDL does not require XML Schema and it allows RDFS.

```
<?xml version="1.0" encoding="UTF-8"?> <definitions
name="LaptopService"
    targetNamespace="http://laptop.wsdl/laptop/"
    <types>
       <rdf:RDF>
          <rdfs:Class rdf:ID="Laptop">
             <rdfs:label>Laptop</rdfs:label>
          </rdfs:Class>
          <rdf:Property rdf:ID="diskSpace">
             <rdfs:label>diskSpace</rdfs:label>
             <rdfs:range rdf:resource="&rdfs;Literal"/>
             <rdfs:domain rdf:resource="#Laptop"/>
          </rdf:Property>
           ...
          <rdf:Property rdf:ID="price">
             <rdfs:label>price</rdfs:label>
             <rdfs:range rdf:resource="&rdfs;Literal"/>
             <rdfs:domain rdf:resource="#Laptop"/>
          </rdf:Property>
           ...
       </rdf:RDF>
    </types>
    <message name="getOffersRequest">
        <part name="processorSpeed" type="processorSpeed"/>
        <part name="diskSpace" type="diskSpace"/>
    </message>
    <message name="getOffersResponse">
        <part name="laptopOffers" type="laptops"/>
    </message>
        ...
    <portType name="LaptopService">
        <operation name="getLaptopOffers"
        parameterOrder="processorSpeed diskSpace">
         <input message="tns:getOffersRequest" name="getOffersRequest"/>
         <output message="tns:getOffersResponse" name="getOffersResponse"/>
        </operation>
        ...
    </portType>
</definitions>
```

Listing 7.1: Web Service Description of Laptop Vendor

running example specifies '`getOffersRequest`' that a potential customer would send to the laptop vendor to solicit an offer. `getOffersRequest` must be provided with two arguments, namely processor speed and disk size. It returns a set of laptop offers with properties such as specified in the vendor ontology (cf. WSDL document in Listing 7.1 and vendor ontology in Listing 7.3).

WSDL provides a naming convention for URIs such that each conceptual element (e.g., `types`, `portType`, etc.) of a WSDL document can be uniquely referenced. Such a URI consists of a `targetNamespace` pointing to the location of the WSDL

document and to element names of the WSDL document. For example, the URI `http://laptop.wsdl/laptop/#part(getOffersRequest/diskSpace)` refers to the second part (`diskSpace`) of the message `getOffersRequest` of the WSDL document in Listing 7.1 (cf. [W3C, 2003] for further specifications).

The Web service description of the insurer looks similarly. We here only mention that the insurer provides the operations `getInsuranceTerms` and `closeServiceContract`. `getInsuranceTerms` requires a description of LAPTOP (according to the insurer's ontology in Listing 7.4) and a TIMEPERIOD, for which the contract is supposed to run. `getInsuranceTerms` returns a set of insurance terms available.

```
@prefix rdfs: <http://www.w3.org/rdf-schema#>. @prefix : <#>.
@prefix a rdf:Type.
```

Listing 7.2: N3 shortcuts

```
:Laptop a rdfs:Class.
:price rdfs:domain :Laptop; rdfs:range :rdfs:Literal.
:diskSpace rdfs:domain :Laptop; rdfs:range :rdfs:Literal.
:processorSpeed rdfs:domain :Laptop; rdfs:range :rdfs:Literal.
:laptopID rdfs:domain :Laptop; rdfs:range :rdfs:Literal.

:Offer a rdfs:Class.
:laptops rdfs:domain :Offer; rdfs:range :Laptop.

:Sale a rdfs:Class.
:laptop rdfs:domain :Sale; rdfs:range :Laptop.
:creditCardNumber rdfs:domain :Sale; rdfs:range :Literal.
:customerReceipt rdfs:domain :Sale; rdfs:range :Literal.
```

Listing 7.3: Ontology of the laptop vendor

```
:Laptop a rdfs:Class.
:id rdfs:domain :Laptop; rdfs:range :Literal.

:ContractTerms a rdfs:Class.
:laptop rdfs:domain :ContractTerms; rdfs:range :Laptop.
:timePeriod rdfs:domain :ContractTerms; rdfs:range :Literal.
:price rdfs:domain :ContractTerms; rdfs:range :Literal.
```

Listing 7.4: Ontology of the insurance company

In the remainder of the paper, we assume that the customer has the plan depicted in Figure 7.4. However, in our running example, we will mostly focus on the first two steps to illustrate our framework.

7.2.2. Overview of the Complete Process of CREAM-Service

Figure 7.5 shows the complete process of our framework, CREAM-Service. First, the Figure consists of process steps, which are illustrated by a circle representing the step and a person icon representing the logical role of the person who executes the step, viz. service provider, annotating Service Web surfer and a user invoking a Web Service. The two latter roles typically coincide. Second, the Figure comprises information that is used by a person or by OntoMat-Service-Surfer in a process step.

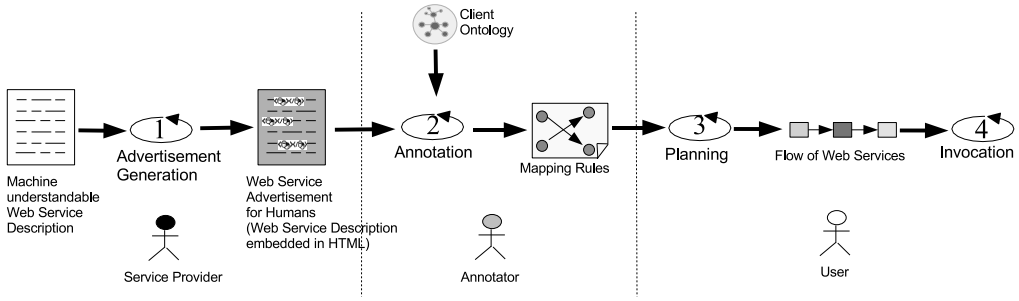

Figure 7.5.: The Complete Process of OntoMat-Service

The four main steps run as follows:

Init: CREAM-Service starts with a common WSDL Web service description by the service provider (e.g., Listing 7.1). Obviously, the WSDL document is primarily intended for use by a machine agent or a software engineer who has experience with Web services. It is not adequate for presenting it to a user who is 'only' expert in a domain.

Web Service Presentation (Step 1): In the first step, the Web service provider makes the Web service presentation readable as a nicely formatted (X)HTML document — possibly including advertisements, cross-links to other HTML pages or services, or other items that make the Web page attractive to the potential customer (cf. Section 7.2.3 for details).

Thereby, it is important that the understandable, but informal description of the Web service is implicitly annotated to relate the textual descriptions to their corresponding semantic descriptions in their WSDL document.

Step 1 is a manual step that may be supported by tools such as *WSDL Documentation Generator* from *http://www.xmlspy.com*. However, we would not assume that tools like *WSDL Documentation Generator* would be sufficient to generate an amenable presentation, as they still produce rather rigid and technically oriented descriptions.

Result 1: Human understandable Web page that advertises the Web service and embeds/refers to machine understandable Web service descriptions (WSDL + ontology).

Deep Annotation (Step 2): At a client side, a potential user of the Web service browses the Web page. OntoMat-Service-Surfer shows the Web page like a conventional browser. In addition, OntoMat-Service-Surfer highlights human-understandable items (e.g. text phrases) that associate an underlying machine-understandable semantics.

The logical role of the user here is one of an annotator/surfer. He can decide to just view the page and proceed directly to step 4 (described below). Alternatively, he can decide to map some of the terminology used in the Web page of the Web service to his own terminology (or to the terminology of someone else).

For the latter purpose, he loads an ontology into OntoMat-Service-Surfer (if it is not already pre-loaded). Then he aligns terminology mentioned in the Web page by drag'n'drop-ping it onto the ontology loaded into OntoMat-Service-Surfer. OntoMat-Service-Surfer generates mapping rules from these annotations that bridge between the ontology of the service provider and the ontology loaded into OntoMat-Service-Surfer (cf. Section 7.2.4 for details).

Typically, the user will map to more than one Web service, i.e. often he will map to different ontologies.

Result 2: Sets of mapping rules between Web service ontologies and pre-loaded ontology.

Web Service Planning (Step 3): At the client side, a user might view the Web services as well as their annotations that yield mapping rules. The third logical role here is one of a service planner and invocator (this logical role is shared between the third and fourth step). For this purpose, the user decides to select

- a set of Web service operations he wants to use and
- a set of mapping rules he wants to use.

The reader may note that very frequently the roles of an annotator/surfer and a service invocator will just coincide. Hence, the two selections just mentioned will take place implicitly — just by the Web service pages he has browsed and the annotations that the service invocator has performed in step 2 of the CREAM-Service process.

Once the two selections have been performed im- or explicitly, a module for Web service planning will compute logically possible Web service flows. For

this objective, Web service planning may employ a rich set of knowledge: goals, pre-conditions of web services, post-conditions of Web services, previous similar cases, etc. In the current version of CREAM-Service we just exploit the pre- and post-conditions derived from mapping one Web service output to another Web service input via the customer ontology. The Web service description in the associated WSDL document describes what types are required for the input of a Web service and what types appear in the output of a Web service. Since data that wanders from one Web service to the next can only proceed if types are compatible, OntoMat-Service-Surfer can compute a restricted set of possible Web service flows.

Though in general this model may be too weak to compute complex flows it is quite sufficient and straightforward to use with a small number of selected and semantically aligned Web services — such as an end user or prototype builder will use.

Result 3: Sets of possible Web service flows.

Web Service Invocation (Step 4): The final user, i.e. the invocator, can select one such flow from the list or modify any, if none of them fits his needs. Obviously, he can always create a new flow on his own. Once the user has a flow that fulfills his current needs, he invocates the flow. During the execution, the transformation of the data of one ontology to another will happen automatically via the mapping rules. The user achieves his goal at the completion of the invocation of the Web service flow.

7.2.3. Semantic Web Page Markup for Web Services

In this section we show how a Web service provider can semantically annotate the Web pages describing his Web services. Such a combined presentation allows for improved ways to find the Web services (e.g., by a combined syntactic/semantic search engine) and it enables a user to understand the functionality of a Web service and define mapping rules between the ontology used in the Web service description and the client's ontology.

The basic idea is that a conventional HTML page about the Web service and Web service parameters is extended by URIs referring to conceptual elements of the corresponding WSDL documents. To carry these two pieces of information, we use `wsdlLocation` and `elementURI` inside the `span` tags. In Listing 7.5, we show how such a Web service advertisement(HTML page) for the laptop vendor service might look like.

When such an HTML page is opened in OntoMat-Service-Surfer, the `span` tags are interpreted and elements between `<span>` and `</span>` are highlighted to support the annotation step described in the next section.

```
<html><head>
<title>Laptop Vendor Service</title></head>
<body>
<h1 align="center">Laptop Vendor Service</h1><p>
<h2>getLaptopOffers</h2>
This service delivers the top offers of the laptops available in
the city. We have the largest archive of the laptop offers for the
city. So, the possibility that you find your desired laptop at a
reasonable price is very high. Just try it and get convinced from
our great offers.
<ul>
<li> <span {\bf{} wsdlLocation="http://laptop-vendor.de/laptop.wsdl"
elementURI="http://laptop.wsdl/laptop/#part(
getOffersRequest/processorSpeed)"}>
<b>Processor speed</b> </span> Specifies the speed of the
processor. Please use only the units "MHz" and "GHz". For example,
"2GHz", "1.4GHz" and "1600MHz" are valid whereas "1800" or
"170000KHz" are invalid. </li>

<li> <span {\bf{} wsdlLocation="http://laptop-vendor.de/laptop.wsdl"
elementURI="http://laptop.wsdl/laptop/#part(
getOffersRequest/diskSpace)"}>
<b>Disk space</b> </span> Specifies the disk space. Please use
only the units "GB" and "MB". For example, "20GB", "30.5GB" are
valid whereas "40" or "25000KB" are invalid. </li>

<li> <span {\bf{} wsdlLocation="http://laptop-vendor.de/laptop.wsdl"
elementURI="http://laptop-vendor.wsdl/laptop/#part(
getOffersResponse/laptopOffers)"}>
<b>Top Offers</b> </span> This is the list of the most reasonable
offers available in the city that fulfill your requirements.
</li>
</ul></p>                    ...
</body></html>
```

Listing 7.5: Web Service Description embedded in HTML Page

7.2.4. Browsing and Deep Annotation

In the following, we describe the second main step of the CREAM-Service process. This step consists of browsing Web pages about Web services with OntoMat-Service-Surfer. Thereby, the user may annotate (cf. Chapter 6) these Web pages generating mapping rules between a client ontology and the ontologies referred to in the WSDL documents as a 'side effect' of annotation. We call this action 'deep-annotation' as its purpose is not to provide semantic annotation about the surface of what is being annotated, this would be the Web page, but about the semantic structures in the background, i.e. the WSDL elements.[3]

Thus, this step is about Web service discovery by browsing and using information

[3]Chapter 6 goes into detail for using deep annotation as the basis of database integration.

retrieval engines like Google as well as about reconciling semantic heterogeneity between different Web services, such as described in the WSDL documents and the Web service ontologies they embed or refer to.

Service Browsing

With OntoMat-Service-Surfer the user can surf the service Web, i.e. he can browse the Web pages of Web service advertisements and OntoMat-Service-Surfer highlights Semantic Annotations added by the Web service provider. OntoMat-Service-Surfer indicates semantically-annotated Web service elements, e.g. input parameters, by graphical icons on the Web page. Thus, the user may easily identify relevant terminology that needs to be aligned with his own ontology.

Deep Annotation

The user selects an ontology to be used for annotation and loads it into OntoMat-Service-Surfer. The user annotates the Web service by drag'n'dropping highlighted items from the Web page into the ontology browser of OntoMat-Service-Surfer. Doing so, he could extend the Web page with metadata if he has write access, primarily however he establishes mappings between concepts, relations and attributes from the ontology used by the Web service provider to his client ontology (cf. Chapter 6).

In the following we describe the deep-annotation of the vendor Web service shown in Figure 7.6. The Web page advertising the Web service describes the `getLaptopOffer` operation and constitutes the context for the usage of the vendor ontology. The aim of the annotator is to translate the terminology used in the description of `getLaptopOffer` (cf. the WSDL document in Listing 7.1 and the vendor ontology in Listing 7.3) into his client ontology (Listing 7.6).

By drag'n'drop, one generates a graph of instances, relations between instances and attribute values of instances in the browser that visualizes the client ontology (cf. the left pane depicted in Figure 7.6).

When performing a drag'n'drop one will create a *literal instance*, if one drops

1. an instance of the vendor ontology onto a concept in the client ontology, or
2. a literal value onto a concept of the client ontology, or
3. if one drop's an attribute value of an instance onto an attribute in the client ontology.

For instance, dropping 'IBM' onto the concept company would create a corresponding literal instance in the client ontology, dropping '7MB' onto a size

```
:Product a rdfs:Class.
:id rdfs:domain :Product; rdfs:range :Literal.

:HardDisk a :Product.
:diskSize rdfs:domain :HardDisk; rdfs:range :Literal.
:Computer a :Product.
:hasHDD rdfs:domain :Computer; rdfs:range :HardDisk.
:price rdfs:domain :Computer; rdfs:range :Literal.
:cpuSpeed rdfs:domain :Computer; rdfs:range :Literal.

:Agent a :rdfs:Class.
:Company a :Agent.
:creditCardNumber rdfs:domain :Company; rdfs:range :Literal.

:Purchase a :rdfs:Class.
:hasBuyer rdfs:domain :Purchase; rdfs:range :Agent.
:hasObject rdfs:domain :Purchase; rdfs:range :Product.

:Insurance a :rdfs:Class.
:hasObject rdfs:domain :Insurance; rdfs:range :Product.
:price rdfs:domain :Insurance; rdfs:range :Literal.
:timePeriod rdfs:domain :Insurance; rdfs:range :Literal.
```

Listing 7.6: Ontology of the client

attribute of a selected instance creates a corresponding attribute value for this selected instance in the client ontology.

When performing a drag'n'drop one will create a *generic instance*, if one drops

- a concept A from the vendor ontology onto a client ontology concept B.

A generic instance is just a variable that states that concept A in the vendor ontology corresponds to concept B in the client ontology.[4]

Thus, one may augment the client ontology (represented in RDF by a graph G) by a graph G_Δ of new and different types of instances.[5] Each subgraph of G_Δ of non-separable, newly created instances and values in the client ontology corresponds to a mapping rule. For instance, one may *(i)*, drag'n'drop 'processorSpeed (from vendor ontology) onto cpuSpeed (from client ontology) that belongs to Computer (again in the client ontology). Thereby, *(ii)*, a generic instance is created for Computer with value Laptop (as cpuSpeed belongs to Computer and processorSpeed belongs to Laptop).

The corresponding interpretation in first-order logic is:

```
FORALL X (instanceOf(X,client:Computer) AND client:cpuSpeed(X,Y)) <-
        (instanceOf(X,vendor:Laptop) AND vendor:processorSpeed(X,Y)).
```

[4] Corresponding generalizations exist for attributes and relationships.

[5] The newly populated ontology would then be $G' := G \cup G_\Delta$.

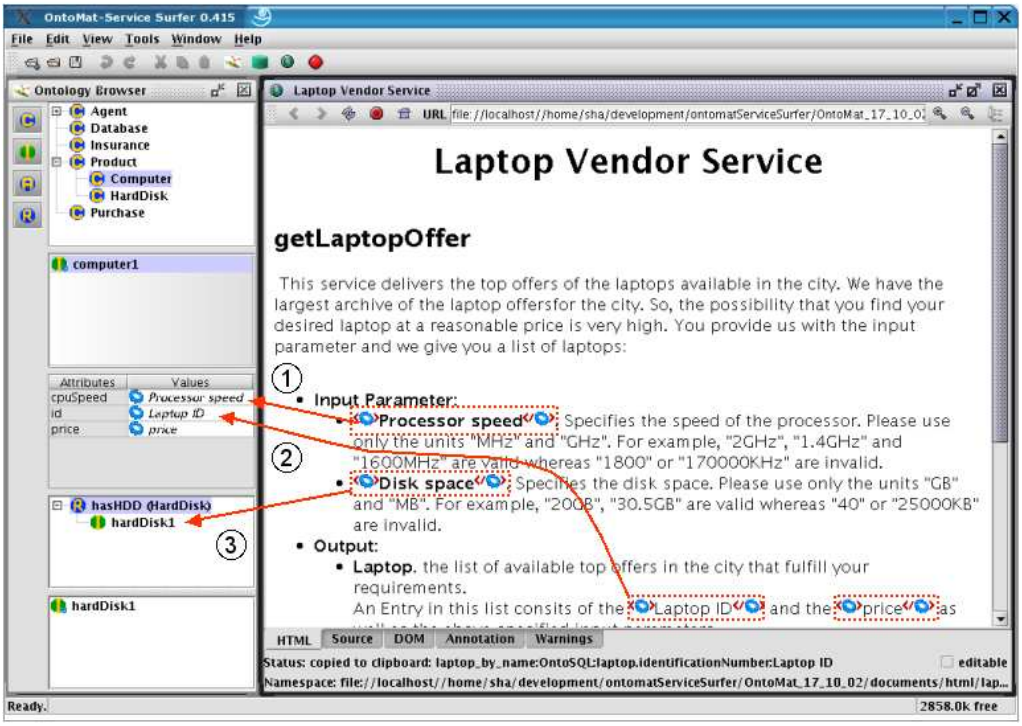

Figure 7.6.: Screenshot of OntoMat-Service-Surfer annotating vendor service

One may trace the later drag'n'drop action in Figure 7.6, where action 1 picks up 'Processor Speed' with its underlying Web service parameter processorSpeed (cf. the markup `elementURI="http://laptop.wsdl/laptop/#part(getLaptop OfferRequest/processorSpeed)"` in Listing 7.5). It is dropped onto the attribute that comes closest in his client ontology, viz. the aforementioned cpuSpeed, and generates the consequences just mentioned. Similarly, the second text item "Disk Space" being annotated with the input parameter `diskSpace` is handled in action 2. This time, however, the annotator must also create a hasHDD relationship between the generic instance hardisk1 and the generic instance of computer1 to build a larger graph representing a mapping rule with two generic attribute values (on cpuSpeed and diskSpace). Finally, the annotator maps the output parameters in action 3 (cf. Figure 7.6).

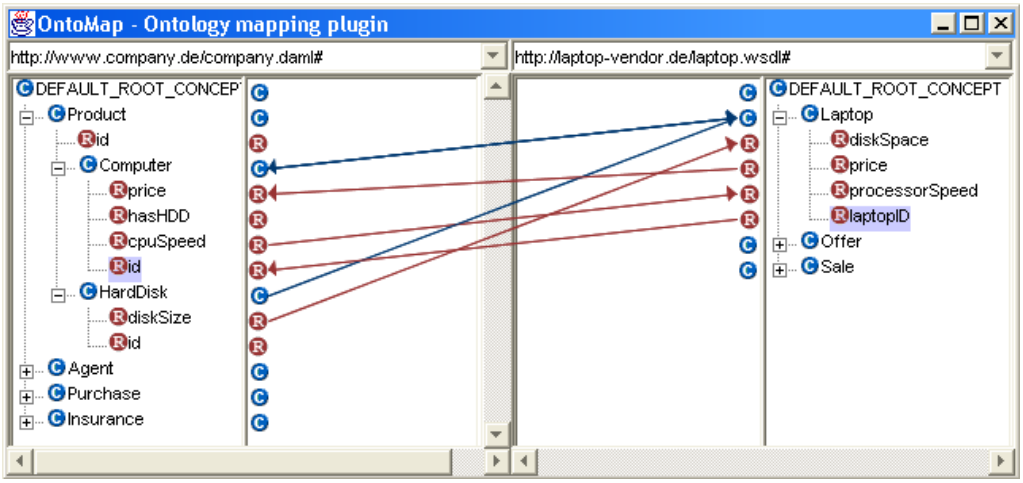

Figure 7.7.: Mapping between Client Ontology (left window) and Vendor Ontology (right window)

Investigating and Modifying Mapping Rules

The results of deep annotation are mapping rules between the client ontology and each service ontology. The annotator may publish the client ontology and the mapping rules derived from annotations. This enables third parties (in particular logical roles that follow in the CREAM-Service process) to execute the services on the basis of the semantics defined in the client ontology.

We use F-Logic to define the mapping rules. F-logic is a deductive, object-oriented database language that combines the declarative semantics and expressiveness of deductive database languages with the rich data modeling capabilities supported by object oriented model [Kifer *et al.*, 1995].[6] However, the annotator does not have to write F-logic rules. They are generated automatically by the OntoMat-Service-Surfer.

Figure 7.7 and Figure 7.8 give the reader an intuition of how such automatically generated mapping rules look like when visualized with the OntoEdit plugins OntoMap (cf., Section 6.7). Figure 7.7 shows the mapping from the company ontology to the vendor ontology which is a result from the annotation effort indicated in Figure 7.6. The result for the corresponding mapping of the insurer's ontology is depicted in Figure 7.8.

[6]Thus in our implementation the aforementioned examplary mapping rule looks slightly different than the depicted first-order logic formulation. Since the first-order presentation is

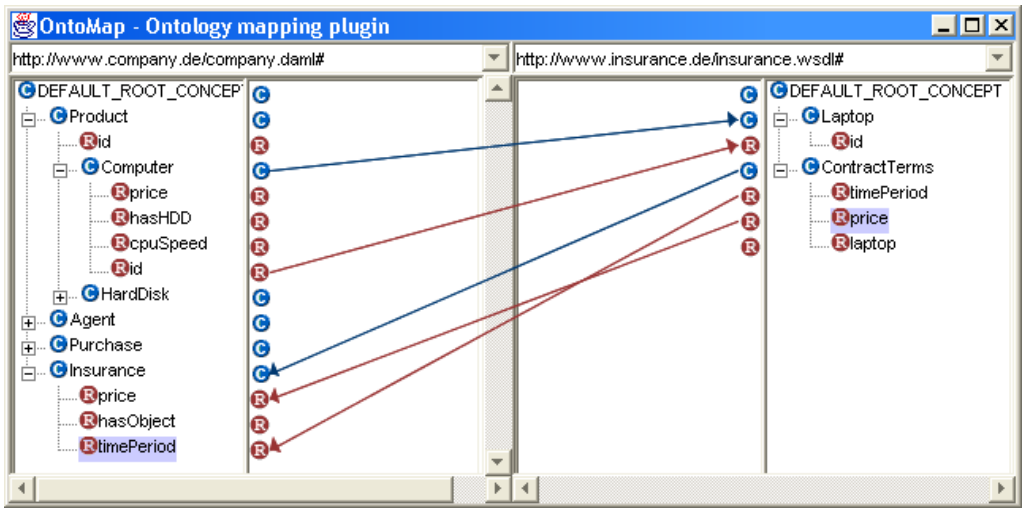

Figure 7.8.: Mapping between Client Ontology (left window) and Insurer's Ontology (right window)

7.2.5. Conclusion

In this Section we have described CREAM-Service (an application of the deep annotation framework presented in Chapter 6) to tie together the World Wide Web and Web services into a Service Web. Germane to CREAM-Service is its blending of browsing the Web, aggregating conceptual descriptions and Web services and then investigating and invoking them from one platform.

We have also presented OntoMat-Service-Surfer, a tool that constitutes a prototype implementation of CREAM-Service. Currently, our prototype understands WSDL with RDF(S) for Web service descriptions, but its flexible architecture allows easy integration of more powerful Web service description languages like DAML-S [Ankolekar *et al.*, 2002].

Clearly, one must be aware of what CREAM-Service and OntoMat-Service-Surfer can do and what they can't do. CREAM-Service is not intended to cater to businesses that want to establish complex Web service connections with intricate interactions. For this objective, the integration by Semantic Annotation may provide a quick, first prototype, but Semantic Annotation cannot provide arbitrary complex mapping rules or arbitrarily complex workflows. On the other hand, CREAM-Service allows exactly for easily building a prototype Web ser-

conceptually close enough, we have decided not to detract the reader by another syntax.

vice integration and it allows for users with domain knowledge (e.g. consultants doing ERP configuration) to participate in the Service Web — without much programming.

CREAM-Service opens up many interesting questions that need to be solved in the future, such as

- how to automate the way that Web Services are presented to the World;
- how to characterize the boundaries of what functionality can be aggregated and executed.
- how to annotate mappings between ontologies (semi-) automatically [Patil *et al.*, 2004].

Eventually, CREAM-Service and OntoMat-Service-Surfer, in conjunction with their counterparts in Semantic Annotation (cf. Chapter 4) and deep annotation (cf. Chapter 6), open up the possibility to bring Web pages, databases and Web Services into one coherent framework and thus progress the Semantic Web to a large Web of data and services.

Part III.

Evaluation

"Designing Annotation Before It's Needed."
— Frank Nack

8. Evaluation of Manual Annotation

This Chapter deals with the evaluation of hand-on-experiences exploring an annotation experiment with human subjects. It describes an empirical evaluation study (Section 8.2 and Section 8.3) of ontology-based Semantic Annotation. Based on a given ontology and a set of documents, we analyze in this chapter inter-annotator-agreement between different humans. The evaluation study uses several standard and two original measures (Section 8.4). The latter take into account a notion of sliding agreement between metadata – exploiting semantic background knowledge provided by the ontology. Section 8.5 introduces a simple sample evaluation and Section 8.6 presents the overall cross evaluation results.

References: This chapter is partly based on [Staab *et al.*, 2001b]

8.1. Introduction

In the previous sections we presented our comprehensive annotation environment. In this section we focus on the human annotators. The *human factor* is easily underestimated, but is extremely critical for the manual creation of metadata. There is a lack of experience in creating semantically interlinked metadata for Web pages. It is not clear how human annotators perform overall and, hence, it is unclear, what can be assumed as a baseline. Though there are corresponding investigations for only indexing documents, e.g. in library science [Leonard, 1977], a corresponding more detailed assignment of interlinked metadata that takes advantage of the structure of RDF is lacking.

In the following we deal with an evaluation of hands-on-experiences, exploring an experiment with human subjects. We describe an empirical evaluation study of ontology-based Semantic Annotation. Based on a given ontology and a set of documents, we have analyzed agreement between different users. Our objective was to find out about inter-annotator agreement and to come up with some measures about what can be expected from Semantic Annotation as an input for machine processing.

8.2. Evaluation Setting

8.2.1. General Setting

We have evaluated our environment for ontology-based Semantic Annotation with human subjects performing Semantic Annotation. In order to determine their inter–annotator agreement, we have undertaken the following experiment: Nine subjects who were undergraduate students in industrial engineering annotated 15 Web pages[1] of our institute as part of fulfilling their requirements in a seminar on the "Semantic Web". The domain expertise of the subjects was very sparse. Some of them had some very minor knowledge about the topics and about semantics from introductory courses in computer science, but no prior knowledge of ontologies and Semantic Annotation. Before doing the actual annotations, subjects received 30 minutes of training. It took about 15 minutes to explain to them the overall goal of semantic annotation and to teach them the basic meaning of the Semantic Web Research Community SWRC ontology[2]. The rest of the time was used to acquaint them with the annotation tool. All in all, we have thus expected that the overall achievements could not score very high compared to an expert annotator.

8.2.2. Semantic Annotation Categories

Individual annotation of the 15 test pages led for each annotator to a set of RDF (see Section 2.4) annotated HTML files. From these files we extracted the corresponding annotations as ground facts. Referring to our definition of annotation in section 3.4.1 we define four semantic categories: i) instance identification, ii) instance-class relationship, iii) instance-attribute relationship and iv) instance-instance relationship. According to these four semantic categories, we distinguished between four different evaluation categories, which are motivated by the varying difficulties they exhibit for the human annotator:

1. The first one only considers the **instance identification**. An annotator may choose to use a string as a identifier to denote a new instance. These instance identifiers play a role similar to that of primary keys in databases. In analogy, we rely on the unique name assumption: Two different identifiers are assumed to denote different instances. In our example, subjects would, e.g., identify an instance with identifier `RudiStuder` based on the identifier proposals given by the tool.

[1] The Web pages are available at http://ontobroker.semanticweb.org/annotation/SemAnn/

[2] The SWRC ontology models the Semantic Web Research Community, its researchers, topics, publications, tools and properties between them. A detailed description of the SWRC ontology and the ontology itself is available at http://ontobroker.semanticweb.org/ontos/swrc.html

2. The second category includes all **instance–class relationships**, such as instance-of(RudiStuder, FULLPROFESSOR) means that RudiStuder belongs to the set of FULLPROFESSORs or — precisely speaking — the string "RudiStuder" is a unique descriptor for an instance of a FULLPROFESSOR.

3. The third one comprises all **instance–attribute relationships**, such as surname-of(RudiStuder, STUDER), which means that the surname-of the entity the identifier of which is RudiStuder is STUDER.

4. The last category is constituted by **instance–instance relationships**. It includes the relations between two distinct instances, such as married-to(RudiStuder, IreneStuder) or heads(RudiStuder, KMResearch Group) with their obvious interpretations.

As we will also see in our evaluation in the following, the first one reaches good values based on the proposals by our tool. Instance–class assignment is very difficult and results in very low inter–annotator agreement. Attributing seems comparatively easy, where recognizing instance–instance relationships appears to be the hardest, as it requires elaborate thinking about the denotation of two distinct instances at a rather abstract level.

8.3. Formal Definition of Evaluation Setting

The comparison and evaluation of ontology-based Semantic Annotation is not this well researched (cf. Section 10 for a detailed comparison of existing work). To the best of our knowledge, no established measure on that we could built did exist. In our semantic annotation scenario we distinguished two different types of measures: On the one hand, we adopt the well-known measures of *precision* and *recall* from the information retrieval community. Whereas these measures denote *perfect agreement*, we additionally define new measures for *sliding agreement*, that take into account string similarity and the sliding scale of the given conceptual structures and compute an inter-annotator accuracy between two annotated document sets.

For the Semantic Annotation scenario we distinguish between the ontology $\mathcal{O}$ (cf. Definition 2.5.1) and the knowledge base KB generated on top (cf. Definition 2.5.3). The ontology acts as the conceptual backbone for generating semantic annotations. In our setting we had a subject S_n that generates annotations for a set of documents. The results produced by each of the subjects are defined as knowledge bases.

Definition 8.3.1 (Semantic Annotation Knowledge Base)
The Semantic Annotation Knowledge Base of subject S_n (KB_n) is a knowledge base as defined in Definition 2.5.3 (page 28). Recall that a knowledge base

is a structure $KB := (C_{KB}, R_{KB}, A_{KB}, I, \iota_C, \iota_R, \iota_A)$. *The knowledge base* KB_n *consists of a set of instances* I_n *that are uniquely identified by their corresponding set of identifiers* $id(I_n)$. *The identifier* $id(i_n)$ *of an instance* i_n *is a string. Each instance* i_n *is assigned to one class* c_n *by the* class assignments AC_n *of subject* S_n. *The instance-attribute relationship are described by the* attribute assignments AA_n *and instance-instance relationships by the* relation assignments AR_n.

Note that in our current setting AC_n has been restricted by the annotation tool to be functional, i.e. every instance could only be assigned to one class.

8.4. Evaluation Measures

8.4.1. Perfect Agreement — Agreement Precision & Agreement Recall

Precision and recall are known from their definition on the document level.

Definition 8.4.1 (Precision, Recall)
Precision is the proportion of retrieved set that is relevant:

$$precision = \frac{relevant \cap retrieved}{retrieved}$$

Recall is proportion of all relevant document in the collection included in the retrieved set:

$$recall = \frac{relevant \cap retrieved}{relevant}$$

We adopted these two measures for computing the degree to which annotators agree on a set of documents with regard to each of the four Semantic Annotation categories. Formally, this agreement is computed from the overlap of the elements of two subjects' Semantic Annotation Knowledge Bases KB_n, KB_m. The general notions of *Agreement Precision* (AP) and *Agreement Recall* (AR) are defined based on a pair of sets Q_n, Q_m:

Definition 8.4.2 (Agreement-Precision, Agreement-Recall)
Agreement-Precision is defined as:

$$P(Q_n, Q_m) := \frac{|Q_n \cap Q_m|}{|Q_n|}$$

and Agreement-Recall defined as:

$$R(Q_n, Q_m) := \frac{|Q_n \cap Q_m|}{|Q_m|}$$

Agreement precision and agreement recall are related of each other, *i.e.* $P(Q_n, Q_m) = R(Q_m, Q_n)$ and $R(Q_n, Q_m) = P(Q_m, Q_n)$. Hence, in the following we will only refer to agreement precision, but we will evaluate agreement precision in "both directions". This means, when cross-evaluating annotation results we will evaluate how precisely subject S_n agrees with subject S_m *and vice versa*. Since agreement recall is related to agreement precision this way will also yield all agreement recall numbers. Now, **agreement precision** for each of the four categories can be simply defined by specifying Q_n and Q_m:

1. **P for Instance Identification**: $Q_n := id(I_n); Q_m := id(I_m)$
2. **P for Class Assignments**: $Q_n := AC_n; Q_m := AC_m$
3. **P for Attribute Assignments**: $Q_n := AA_n; Q_m := AA_m$
4. **P for Relation Assignments**: $Q_n := AR_n; Q_m := AR_m$

These measures gave us first ideas about the human intra-annotator agreement reachable in our evaluation study using the annotation tool However, the problem is that they lack a sense for the sliding scale of adequacy prevalent in our hierarchical structures. This became obvious especially when comparing the set of instance-class relationships AC (cf. Subsection 8.5). To evaluate the quality of this kind of Semantic Annotations, we also wanted to add some bonus to annotations that *almost* fitted an annotation in another ontology and, then, to compare annotation schemes on this basis.

8.4.2. Sliding Agreements

In this section we introduce the measures we used to compute the sliding agreements for instance identification and class assignment to instances.

Sliding Agreement for Instance Identification

In order to compare instances on a string level, one needs a method for comparing and classifying strings that represent the instance identifiers in the ontology. One method for judging the similarity between two strings is the *edit distance* formulated by Levenshtein [Levenshtein, 1966]. This is a similarity measures based on the minimum number of token insertions, deletions, and substitutions required to transform one string in another using a dynamic programming algorithm. For example if we calculate the edit distance between the two instance identifiers `RudiStuder` and `Rudi Studer` we compute an edit distance of $leven(\texttt{RudiStuder}, \texttt{Rudi Studer}) = 1$.

In order to compare two Semantic Annotation knowledge bases KB_n, KB_m on a norm scale of $[0, 1]$ with 1 for perfect match and near zero for bad match according to the levenshtein measure, we introduce the averaged Identifier Matching Accuracy (IdMA) as follows:

$$\overline{\text{IdMA}}(I_n, I_m) = \frac{1}{|I_n|} \sum_{i_k \in I_n} \text{IdMA}(i_k, I_m) \in [0, 1]$$

$$\text{IdMA}(i_k, I_m) = \max_{i_l \in I_m} \frac{1}{1 + leven(id(i_k), id(i_l))}$$

Sliding Agreement for Class Assignments – Relative Inter-Annotator Agreement

Our new evaluation measure should reflect the distance between the annotation of one annotator to annotations of another annotator. The so called upwards cotopy [Maedche and Zacharias, 2002] is the underlying measure to compute the semantic distance in a concept hierarchy.

$$\uparrow c := \{d \in C \mid c \leq_C d\}$$

The semantic characteristics of $\leq_C$ are utilized: The attention is restricted to superconcepts of a given concept c. Based on the definition of the upwards cotopy ($\uparrow c$) the concept match accuracy (CMA) is defined:

$$CMA(c_n, c_m) := \frac{| \uparrow c_n \cap \uparrow c_m |}{| \uparrow c_n \cup \uparrow c_m |}$$

Based on the concept matching accuracy defined above we introduce the **instance matching accuracy** IMA. Given two instance-class relationships (i_n, c_n) and (i_m, c_m) IMA is calculated as:

$$\text{IMA}((i_n, c_n), (i_m, c_m)) := \text{CMA}(c_n, c_m) \in [0, 1]$$

where $i_n, i_m \in I$ and $c_n, c_m \in C$.

The **relative intra-annotator agreement** $\overline{RIAA}$ is based on a weighted IMA. $\overline{RIAA}$ is the averaged accuracy that the instance–class annotations of an annotator match against their best counterparts contained in another semantic annotation knowledge base:

$$\overline{\text{RIAA}}(AC_n, AC_m) = \frac{1}{|AC_n|} \sum_{(i_n,c_n)\in AC_n} \text{RIAA}((i_n, c_n), AC_m)$$

$$\text{RIAA}((i_n, c_n), AC_m) = \max_{(i_m,c_m)\in AC_m} \text{IMA}((i_n, c_n), (i_m, c_m))$$

8.5. Example of an Evaluation

Figure 8.1 depicts an example scenario. In the upper part of the figure, parts of the SWRC ontology are depicted as conceptual backbone for Semantic Annotation. In the left part of the figure some example we see some annotations done by Annotator 1, in the right part we see some given by Annotator 2. They have produced two different Semantic Annotation knowledge bases KB_1 and KB_2, based on the SWRC ontology and the given example Web page.

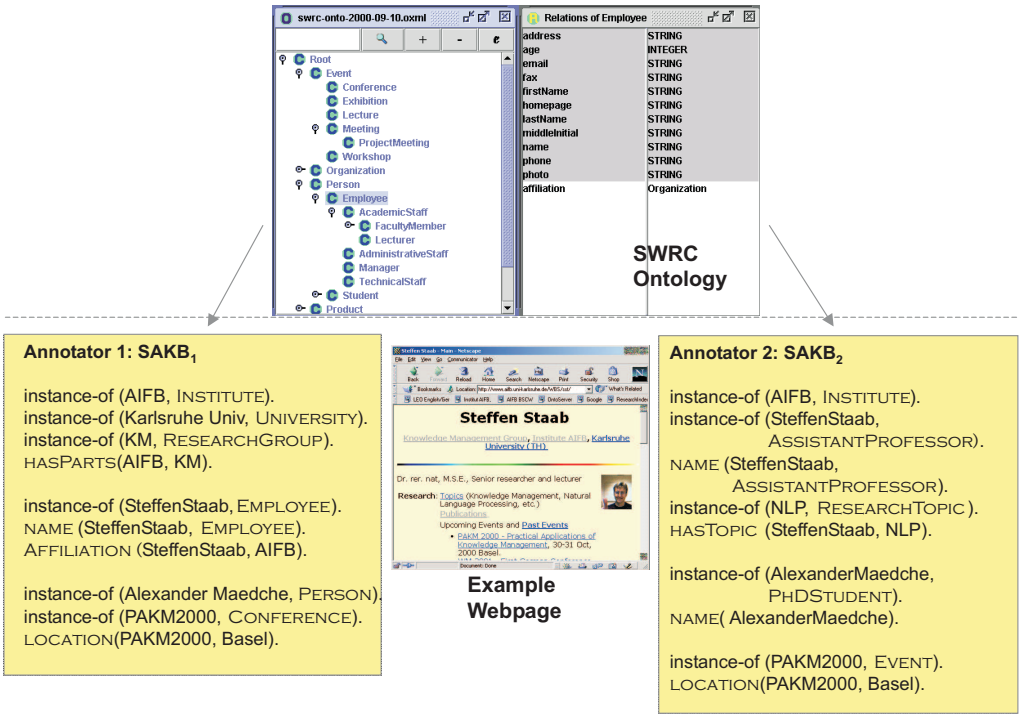

Figure 8.1.: Example Evaluation.

In the example scenario we can see that the instance identifiers `AIFB`, `SteffenStaab` and `PAKM2000` match directly. This results in an agreement-precision for instance identifiers computed as $P(I_1, I_2) = 3/6 = 0.5$. We also

see that the instance identifier `Alexander Maedche` and the instance identifier `AlexanderMaedche` are only similar. Their similarity computes to 0.5, leading to an overall $\overline{IdMA}(I_1, I_2)$ of $3.5/6 = 0.58$. The sliding measure $\overline{IdMA}$ reflects the fact that there are identifiers that match nearly perfectly.

Looking at the instance-class relationships delivers much worse results. One counts only 1 directly matching instance-class relationship, namely instance-of (`AIFB`, INSTI TUTE). So we get an $P(AC_1, AC_2)$ of $1/6 = 0.17$.

The sliding agreement for instance-class relationships recognizes that there are more near hits, namely instance-of(`SteffenStaab`, EMPLOYEE) with instance-of(`Steffen Staab`, ASSISTANTPROFESSOR) and instance-of(`PAKM2000`, CONFERENCE) with instance-of(`PAKM2000`, EVENT). We calculate according to the measure defined above $\overline{RIAA}$ of 0.34. Additionally we count 0 matching instance-attribute relationships and 0 matching instance-instance relationships, viz. we obtain an $P(AA_1, AA_2)$ of 0 and an $P(AR_1, AR_2)$ of 0 respectively.

8.6. Cross-Evaluation Results

As already mentioned our evaluation is based on the following input parameters: We selected 15 Web documents describing actual persons, events, research projects and organizations from our institute. The ontology given to the subjects was the SWRC vocabulary in its current version 2000-10-09 containing 55 classes and 157 attributes and relations. The annotations have been stored in RDF on the Web pages. We extracted the annotations from these Web pages as ground facts. Our cross-evaluation scenario can be divided into three parts. First, we present some basic statistics we calculated from the Semantic Annotation knowledge bases. Subsequently, the measures computed using agreement-precision and agreement-recall are explained and interpreted. Additionally, we use $\overline{IdMA}$ and $\overline{RIAA}$ to compute the sliding agreement.

8.6.1. Basic statistics

Table 8.1 shows the basic statistics we obtained in the two phases by counting each semantic annotation category of the Semantic Annotation knowledge bases. KB_0 is the semantic knowledge annotation base that has been generated by an expert annotator. It will serve as the gold standard in our evaluation framework.

One may see that the instance identifiers with their corresponding instance–class relationships have an average of 106 elements, with a low standard deviation of 25 elements. Standard deviation of instance–attribute and instance–instance relationships results in a higher value with approx. 50 elements. Some of our students (KB_5, KB_8) have outperformed the gold standard with respect to the basic

Subject	$\|I_n\|$ and $\|AC_n\|$	$\|AA_n\|$	$\|AR_n\|$
KB_0 (Gold)	123	230	197
KB_1 (anso)	110	203	108
KB_2 (eryi)	115	162	132
KB_3 (hela)	81	157	17
KB_4 (makr)	71	121	32
KB_5 (mama)	152	292	175
KB_6 (mari)	97	150	68
KB_7 (midu)	80	137	60
KB_8 (stse)	126	226	87
KB_9 (taso)	104	173	129
mean	106	186	101
standard deviation	25	53	59

Table 8.1.: Basic statistics computed for the generated Semantic Annotation knowledge bases

statistics. In the following we will see what agreement measures are computed based on these 10 given Semantic Annotation knowledge bases.

8.6.2. Perfect Agreement: Agreement-Precision & Agreement-Recall

The Tables 8.2, 8.3, 8.4, 8.5 lists all measures of perfect agreement that we computed in our semantic evaluation study. As highest value for agreement-precision of instance identification (8.2) we obtained 0.8, by comparing the semantic annotation knowledge bases of subject 7 with subject 2. This high value could be obtained by the tool strategy for generating and proposing instance identifiers to the users. Agreement-precision of instance–class relationships (8.3) scores much worse, the highest value has been reached with comparing subject 4 with subject 2, namely 0.46. Analyzing Agreement-precision of instance–attribute relationships (8.4) resulted in a maximum reachable value of 0.6 by comparing subject 2 with subject 5. The best value for the Agreement-precision of instance–instance relationships (8.5) was 0.29, achieved by comparing subject 3 with subject 2.

Figure 8.2 shows agreement-recall vs. agreement-precision diagrams for matching instance identifiers (id(i)), matching instance–class relationships (AC), matching

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0.44	0.49	0.21	0.24	0.48	0.24	0.25	0.32	0.41
1	0.49	1	0.73	0.46	0.37	0.68	0.43	0.57	0.45	0.71
2	0.52	0.7	1	0.43	0.49	0.78	0.46	0.56	0.5	0.66
3	0.32	0.63	0.6	1	0.33	0.51	0.44	0.42	0.38	0.58
4	0.42	0.58	0.79	0.38	1	0.73	0.54	0.51	0.48	0.59
5	0.39	0.49	0.59	0.27	0.34	1	0.32	0.39	0.44	0.49
6	0.31	0.48	0.55	0.37	0.39	0.49	1	0.39	0.36	0.46
7	0.39	0.79	0.8	0.43	0.45	0.75	0.48	1	0.46	0.76
8	0.31	0.39	0.45	0.25	0.27	0.53	0.28	0.29	1	0.36
9	0.49	0.75	0.73	0.45	0.4	0.72	0.43	0.59	0.43	1

Table 8.2.: Perfect Agreement with P computed for $id(I)$

instance–attribute relationships (AA) and matching instance–instance relationships (AR). Each point in the diagram represents one comparison. We can see from the diagrams that the values obtained for instance identifier agreements range between 0.21 and 0.8. The comparison for instance–class relationships results range between 0.1 and 0.46. Instance–attribute relationships score between 0.09 and 0.6. Instance–instance relationships score very badly, ranging between 0 and 0,29 with an average of 0,07.

8.6.3. Sliding Agreement

We also computed the sliding agreement measures defined above. Table 8.6 contains $\overline{IdMA}(I_n, I_m)$ and Table 8.7 contains $\overline{RIAA}(AC_n, AC_m)$ computed for two Semantic Annotation knowledge bases, respectively. The values obtained for the identifier matching accuracy did not outperform the values we obtained by computing agreement-precision for identifiers. The average value improved only from 0.49 to 0.54. The largest value we received was 0.82 by comparing the knowledge bases of subject 4 with subject 2.

The values obtained for the relative-inter annotator agreement scored obviously better than the corresponding agreement-precision values for instance–class relationships. Here, the average improved from 0.26 to 0.37. The best value of

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0.37	0.37	0.2	0.17	0.22	0.16	0.16	0.14	0.18
1	0.41	1	0.45	0.28	0.19	0.34	0.25	0.29	0.15	0.32
2	0.39	0.43	1	0.27	0.29	0.28	0.23	0.31	0.2	0.26
3	0.3	0.38	0.38	1	0.23	0.19	0.27	0.25	0.16	0.25
4	0.3	0.3	0.46	0.27	1	0.42	0.31	0.25	0.25	0.42
5	0.18	0.24	0.21	0.1	0.2	1	0.15	0.19	0.14	0.22
6	0.21	0.28	0.28	0.23	0.23	0.24	1	0.29	0.15	0.25
7	0.25	0.4	0.45	0.25	0.23	0.36	0.35	1	0.15	0.31
8	0.13	0.13	0.18	0.1	0.14	0.17	0.12	0.1	1	0.16
9	0.21	0.34	0.29	0.19	0.29	0.33	0.23	0.24	0.19	1

Table 8.3.: Perfect Agreement with P computed for AC

0.65 was reached by the comparison of Semantic Annotation knowledge bases generated by subject 7 and subject 2.

Figure 8.3 depicts the obtained results graphically. On the left side of Figure 8.3 the resulting comparison values for the identifier matching accuracy are shown. $\overline{IdMA}$ ranges between 0.32 and 0.82 with an average of 0.54. As shown in the right part of Figure 8.3 the results obtained for computing $\overline{RIAA}$ range between 0.19 and 0.65 with an average of 0.37.

8.6.4. Overall results

Due to circumstances in our setting, like lack of domain knowledge and no prior experience with the ontologies or with the tool, we believe that this result ranks among the baseline worst cases that will be found in typical Semantic Annotation settings. Our conjecture is that further training may considerably improve inter-annotator agreement — though we do not expect any numbers for agreement-precision and agreement-recall that range in the vicinity of 100%.

Our evaluation case study goes ahead with several limitations that became obvious during the annotation experiments. Firstly, the sequence of Web pages was given. This may lead to some unwanted similarities of perception. Secondly, the subjects were non-experts but a large part of the domain was about common

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0.18	0.19	0.12	0.09	0.2	0.11	0.11	0.17	0.18
1	0.21	1	0.46	0.33	0.25	0.5	0.31	0.37	0.33	0.5
2	0.27	0.57	1	0.41	0.41	0.6	0.44	0.49	0.43	0.54
3	0.17	0.43	0.42	1	0.26	0.39	0.34	0.25	0.29	0.38
4	0.17	0.42	0.55	0.34	1	0.5	0.42	0.4	0.32	0.48
5	0.15	0.35	0.33	0.21	0.21	1	0.22	0.27	0.27	0.32
6	0.17	0.42	0.48	0.35	0.34	0.42	1	0.33	0.36	0.39
7	0.19	0.55	0.58	0.29	0.35	0.57	0.36	1	0.38	0.5
8	0.17	0.29	0.31	0.2	0.17	0.35	0.24	0.23	1	0.23
9	0.24	0.58	0.51	0.34	0.34	0.53	0.34	0.4	0.3	1

Table 8.4.: Perfect Agreement with P computed for AA

knowledge. This fact may lead to better results than in more specific domains without common knowledge.

8.7. Conclusion

This section presents an evaluation of the creation of metadata by annotating Web pages. Starting from our ontology-based annotation environment, we have collected experiences in an actual evaluation study. The results provide a baseline that one may consider for further research about automatic annotation tools. The evaluation study we have described was performed using several standard and two original measures. The latter take into account a notion of sliding agreement between metadata – exploiting semantic background knowledge provided by the ontology.

Future work will have a start on current studies that have looked at the feasibility of automatic building of knowledge bases from the Web. In our future work, we want to integrate such methods into an even more comprehensive annotation environment – including, e.g. the learning of ontologies from Web documents and (semi-)automatic ontology-based Semantic Annotation. The general task of knowledge maintenance, including evolving ontologies and Semantic Annotation knowledge bases, remains a topic for much further research in the near future.

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0	0.05	0	0	0.05	0.01	0	0	0
1	0	1	0.13	0.03	0.01	0.21	0.09	0.09	0.02	0.23
2	0.07	0.11	1	0.04	0.02	0.22	0.08	0.11	0.02	0.09
3	0	0.18	0.29	1	0	0	0	0	0	0.12
4	0	0.03	0.06	0	1	0.03	0.06	0.06	0.19	0.28
5	0.06	0.13	0.17	0	0.01	1	0.07	0.07	0.02	0.11
6	0.01	0.15	0.15	0	0.03	0.18	1	0.06	0.03	0.13
7	0	0.17	0.25	0	0.03	0.22	0.07	1	0.05	0.05
8	0	0.02	0.02	0	0.07	0.05	0.02	0.03	1	0.02
9	0	0.19	0.09	0.02	0.07	0.16	0.07	0.02	0.02	1

Table 8.5.: Perfect Agreement with P computed for AR

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0.52	0.58	0.32	0.34	0.57	0.34	0.35	0.44	0.5
1	0.54	1	0.76	0.51	0.43	0.71	0.48	0.62	0.5	0.74
2	0.58	0.73	1	0.48	0.53	0.8	0.52	0.6	0.55	0.7
3	0.4	0.66	0.64	1	0.39	0.55	0.5	0.47	0.44	0.61
4	0.5	0.63	0.82	0.44	1	0.77	0.58	0.56	0.54	0.64
5	0.45	0.54	0.62	0.33	0.39	1	0.37	0.45	0.49	0.54
6	0.39	0.54	0.6	0.44	0.45	0.55	1	0.45	0.43	0.52
7	0.46	0.82	0.82	0.47	0.5	0.78	0.52	1	0.52	0.79
8	0.4	0.45	0.52	0.32	0.35	0.59	0.35	0.37	1	0.43
9	0.55	0.78	0.76	0.5	0.46	0.75	0.49	0.63	0.49	1

Table 8.6.: Sliding Agreement Measures: $\overline{IdMA}$

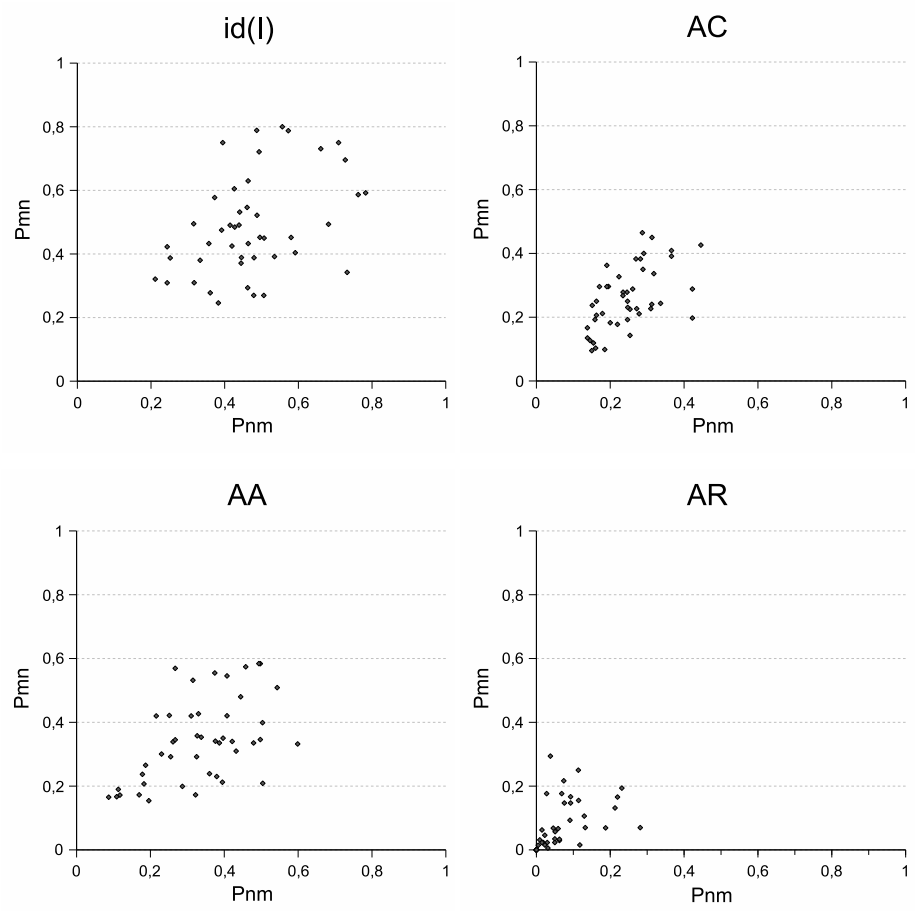

Figure 8.2.: Perfect Agreement

	Subject									
Subj.	0	1	2	3	4	5	6	7	8	9
0	1	0.41	0.42	0.2	0.19	0.34	0.21	0.2	0.24	0.31
1	0.46	1	0.59	0.36	0.27	0.49	0.34	0.44	0.29	0.51
2	0.45	0.56	1	0.35	0.38	0.53	0.36	0.45	0.37	0.46
3	0.31	0.49	0.5	1	0.28	0.31	0.35	0.33	0.28	0.41
4	0.34	0.42	0.61	0.32	1	0.57	0.4	0.35	0.39	0.52
5	0.28	0.35	0.4	0.17	0.26	1	0.22	0.27	0.31	0.35
6	0.26	0.39	0.43	0.29	0.3	0.35	1	0.33	0.26	0.35
7	0.3	0.6	0.65	0.34	0.31	0.52	0.4	1	0.32	0.51
8	0.24	0.26	0.34	0.18	0.22	0.37	0.2	0.2	1	0.26
9	0.36	0.54	0.51	0.32	0.35	0.52	0.32	0.4	0.31	1

Table 8.7.: Sliding Agreement Measures: $\overline{RIAA}$

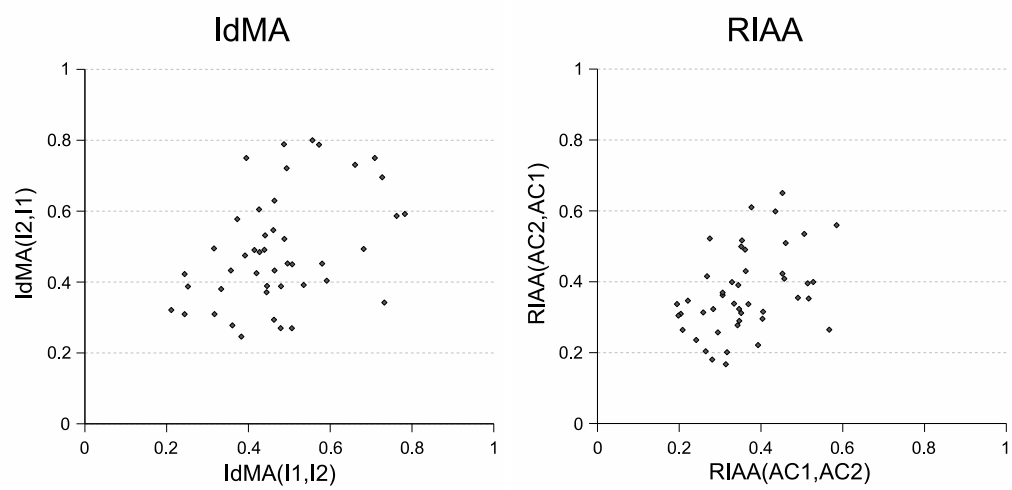

Figure 8.3.: Sliding Agreement

9. Evaluation of Semi-Automatic Annotation

This Chapter investigates the approach of semi-automatic annotation. Based on the evaluation study for manual annotation introduced in the previous Chapter, we adapt the measures to evaluate the similarity between the annotation produced by different users applying the process of manual annotation in comparison to applying the semi-automatic annotation process. Hence, the evaluation confirms that semi-automatic annotation is faster and produces more homogeneous metadata than manual annotation. The chapter describes the principal methods of evaluation (Section 9.2) and presents the results (Section 9.3) and a discussion (Section 9.4).

9.1. Introduction

In Section 5.1 we presented S-CREAM, an extension of CREAM that integrates a learnable information extraction component. S-CREAM allows semi-automatic annotation of documents. After a training phase, S-CREAM processes a document and identifies instances of concepts, of attributes, and of relations. These instances are proposed to the user for annotation. Subsequently, the user supplements the annotation manually with the help of the tool.

Our assumption is that the S-CREAM approach reduces the time and the workload for the annotator in comparison to purely manual annotation. In order to test this we present in this chapter an evaluation of the semi-automatic annotation approach.

Based on the evaluation study for manual annotation introduced in Chapter 8, we adapt the metrics to our extended setting here. Hence, we evaluate the similarity between the annotations produced by different users applying the process of manual annotation in comparison to applying the semi-automatic annotation process.

We conducted an evaluation with 16 participants. The participants were divided into two groups: group A and group B. Group A undertook first a manual annotation then a semi-automatic annotation. The sequence of annotation was reversed for group B.

The results of the evaluation are promising. The time for annotation decreases for group A by about 33% and the inter-annotator agreement increases for both groups by between 14,57% and 96,33%. Hence, the evaluation proved that our hypothesis was correct and that semi-automatic annotation is faster and produces more homogeneous metadata than manual annotation.

The structure of this chapter is as follows: Section 9.2 describes the principal method of the evaluation. In Section 9.3 we present the results of our evaluation. Subsequently, we discuss the results and conclude.

9.2. Method

9.2.1. General Setting

The general rule for random samples is: the smaller the sample is, the less information it gives about the basic population. In general, one should not consider less than 10 to 15 human subjects (cf. [Clauß and Ebner, 1995]). Therefore we choose 16 human subjects for the evaluation. We decided on an even number in order to divide them into two equal groups. The subjects were students without any prior knowledge of ontologies and Semantic Annotation. The participants were between 21 and 31 years of age, eight of them were female.

Each participant annotated ten Web pages, five using manual and five using semi-automatic annotation. For the purpose of training the subjects we choose two additional pages. The twelve pages where randomly chosen from the hotel Web site[1], considering the following criteria:

- To ensure a variety for the training of the information extraction system, each city, where a hotel can be located, occurs not more than once.
- To reduce the variance between the document types, we only consider Web pages about hotels, no boarding houses, summer cottages, or villas.
- The Web page should have information about the hotel room prices. The Web page should not be too short or too simple and should not consist only of pure text. Hotel room prices are usually presented in a structured form, e.g. a table. A mixture between text and structural information is typical for documents on the Web and should therefore be presented in the sample.
- Finally, we ensured that the annotation tool is capable of a proper and fast rendering of the chosen Web page.

[1] http://www.all-in-all.de

9.2.2. The Domain Ontology

We choose the GETESS ontology as a basis for the evaluation. GETESS is an extensive tourism ontology with 1042 concepts, 162 relations and an average depth of 5.7. The ontology was developed at the institute AIFB for the GETESS projects (German Text Exploitation and Search System). Based on this we developed a smaller, simpler and easier to use ontology for the experiment. We applied the following steps: i) clarification of the goals and ii) development of the ontology.

Clarification of the Goals. Noy and McGuinness [Noy and McGuinness, 2001] suggest starting the development of an ontology by defining its domain and scope. That is, several basic questions need to be answered:

1. For what we are going to use the ontology? The ontology should be used in the evaluation by students with a very sparse knowledge about annotation. The ontology should be small and simple to ensure a good overview. On the other hand, it should contain all concepts and relations to ensure a reasonable annotation of the Web pages. An annotation expert pruned the ontology, after a sample annotation of some pages. He reduced the ontology to $\frac{1}{5}$, i.e. about 200 concepts with an average depth of 3.5.
2. What is the domain that the ontology will cover? We chose hotel Web pages, in particular Web pages with information about hotel, rooms, configuration, price, address, and the features of the hotel.
3. For what types of questions should the information in the ontology provide answers? In our case useful questions are:
   - In which hotel can I stay overnight for less than 30 Euro?
   - Which hotels are in the city of Dresden?
   - How many beds are in the hotel "Seelust" in Goldberg?
   - Which hotels in Dresden have a conference room for 50 persons?

Development of the Ontology. The goal is to reuse the GETESS ontology as much as possible. The approach is to simplify the ontology: deletion of unnecessary concepts, adding missing concepts and relations, and a reorganization of the hierarchy of the ontology.

1. Simplify the ontology: We reorganized the ontology in such a way that each concept belongs to only one superconcept, because single inheritance is easier to understand for the inexperienced annotator. Furthermore, we

looked for classes that have only one direct subclass, which occurs often in the GETESS ontology and is an indicator of an modeling problem or an incomplete ontology. Each single subclass has been put into the same hierarchy as it former superclass and unnecessary subclasses have been deleted. Concept and relations not necessary for the annotation task have been deleted.

2. Adding of new concepts and relations: The GETESS ontology is a general tourism ontology. It considers virtually all essential concepts. However, some relationships were missing to model the information about the hotel in a proper way. It was possible to replace some deleted concepts by relations. For example the subconcepts "cable TV" and "color TV" of class "TV" have been deleted, but the concept "TV" got the boolean attributes "is_color_TV" and "is_cable_TV". Thus, we were able to reduce the amount of concepts and the depth of the ontology.

3. Testing the ontology: The creation of an ontology is an iterative process. To support this, annotation is a very suitable to test the ontology, i.e. to see if concepts or relationships in the ontology are missing.

The pruned GETESS ontology used for the evaluation, called GETESS_EVS Ontology consists of 187 concept, 150 relations and an average depth of 2,83.

9.2.3. Training of the Annotation

As pointed out in Sections 5.1 and 5.1.3 there is the problem of the input and output of Amilcare, the information extraction component. The S-CREAM framework is ontology based and Amilcare expected flat tags. To ease the task of the mapping from the flat tags to the ontological concepts and attributes we used a labeling scheme for the tags to train Amilcare. The tags for the instances start with the prefix "i_" and the tags for the attributes with the prefix "a_". This helps S-CREAM to distinguish between instances and attributes. For example, <i_hotel> and <i_lounge> leads to the creation of a new instance Hotel and a new instance of Lounge. The tag <a_hotel_name> stands for the attribute name of an instance of the concept Hotel, and <a_address_city> is the value of the attribute city for the instance of address. S-CREAM uses a rule-based discourse model (cf., Section 5.1.4) to create the instances, attributes, and relationships. For example, the discourse model states that an instance of Room following an instance of Hotel is connected by the relationship has_room to the Hotel instance. The training corpus, the Web pages for the training of Amilcare have been annotated after the development of the Ontology according to the above tag labeling scheme. For the training we used eleven Web pages. The selection of samples followed the same rationale as the selection for the user annotation. The training corpus, the test corpus and the evaluation corpus are disjoint.

9.2.4. Experimental Design

The goal is to observe whether the annotation process can be improved and accelerated by the application of information extraction. Hence, we need two random samples to compare. One possibility is that two independent groups work with the tool in the two annotation modes. In such a case, one can observe bigger differences which have greater relation to differences of the annotation [Siegel, 2001]. The human factor is very important for the annotation. Because it is nearly impossible to keep the motivation and the intelligence of two person on the same level, it is advisable to take the same people for both test runs of the annotation modes. To avoid a possible learn effect, the whole group is randomly divided into two groups. The first group, called group A, has eight members and annotated at first in manual mode and later in semi-automatic mode. The second group, called group B, also has eight members and uses the modes in the reverse order. This is called "two dependent samples" according to [Siegel, 2001; Clauß and Ebner, 1995]. On each annotation cycle there are five pages annotated, i.e. there are five pages manually annotated and five pages semi-automaticly annotated. For each annotation mode there is a list of Web pages[2]. Each list contains six pages, because the first page is for the training of the annotators. The annotation of the first page is not considered in the evaluation.

9.2.5. Application of a statistical test

The formulation of the null hypothesis is the first step in the decision process and it is usually formulated to be rejected. The null hypothesis states that there is no difference between the samples. If the null hypothesis is rejected then the alternative hypothesis H_1 can be accepted. In our case H_1 states: *Semi-automatic annotation using information extraction is faster and more homogeneous than manual annotation.* In contrast H_0 states: *There is no difference between manual and semi-automatic annotation.*

In our evaluation we compare two dependent samples, i.e. they have been created by the same person at different points in time in different annotation modes.

9.2.6. Test procedure

Before doing the actual annotations, subjects received 30 minutes of training. They were given a presentation about annotation and metadata. Group B, starting with semi-automatic annotation, were also given an introduction to information extraction in the first annotation session. Group A was given this introduc-

[2]For manual annotation: http://annotation.semanticweb.org/evaluation/july2002/Cream.html. For semi-automatic annotation: http://annotation.semanticweb.org/evaluation/july2002/Scream.html

tion in the second annotation session. In the test phase the groups learned to work with the annotation tool and to use the support of the trained information extraction component.

The subjects annotated one hotel Web page as a warm up task. This took about 20 to 30 minutes. The goals were to give them a better understanding of the annotation task, to allow the possibility for questions and to become familiar with the ontology.

After the warm up each subject annotated five Web pages. Each subject was given the task of annotating only facts about the hotel, address, rooms, room inventory and hotel facilities. In principle annotation is not a restricted task and to annotate everything that suits to the ontology is not a negative outcome. However, we restricted the annotation task to ensure that the annotators accomplished the task on time[3].

The tool recorded the annotation time for each Web page. We asked the subject not to pause on a hotel page, but on the overview page, to get a correct time measurement.

After the evaluation the annotators filled out a questionnaire, in which they could describe their impression about the tool, problems, suggestions for improvement and additional remarks.

9.3. Results

In this section we describe the results of the evaluation. As mentioned before the 16 human subjects where divided in group A and group B. An annotation expert did the same annotation task to provide a gold standard for the evaluation. Both groups are compared to the gold standard.

The results of the evaluation consists of the following parts: i) a time measurement, ii) a basic statistic about the annotation, iii) the calculation of the *perfect-inter-annotator agreement* and the *sliding agreement* and finally iv) the statistical test to verify the significance.

The member of both groups worked under different conditions. Therefore the presentation of the results is subdivided into the groups to get better insights.

9.3.1. Time Measurement

The annotation tool has recorded for both annotation modes the annotation time for each Web page in the test sample. The time was added for each participant

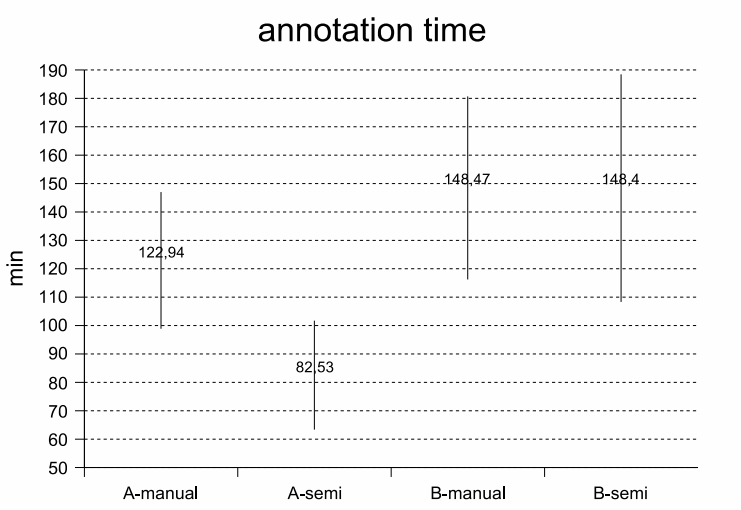

Figure 9.1.: Mean values divided into annotation modes and groups.

to a total for all five Web pages. A grouping of the values is shown in Figure 9.1. The values are presented in minutes. The mean values are shown in Table 9.1. The exact values for each participant can be found in the Appendix.

- For Group A, the annotation time is reduced by the application of semi-automatic annotation by about 33%, from 122.94 minutes to 82.54 minutes. We have two interpretations for this:

  1. Semi-automatic annotation does indeed make the annotation process faster.
  2. Usually the first pages are more slowly annotated because the inexperienced human subject is in the learning phase.

- For Group B, there is nearly no difference: 148.7 minutes for manual annotation and 148.4 minutes for semi-automatic annotation. On initial assessment, this is not what we had expected. However, we interpret this as a compensation for the learn phase of the unexperienced user accompanied by a possible time-gain of the semi-automatic annotation.

- Apparently, group A was faster than group B. Also, group B shows a greater variance of results. The question, whether group A was more gifted or

[3]The complete results of the annotation can be found at:
http://annotation.semanticweb.org/evaluation/july2002/Evaluation.html.

whether group B was more hard-working on the annotation, can be answered with the help of the basic statistic relating to the different groups (Tables 9.3 and 9.4). However, these findings are of small impact for the evaluation due to the depended samples. We are only interested in the difference between the annotation modes.

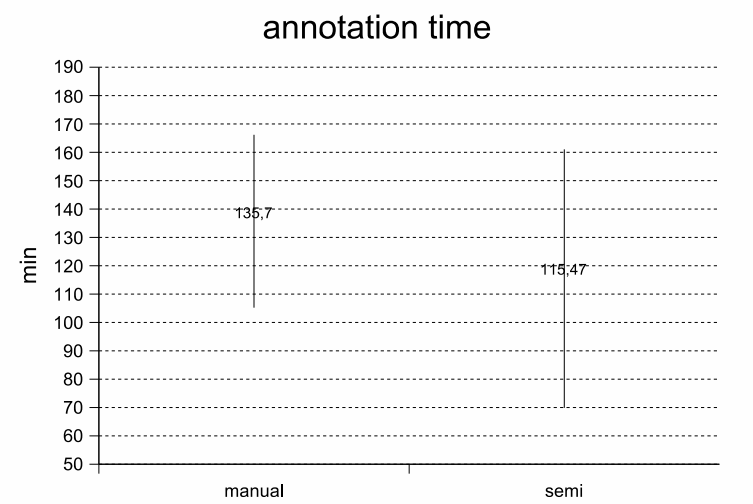

Figure 9.2.: Mean values grouped by annotation modes

Figure 9.2 shows an overall comparison of the annotation modes. It shows clearly that the annotation process is faster in the semi-automatic mode. Because the complete data set is used, all learn effects and idiosyncrasies of the human subjects have been balanced. The average time of the manual annotation mode is 135.70 minutes and of the semi-automatic mode is 115.47 minutes, which results in an average improvement of 15%. Considering the gold standard average time is improving also by about 15% from 116.51 to 99.29 minutes.

	manual	semi	mean
group A	122.94	82.54	102.74
group B	148.47	148.40	148.44
gold standard	58.93	50.75	54.84
mean value (without gold standard)	135.70	115.47	125.59
mean value (gold standard)	116.51	99.29	107.90

Table 9.1.: Average annotation times.

The difference between the two groups and the gold standard is quite considerable

big: on average the gold standard is 2.3 times faster than the other participants. This is because the gold standard has developed the ontology and was familiar with the Web pages. In addition, he followed the annotation instructions more strictly. Other participants annotated more information on the Web pages than was necessary. The disparity is shown by the basic statistics.

It is difficult to specify a general duration time for the annotation of a Web page. There are a number of factors to be considered: size of the Web pages and the structure, domain, complexity of the ontology. However, in the assigned evaluation task, an annotation expert (gold standard) needed about 60 minutes for manual annotation of five web pages and he achieved a time saving of 17% with the semi-automatic annotation. A layman needs after a short introduction in average 135 minutes and got a time saving of 15%.

1. variable	2. variable	t	$t_{\alpha;n-1}$	PR	n	significance
manual	semi	2.93	1.74	0.01	17	yes
A, manual	A, semi	8.66	1.86	0	9	yes
B, manual	B, semi	0.11	1.86	0.91		no

Table 9.2.: Results of the t-test for the annotation time.

We are using the t-test to prove the significance of the evaluation. Table 9.2 gives a summary for the annotation time of all tests. The symbols used have the following meaning:

- Variable 1 and variable 2 are compared by the test.
- The test value is t, it is compared with the tabular value of $t_{\alpha;n-1}$.
- The PR value is the probability of the coincidence of the measured pair difference. For example a PR of 0.01 means a significance of 99%.
- The number of valid pairs is n.
- The variable name consists of the group and the annotation mode. The first row of the table gives the summery of both groups.

The criteria is:
If $|t| < t_{\alpha;n-1}$, then H_0 is accepted.
If $|t| \geq t_{\alpha;n-1}$, then H_0 is rejected.

The results of the test confirm our assumption, i.e. the results of group A and the total results are significant. With respect to the annotation time we can accept our hypothesis, viz. *semi-automatic annotation is faster than manual annotation.* In the following we will consider the homogeneity of the results.

	manual				semi		
user	$\mid I_n \mid, \mid AC_n \mid$	$\mid AA_n \mid$	$\mid AR_n \mid$		$\mid I_n \mid, \mid AC_n \mid$	$\mid AA_n \mid$	$\mid AR_n \mid$
0	55	62	40		47	48	33
1A	54	29	42		56	36	48
2A	42	69	15		54	38	38
3A	48	70	30		45	66	15
4A	68	22	27		51	22	35
5A	60	89	32		54	34	34
6A	48	52	36		50	56	42
7A	51	89	0		46	46	2
8A	64	59	47		50	49	51
total	490	541	269		453	395	298
mean	54.44	60.11	29.88		50.33	43.88	33.11
standard deviation	8.33	23.28	14.6		3.84	13.02	15.59

Table 9.3.: Basic statistic for group A manual and semi-automatic annotation.

9.3.2. Results of Group A

Basic Statistic. Table 9.3 shows the basic statistics of group A for both annotation modes. The user with number 0 represents the gold standard. The results are comparable taking into account that the same user has not annotated the same Web pages using both annotation modes – because he would already known the annotations – but rather similar pages. The average number of instances is 54.44 with a deviation of 8.33. For semi-automatic annotation it is 50.33 instances with a deviation of 3.84. Some annotators produced more instances than the gold standard. User 7A did not created instance-instance relationships in manual mode, and only two in semi-automatic mode.

Perfect-Interannotator Agreement Figure 9.3 shows the results for the instance identification of group A.

- The values are rather low for manual annotation, most of them are below 0.4. The variance is rather high with values between 0.8 and 0.52.

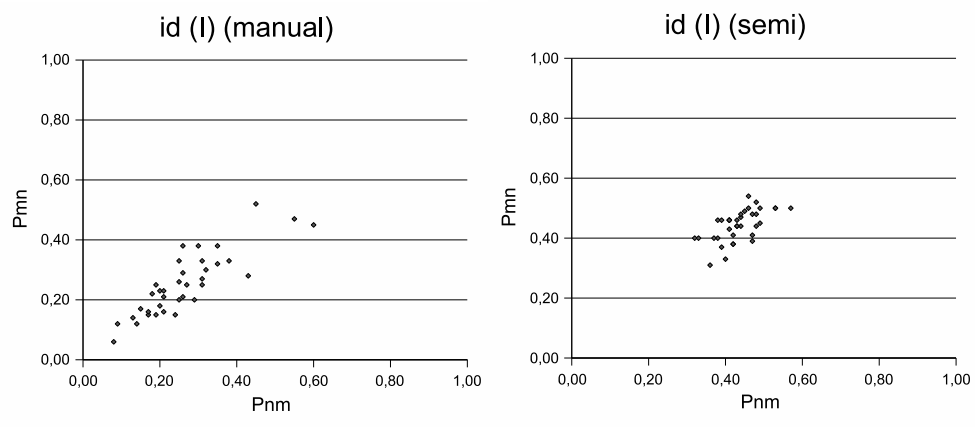

Figure 9.3.: Instance identification of group A

- Semi-automatic annotation results in clearly higher results, most of the values are above 0.4. The variance is lower with values between 0.31 and 0.57.

„

The graphical results for the instance-concept relationships of group A are shown in Figure 9.4

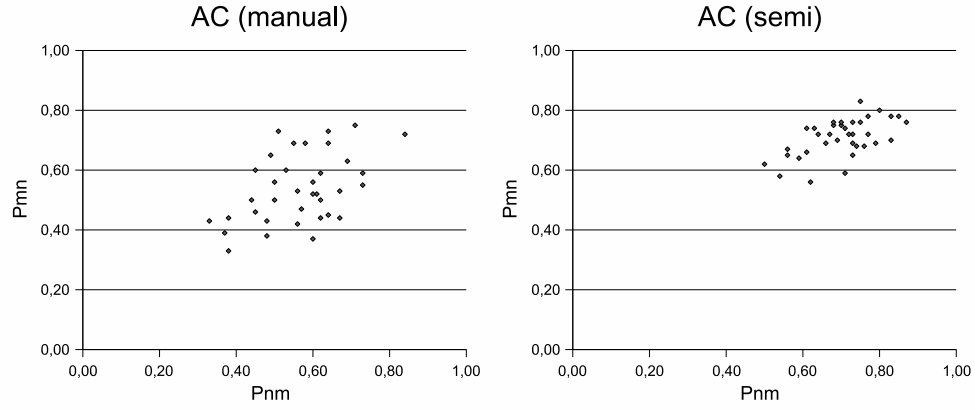

Figure 9.4.: Instance-concept relationship of group A

- The values for manual annotation are rather high with a median variance. The values are between 0.33 and 0.84.

- The values for semi-automatic annotation are also high, the variance is low. The minimum and maximum are 0.5 and 0.85.

The results for the instance-attribute relationship are shown in Figure 9.5.

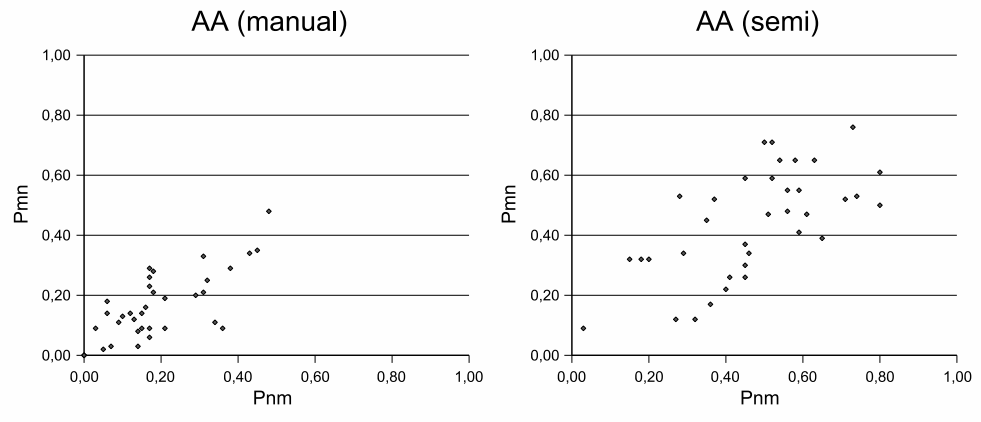

Figure 9.5.: Instance-attribute relationship of group A

- The values for manual annotation are low, ranging from 0 to 0.48 and they have a slight variance.
- The values for semi-automatic annotation are higher, ranging from 0.03 to 0.8 with a high variance.

The results are shown in Figure 9.6 for the instance-instance relationship.

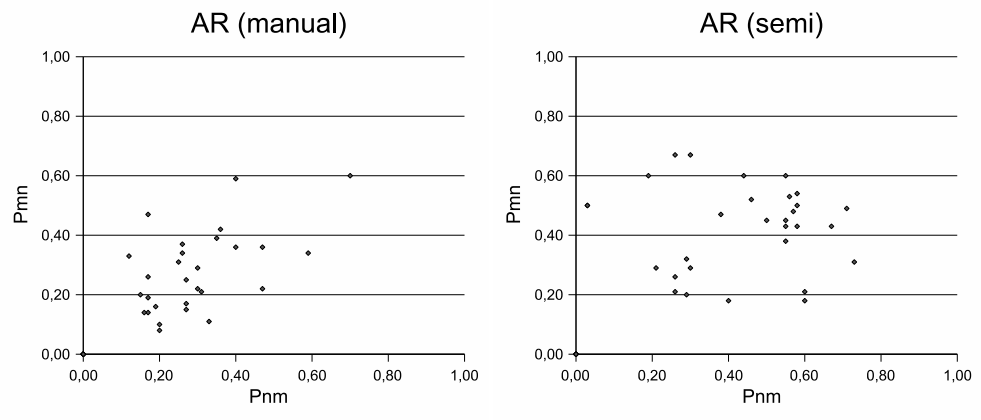

Figure 9.6.: Instance-instance relationship of group A

- The values for manual annotation are rather low, a number of data points are below the 0.4 line with a high variance. The values are ranging between 0 and 0.7.
- The values for semi-automatic annotation are mostly over 0.4, ranging between 0 and 0.73. The variance is also high.

Sliding Agreement Figure 9.7 shows the $\overline{IdMA}$ of group A.

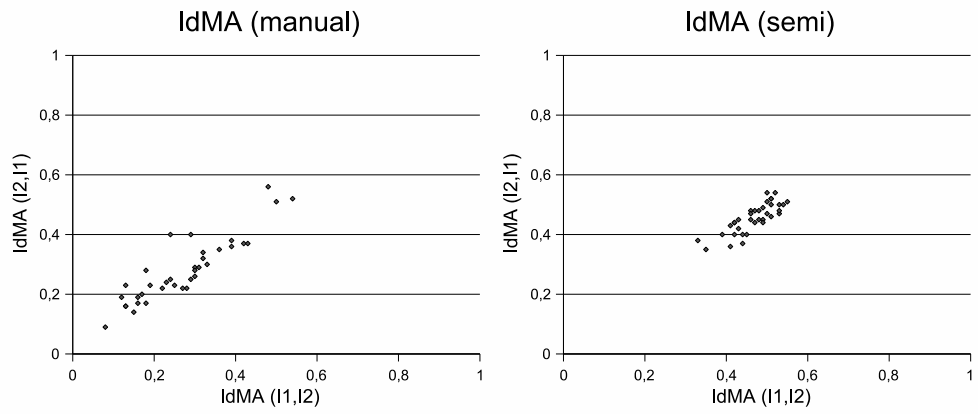

Figure 9.7.: Sliding agreement: $\overline{IdMA}$ of group A

- The values for manual annotation are higher, the values ranging between 0.08 and 0.64, most of them are below 0.4.
- The values for the semi-automatic are higher with a slight variance. The values are between 0.32 and 0.58. Nearly all are about 0.4.

Figure 9.8 shows the $\overline{RIAA}$ of group A:

- Similar to the $\overline{IdMA}$ values, the manual annotation has higher variance and lower $\overline{RIAA}$ values, ranging from 0.07 and 0.64. Most of the values are below 0.4.
- Semi-automatic annotation has the higher values, from 0.32 to 0.57 and the lower variance. Most of the points are above 0.4.

In conclusion, the values for the agreement recall are for the most part higher when working in semi-automatic annotation mode. Also the variance is lower and therefore the annotation more homogeneous in the case of the instance classification, the instance-concept relationships and for the sliding agreement.

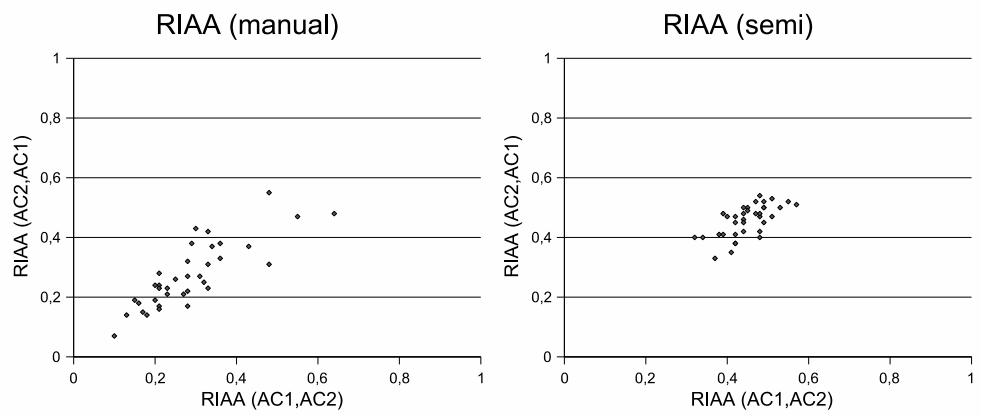

Figure 9.8.: Sliding agreement: $\overline{RIAA}$ of group A

9.3.3. Results of Group B

Basic Statistic. Table 9.4 shows the basic statistic for group B for both annotation modes. A comparison with Table 9.3 shows that group B produced systematically more annotation than group A. The average number of instances for manual annotation is 74.33 with a deviation of 14.88. For semi-automatic annotation it is 60.88 instances with a deviation of 11.12. Nearly all annotators surpass the gold standard with regard to the amount of instances.

Perfect-Interannotator Agreement Figure 9.9 shows the results for the instance identification.

- The annotation is quite homogeneous, with values between 0.15 and 0.51.
- Semi-automatic annotation yields higher results, with a higher variance. The values are from 0,18 to 0.66.

The graphical results for the instance-concept relationships are shown in Figure 9.10.

- The values for manual annotation are rather high with a median variance. The values are between 0.41 and 0.87.
- The values for semi-automatic annotation are higher, the minimum and maximum are 0.37 and 0.89.

	manual				semi		
user	$\mid I_n \mid, \mid AC_n \mid$	$\mid AA_n \mid$	$\mid AR_n \mid$		$\mid I_n \mid, \mid AC_n \mid$	$\mid AA_n \mid$	$\mid AR_n \mid$
0	55	62	40		47	48	33
1B	60	44	25		61	52	50
2B	88	73	51		53	37	40
3B	74	71	46		63	61	47
4B	69	67	51		62	61	40
5B	63	56	33		45	44	17
6B	98	116	42		79	87	55
7B	71	82	52		66	75	49
8B	91	79	65		72	78	48
total	669	650	405		548	543	379
mean	74.33	72.22	45		60.88	60.33	42.11
standard deviation	14.88	20.16	11.72		11.12	16.85	11.49

Table 9.4.: Basic statistics for group B

The results for the instance-attribute relationship are shown in Figure 9.11.

- The values for manual annotation are low. Most of them are below 0.4. The variance is high and the values range between 0 and 0.59.
- The values for semi-automatic annotation are higher. Most of the values are over 0.4. The variance is higher and the values are between 0 and 0.81.

The results for the instance-instance relationship are shown in Figure 9.12.

- The values for manual annotation are rather low, a number of data point are below the 0.4 line with a high variance. The values range between 0.04 and 0.65.
- The values for semi-automatic annotation are mostly over 0.3, ranging between 0.17 and 0.76. The variance is also high.

In summery, semi-automatic annotation results for the perfect inter-annotator agreement show more conformity and more variance.

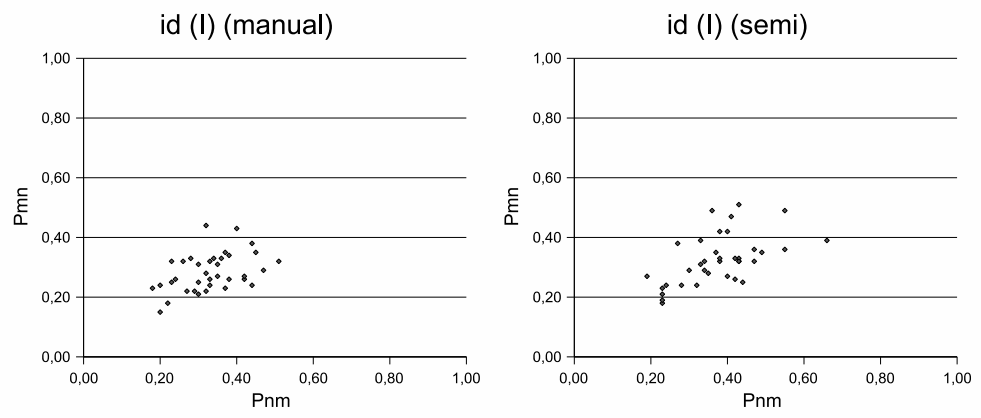

Figure 9.9.: Instance identification of group B

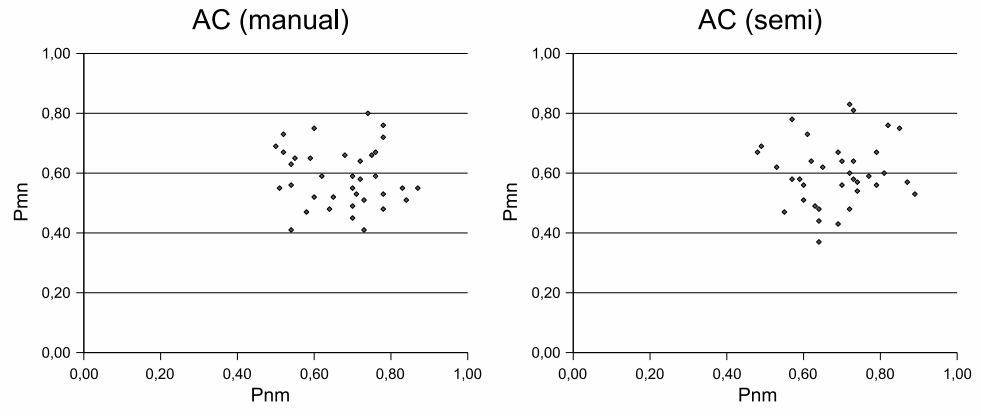

Figure 9.10.: Instance-concept relationship of group B

Sliding Agreement Figure 9.13 shows the $\overline{IdMA}$ of group B.

- The values for manual annotation are between 0.18 and 0.57.
- Semi-automatic annotation shows more values above 0.4. The values are between 0.22 and 0.68.

Figure 9.14 shows the $\overline{RIAA}$ of group B.

- The values for manual annotation range between 0.18 and 0.53.

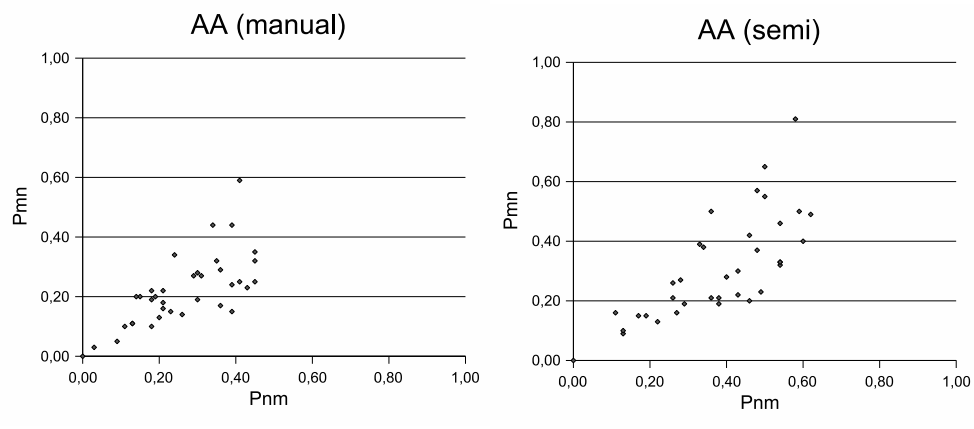

Figure 9.11.: Instance-attribute relationship of group B

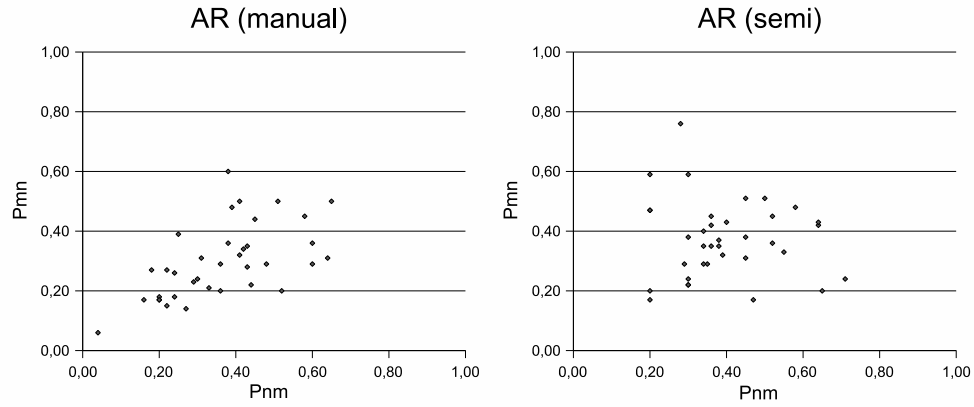

Figure 9.12.: Instance-instance relationship of group B

- Semi-automatic annotation has, similar to those of the $\overline{IdMA}$, higher values and a greater variance. The values range between 0.18 and 0.57

9.3.4. Statistical Results

We used the t-test for the samples to verify their significance. Table 9.5 shows an overview.

The meaning of the symbols are the same as in Table 9.2. We used the f-measure for the test: the inter-annotator agreement between annotator n and annotator m is the f-measure of $R(n, m)$ and $P(n, m)$, whereas $P(n, m) = R(m, n)$. We

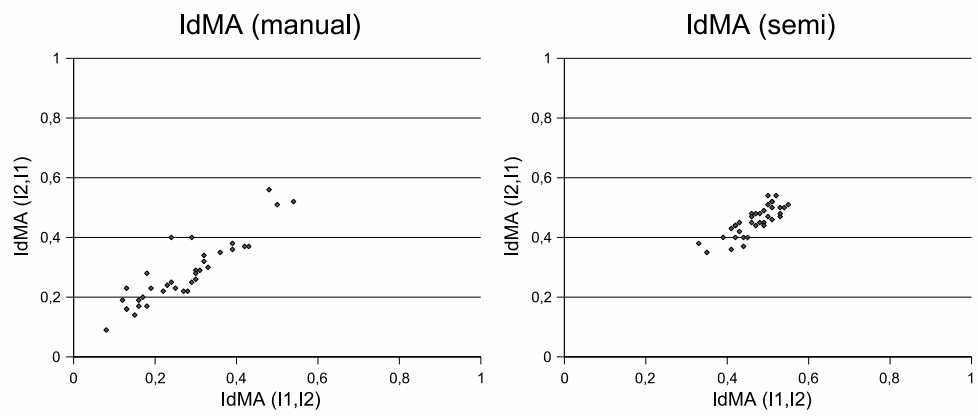

Figure 9.13.: Sliding agreement: $\overline{IdMA}$ of group B

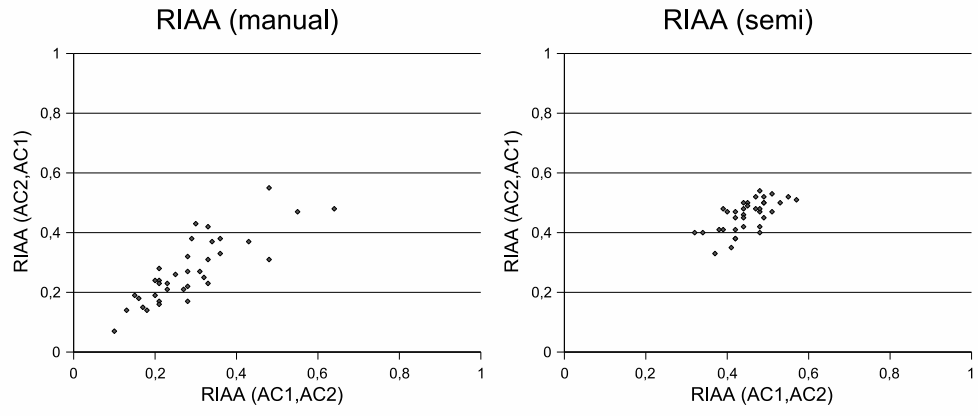

Figure 9.14.: Sliding agreement: $\overline{RIAA}$ of group B

compared the inter-annotator agreement between two annotators n and m in the manual mode with two annotators using the semi-automatic mode of the tool, for the aggregated test. The tables shows the following results:

- For the tests evaluating the groups, all differences are significant with the exception of the instance-concept relationship for group B. In this case the results for manual and semi-automatic annotation are the same.
- For the tests evaluating the overall results of the group, all differences are significant. For this reason we able to accept the second part of our research hypothesis: *Semi-automatic annotation produces more homogeneous anno-*

1. Variable	2. Variable	t	$t_{\alpha;n-1}$	PR	n	Significance
instance identification						
manual	semi-automatic	-9,3	1,67	0	72	yes
A, manual	A, semi-automatic	-13,02	1,69	0	36	yes
B, manual	B, semi-automatic	-3,91	1,69	0	36	yes
instance-concept						
manual	semi-automatic	-5,93	1,67	0	72	yes
A, manual	A, semi-automatic	-8,97	1,69	0	36	yes
B, manual	B, semi-automatic	-0,88	1,69	0,38	36	no
instance-attribute						
manual	semi-automatic	-8,96	1,67	0	72	yes
A, manual	A, semi-automatic	-10,76	1,69	0	36	yes
B, manual	B, semi-automatic	-3,82	1,69	0	36	yes
instance-instance						
manual	semi-automatic	-4,79	1,67	0	72	yes
A, manual	A, semi-automatic	-5,12	1,69	0	36	yes
B, manual	B, semi-automatic	-1,89	1,69	0,07	36	yes
$\overline{IdMA}$						
manual	semi-automatic	-8,58	1,67	0	72	yes
A, manual	A, semi-automatic	-12,85	1,69	0	36	yes
B, manual	B, semi-automatic	-3,12	1,69	0	36	yes
$\overline{RIAA}$						
manual	semi-automatic	-8,36	1,67	0	72	yes
A, manual	A, semi-automatic	-12,03	1,69	0	36	yes
B, manual	B, semi-automatic	-2,88	1,69	0,01	36	yes

Table 9.5.: Results of the t-test over the annotation time

tation than manual annotation.

Percentage We calculated the mean value for each annotation mode to quantify the evaluation results. Subsequently, we calculated the percentage increase of the inter-annotator agreement using semi-automatic annotation (see Table 9.6).

Metrics	manual	semi-automatic	Change
instance identification	0.28	0.39	40.3%
instance concept	0.58	0.67	14.58 %
instance attribute	0.19	0.38	96.33 %
instance instance	0.26	0.34	29.82 %
$\overline{IMA}$	0.32	0.43	34.19 %
$\overline{RIAA}$	0.31	0.41	33.61%

Table 9.6.: Percentage change of the mean value for the inter-annotator agreement

The table shows that the increases range from 14.57% for the instance-concept relationship to 96.33% for the instance-attribute relationship. The minimal value for the instance-concept relationship corresponds to the non-significance of this value in the t-test.

Analysis of the Questionnaire In addition to the evaluation, the annotators also completed a questionnaire. The results are presented in the following:

- General Questions:
  - Q: The concept of Semantic Annotation is . . . ?
    A: Ranging from 1 (very easy to understand) to 6 (very hard to understand).
    mean value: 2.25, max: 4, min: 1
  - Q: How do you evaluate your previous knowledge in the area of Semantic Annotation?
    A: Ranging from 1 (very good) to 6 (very poor).
  - Q: How well did you cope with the ontology?
    A: Ranging from 1 (very well) to 6 (very badly).
- Manual Annotation
  - Q: How do you like the user interface?
  - Q: How understandable is the layout of the user interface?
  - Q: How clear is the information presented?
  - Q: How self-explanatory are the symbols?

  - Q: How did you like the handling of the tool?
  - Q: How was the overall impression of the tool?

- Semi-automatic Annotation
  - Q: How do you like the user interface?
  - Q: How understandable is the layout of the user interface?
  - Q: How clear is the information presented?
  - Q: How self-explanatory are the symbols?
  - Q: How did you like the handling of the tool?
  - Q: What was your overall impression of the tool?

- General Impression
  - Q: Which annotation mode would you prefer for the task?

In conclusion the questionnaire shows the following:

- The participants had little previous knowledge about semantic annotation. However, they found the idea easy to understand.
- There were some suggestions on ways to improve the ontology, but in general they found it easy to use.
- The user interface for the manual annotation mode made a good impression on the participants. Their usage was problem-free.
- Similarly the user interface for semi-automatic annotation made a good impression. The marks for this interface are higher than for manual annotation.
- Nearly all participants preferred semi-automatic annotation, four of them without reservations.
- There were three participants who preferred manual annotation mode. The main reason was the problem of annotating systematically with the semi-automatic mode in the way they liked.

9.3.5. Summary

The evaluation proved the advantage of semi-automatic annotation with respect to annotation time and homogeneity. However, some results of group B, the participants in which annotated initially in the semi-automatic annotation mode, show no difference. Presumably, the training effect was offset by the positive

impact of the semi-automatic annotation. In general the participants of the evaluation were happy with the improvement of semi-automatic annotation. However, there were some participants in group B who had some doubts. Especially for group B it wasn't easy to learn all at once the interaction and the concept of Semantic Annotation and to handle the information extraction necessary for semi-automatic annotation. Hence, we can deduce that one first has to understand the concept of Semantic Annotation and face the inherent problems in order to appreciate and profit from the assistance of semi-automatic annotation.

The comments of the participants with regards to the user interface reflect the development status of the annotation tool at the moment of the evaluation. Some of these improvements have been implemented in the meantime. However, issues of human computer interaction are important when dealing with manual and semi-manual annotation but these issues haven't been the main focus in this thesis and they are therefore a possible target for future work.

9.4. Discussion

The results of the evaluation allow us to accept our research hypothesis, viz.:*Semi-automatic annotation is faster and more homogeneous than manual annotation.* We are satisfied with this overall result but we will in the following paragraph have a discussion about possible improvements to the evaluation and the annotation tools.

Evaluation It is important to find standardized methods and metrics for the evaluation of annotation frameworks. Hence, we presented such an approach in this work. Our experience with the actual evaluation shows also some of the unsettled points, especially soft facts, when measuring the inter-annotator agreement:

- **Similarity of Web pages**. One problem is the similarity of the Web pages in our use case, e.g. every page contained the concepts: Hotel, Address, SingleRoom, and so on. Very similar Web pages facilitates training in information extraction and helps therefore to improve instance recognition by the semi-automatic annotation. But also it alleviates the task for the annotator, so that he can easily spot the necessary pieces of text on the Web page. In this case the support of the semi-automatic annotation might not be considered by the user as useful as in a scenario with greater variance among the Web pages.
- **Instance Names** In the hotel use case presented here, many instances have the same names as the concepts of the ontology, e.g. the text "TV"

in the description of a hotel room indicates an unspecific instance of the TV that is related to the instance of the concept Room. This simplifies the information extraction and therefore the annotation task, which leads to a higher value for the instance-concept relationship metric than in the home page use case.

- **Guidelines** We gave the annotator some guidelines for the annotation, i.e. which parts of the page to annotate. This was done in order to avoid too much being annotated, meaning too much time would be needed, and also to avoid too little of a page being annotated and too few data for an evaluation being produced. Some participants follow these guidelines. Others did not follow the guidelines and annotated everything which was represented in the ontology. Although we can not measure the influence of the guidelines in the evaluation results, it is clear that it improved the general level of inter-annotator agreement compared with a free annotation.

- **Relational Metadata** A important concept of the CREAM framework is the creation of relational metadata not only on one web page but over several Web pages, i.e. to inter-connect Web pages with metadata. However, in the hotel use case there weren't enough pages with interrelated information. Also, the focus on annotating one page facilitates the annotation task for the unexperienced user. However, future evaluations should also consider the aspect of related metadata between Web pages.

The Annotation Tool OntoMat. Clearly the annotation tool is the most important factor when it comes to the improvement of the annotation task. Specifically, we found the following aspects during the evaluation:

- The hotel use case shows the problem for the need to annotate alternatives. For example, there is the description that a room might either have a bathtub or a shower. Here the annotation tool didn't allow a logical "OR" relationship between the instances to be modeled.

- To support inverse relationships is important. At the time of the evaluation the annotation tool didn't supported this. Therefore, one had to model the relationship "has_room" for a hotel and "in_hotel" for a room. A support facility for inverse relationships would automatically create the inverse.

- The automatic creation of "anonymous" instance names was only partially useful in the evaluation. It just numbered the instances, e.g. "room1", "room2", which gave the annotators some trouble to distinguish rooms of different hotels, e.g. different Web pages, in one session. Hence, the created name or rather the representation of the instance in the annotation tool

should give information for an instance about the selected text, the Web page of origin and the related concept.

- The description of hotel Web pages requires a lot of "anonymous" instances, of concepts such as Bath, Shower, SingleRoom, ParkingPlace. Most of these instances do not have attributes and it is very ineffective to create these instances again and again. A practical idea might be to reuse these instances. However, this would result in an identity problem, e.g. the shower of room X is then the same as the shower of room Y. This shows the need for a special handling of such "anonymous" instances in an ontology.

- In a lot of cases, rooms of different hotels have the same or similar furnishings. Therefore it should be possible to copy such room instances from one hotel to another hotel and change the appropriate attributes.

- In the original version – not used in the evaluation – of the *GETESS* ontology is a concept of currency, with the subconcepts Euro, US Dollar, Yen, and so on. The annotation tool so far only allows relationships between concept instances, so that a relationship to a concept is not possible.

The Information Extraction Component Amilcare The success of semi-automatic annotation depends on the information extraction component. The finding of the information extraction is acceptable, but could be improved. Amilcare was able during the evaluation to find about half of the instances and a third of the attributes. There are some possibilities for improving this:

- At the time of the evaluation it was not possible to edit the extraction rules of Amilcare to enable a fine tuning. In the meantime it has become possible to do this. However, this demands expert knowledge, because the rules are not intuitive to understand and change.

- Amilcare works with positive and negative examples. This requires that every occurrence of an instance on the Web page has to be annotated for the training. This leads to additional requirements we did not fulfill with our paradigm of Semantic Annotation, because i) a fact is represented and annotated only once for Semantic Annotation and ii) information extraction considers the whole Web page source not the rendered Web page, i.e. comments, script, meta tags etc. hidden in the Web page as well as the title tag also needed to be marked-up for Amilcare. Otherwise it would be handled as a negative example. This full source annotation is not supported by our annotation tool.

In addition there is always information which is more difficult to extract using Amilcare, e.g. the sentence "we are happy to organize your business meeting with

up to 60 persons". Amilcare didn't extract the information that this concerns a conference room with the appropriate capacity, because sentences like this appear seldom and varying in the corpus, so that Amilcare is not able to create a rule in that instance.

GATE Amilcare is based on GATE[4] (General Architecture for Text Engineering), a tool for language engineering developed by the University of Sheffield. The task of GATE is the preprocessing of the text, for example the *tokenization* or *sentence identification*. GATE allows also the markup of documents. However, it supports a flat annotation without the basis of an ontology. The kind of annotation generated by GATE is the input that Amilcare needs for the training. We considered using GATE to create the annotation for the training. However, at the time of the evaluation the GATE annotation tool was not able to deal with HTML files without destroying the HTML tags. Hence, we extended our annotation tool to transform the ontology-based Semantic Annotation into the flat annotation needed by Amilcare and GATE.

9.5. Conclusion

The evaluation shows that semi-automatic annotation with the help of information accelerates the annotation task and increases the inter-annotator agreement. In general the annotators are pleased with the proposal of the tool and therefore gave better marks for the semi-automatic annotation mode. The evaluation also shows the potential for improving automatization in terms of i) user interface of the annotation tool, ii) the ontology guidance of the annotation tool, and iii) the training of the information extraction component. However, the evaluation also shows the limits of this approach, especially the need for a big corpus of similar documents for the training of the information extraction component as we saw in the hotel use case.

[4] http://gate.ac.uk

Part IV.

Related Work & Conclusions

"All Life Is Problem Solving."
— Karl R. Popper

10. Comparison with Related Work

Semantic Annotation as we have presented it in this thesis is a cross-sectional enterprise.[1] Therefore there are a number of communities that have contributed towards achieving the objective of Semantic Annotation. For our annotation framework we distinguish the related work between the i) basic framework CREAM (Section 10.1) ii) the extended framework (Section 10.2) and iii) special annotation applications (Section 10.3).

10.1. Related Work for the Basic Framework

The basic framework of CREAM can be compared along four dimensions: Firstly, it is a framework for markup in the Semantic Web. Secondly, it may be considered as a particular knowledge acquisition framework that is to some extent similar to Protégé-2000 [Eriksson *et al.*, 1999]. Thirdly, it is certainly an annotation framework, though with a different focus than ones like Annotea [Kahan *et al.*, 2001]. And finally it is an authoring framework with an emphasis on metadata creation.

10.1.1. Knowledge Markup in the Semantic Web

We know of three major **early systems** that use knowledge markup intensively in the Semantic Web, viz. SHOE [Heflin and Hendler, 2000b], Ontobroker [Decker *et al.*, 1999], and WebKB [Martin and Eklund, 1999]. All three of them rely on markup in HTML pages. They all started with providing manual markup by editors. However, our experiences (cf. [Erdmann *et al.*, 2000]) have shown that text-editing knowledge markup yields extremely poor results, viz. syntactic mistakes, incorrect references, and all the problems outlined in the scenario section.

The SHOE Knowledge Annotator is a Java program that allows users to mark-up Web pages with the SHOE ontology. The SHOE system [Luke *et al.*, 1997] defines additional tags that can be embedded in the body of HTML pages. The SHOE Knowledge Annotator is a little helper (like our earlier OntoPad [Fensel *et al.*, 1999], [Decker *et al.*, 1999]) rather than a fully-fledged annotation environment. WebKB [Martin and Eklund, 1999] uses conceptual graphs for representing the

[1] Just like the Semantic Web overall!

semantic content of Web documents. It embeds conceptual graph statements into HTML pages. Essentially it offers a Web-based template-like interface such as knowledge acquisition frameworks described in the following section.

A more **recent contribution** is the RDF annotator SMORE[2]. SMORE allows markup of images and emails as well as HTML and text. A tool with similar characteristics to SMORE is the Open Ontology Forge (OOF) [Collier *et al.*, 2004]. OOF is seen by its creators as an ontology editor that supports annotation, taking it a step further towards an integrated environment to handle documents, ontologies and annotations.

The next group of markup tools that we will discuss is semi-automatic, they have automatic components but assume intervention by the user in the annotation process. In this line, the tool the most similar to OntoMat is the system from The Open University [Lei *et al.*, 2002] and the corresponding MnM [Vargas-Vera *et al.*, 2002] annotation tool. MnM [Vargas-Vera *et al.*, 2002] also uses the Amilcare information extraction system. It allows the semi-automatic population of an ontology with metadata. So far, they have not dealt with relational metadata or authoring concerns. Another weakness is that it is restricted to either marking up the slots for a single concept at a time or marking up all the concepts on a single hierarchical level of a single ontology (but not their slots).

AeroSWARM[3] is an automatic tool for annotation using OWL ontologies based on the DAML annotator AeroDAML [Kogut and Holmes, 2001]. This has both a client server version and a Web enabled demonstrator in which user enters a URI and the system automatically returns a file of annotations on another Web page. To view this in context the user would have to save the RDF to an annotation server and view the results in an annotation friendly browser such as Amaya. AeroDAML is therefore not in itself an annotation environment. SemTag is another example of a tool which focuses only on automatic mark-up [Dill *et al.*, 2003]. It is based on IBMs text analysis platform Seeker and uses similarity functions to recognize entities which occur in contexts similar to marked up examples. The key problem of large scale automatic markup is identified as ambiguity, e.g. identical strings, such as Niger which can refer to different things, a river or a country. A Taxonomy Based Disambiguation (TBD) algorithm is proposed to tackle this problem. SemTag is proposed as a bootstrapping solution to get a semantically tagged collection off the ground. AeroSWARM and SemTag, as the most large scale automatic markup systems, focus on class instantiation, in the goal comparable to our PANKOW approach. Hence, they also have not dealt with the creation of relation metadata.

KIM [Popov *et al.*, 2003] uses information extraction techniques to build a large knowledge base of annotations. The annotations in KIM are metadata in the

[2]http://www.mindswap.org/~aditkal/editor.shtml
[3]http://ubot.lockheedmartin.com/ubot/hotdaml/aeroswarm.html

form of named entities (people, places etc.) which are defined in the KIMO ontology and identified mainly from reference to extremely large gazetteers. This is restrictive, and it would be a significant research challenge to extend the KIM methodology to domain specific ontologies. However named entities are a class of metadata with broad usage and the KIM platform is well placed to showcase the kinds of retrieval and data analysis services that can be provided over large knowledge bases of annotations.

10.1.2. Comparison with Knowledge Acquisition Frameworks

The CREAM framework allows for creating class and property instances and for populating HTML pages with them. Thus, it targets a roughly similar objective to the instance acquisition phase in the Protégé-2000 framework [Eriksson *et al.*, 1999] (the latter needs to be distinguished from the ontology editing capabilities of Protégé). The obvious difference between CREAM and Protégé is that the latter does not (and was not intended to) support the particular Web setting, *viz.* managing and displaying Web pages — not to mention Web page authoring. From Protégé we have adopted the principle of a meta ontology that allows a distinction between different ways that classes and properties are treated. We just named Protégé-2000 as one example for other existing knowledge acquisition frameworks.

Another recent knowledge acquisition framework with particular applications in mind is TRELLIS [Gil and Ratnakar, 2002]. It is designed to support argument analysis in decision making scenarios. It demonstrates the additional support that can be given to user when an annotation environment is designed for a specific purpose. For example, annotations in TRELLIS are in the form of free text statements. This presents a problem since statements about the same thing can be phrased differently and consequently not matched up by the user. Therefore a component called ACE has been built which helps users to formulate statements in ways which are consistent with terms in the ontology [Blythe and Gil, 2004]. The annotations in TRELLIS can be output as RDF. However, perhaps because it is designed as a tool for analyzing a wide range of document formats, the authors do not discuss whether it is possible to anchor annotations to a particular part of a text.

10.1.3. Comparison with Annotation Frameworks

Types of Annotations

There are certain types of annotation. For example (i) the idea of creating a kind of user comment about Web pages (cf. [Kahan *et al.*, 2001]), (ii) to annotate a document with further links (cf. Section 2.2 in [Bechhofer *et al.*, 2002]) or (iii)

semantic annotation. Also, we can distinguish the Semantic Annotation into instance creation or aboutness (cf. Section 5 in [Bechhofer *et al.*, 2002]). Our viewpoint of annotation lies with instance creation (see Section 3.4.1), but we can emulate most of the other annotation types with our framework.

- User comment: For instance, a user attaches a note like "An interesting research topic!" to the term "Semantic Web" on a Web page. In CREAM we can design a corresponding ontology that allows the comment (an unlinked fact) "A interesting research topic" to be typed into an attribute instance belonging to an instance of the class comment with a unique XPointer at "Semantic Web".
- Link annotation: A user attaches a URL to a Web page. In CREAM we can design a corresponding ontology that allows the insertion of the URL (as reference) into an attribute instance belonging to an instance of the class URL.
- Aboutness: A user expresses that a particular resource (i.e. a web page) is about a certain concept, but not an instance of an concept. In CREAM we can introduce a meta-concept into the corresponding ontology. The instances of this meta concept are all concepts of the ontology. Therefore the user interface supports the creation of an aboutness annotation. For instance, a user inserts the URL of a page about students (e.g. http://www.student.de) (as a reference) to an attribute instance belonging to the instance of the class AboutStatement. Then he creates a relationship instance between the AboutStatement and the concept "student" of the ontology.

The term annotation is also used in the area of Natural Language Processing (NLP) with a different meaning (cf. Chapter 7.1 and 10.3.2). The levels of linguistic annotation comprise lemma, morphosyntactic, syntactic, semantic and discourse annotation [Leech, 1997]. There is also some preliminary work in order to attempt the combination of semantic Web page annotation with linguistic annotation [de Cea *et al.*, 2002].

Annotation Tools

There are a lot of — even commercial — annotation tools like ThirdVoice[4], Yawas [Denoue and Vignollet, 2000], CritLink [Yee, 1998] and Annotea (Amaya) [Kahan *et al.*, 2001]. These tools all share the idea of creating a kind of user comment about Web pages. The term "annotation" in these frameworks is understood as a remark assigned to an existing document.

[4] http://www.thirdvoice.com

Annotea actually goes one step further. It allows an RDF schema to be used as a kind of template that is filled by the annotator. For instance, Annotea users may use a schema for Dublin Core and fill the author-slot of a particular document with a name. This annotation, however, is again restricted to attribute instances. The user may also decide to use complex RDF descriptions instead of simple strings for filling such a template. However, he then has no further support from Amaya that helps him provide syntactically correct statements with proper references.

10.1.4. Comparison with Authoring Frameworks

An approach related to CREAM authoring is the Briefing Associate of Teknowledge [Tallis *et al.*, 2001]. The tool is an extension of Microsoft PowerPoint. It pursues the idea of producing PowerPoint documents with the metadata coding as a by-product of the document composition. For each concept and relation in the ontology, an instantiation button is added to the PowerPoint toolbar. Clicking on one of these buttons allows the author to insert an annotated graphical element into his presentation. Thus, a graphic element in the presentation corresponds to an instance of a concept and arrows between the elements corresponding to relationship instances. In order to be able to use an ontology in PowerPoint one must have assigned graphic symbols to the concepts and relations, which is done initially by the visual-annotation ontology editor (again a kind of meta ontology assignment). The Briefing Associate is available for PowerPoint documents. In contrast, CREAM does not provide graphic support like the Briefing Associate, but it supports both parts of the annotation process, i.e. it permits the simultaneous creation of documents and metadata and, in addition, the annotation of existing documents. However, the Briefing Associate has shown very interesting ideas that may be of future value to CREAM.

The authoring of hypertexts and the authoring with concepts are topics in the COHSE project [Goble *et al.*, 2001]. They allow for the automatic generation of metadata descriptions by analysing the content of a Web page and comparing the tokens with concept names described in a lexicon. They support ontology reasoning, but they do not support the creation of relational metadata. It is unclear to what extent COHSE considers the synchronous production of document and metadata by the author.

Somewhat similar to COHSE concerning the metadata generation, Klarity [Klarity, 2001] automatically fills Dublin Core fields taking advantage of statistical methods to allocate values based on the current document.

10.2. Related Work for the Extended Framework

In addition the extensions of the basic CREAM framework for deep annotation and pattern based annotation requires also the comparison with a number of community work.

10.2.1. Related Work for Deep Annotation

A number of communities that have contributed towards reaching the objective of deep annotation. So far, we have identified communities for information integration (Section 10.2.1), mapping frameworks (Section 10.2.1), wrapper construction (Section 10.2.1), and annotation (Section 10.2.1).

Information Integration

The core idea of information integration lies in providing an algebra that may be used to translate information proper between different information structures. Underlying algebras are used to provide compositionality of translations as well as a sound basis for query optimization (cf., e.g. a commercial system as described in [Papakonstantinou and Vassalos, 2002] with many references to previous work — a lot of the latter based on principal ideas issued in [Wiederhold, 1993].

Unlike [Papakonstantinou and Vassalos, 2002], our objective has not been the provision of a flexible, scalable integration platform *per se*. Rather, the purpose of deep annotation lies in providing a flexible framework for *creating the translation descriptions* that may then be exploited by an integration platform like EXIP (or Nimble, Tsimmis, Infomaster, Garlic, etc.). Thus, we have more in common with the approaches for creating mappings for the purpose of information integration described next.

Mapping and Merging Frameworks

Approaches for mapping and/or merging ontologies and/or database schemata may be divided mainly into the following three categories: discovery, [Rahm and Bernstein, 2001; Cohen, 1998; Doan *et al.*, 2002; Bergamaschi *et al.*, 2001; Noy and Musen, 2000; McGuinness *et al.*, 2000], mapping representation [Madhavan *et al.*, 2001; Bergamaschi *et al.*, 2001; Mitra *et al.*, 2000; Park *et al.*, 1997] and execution [Critchlow *et al.*, 1998; Mitra *et al.*, 2000].

In the overall area, closest to our own approach is [Maedche *et al.*, 2002], as it handles — like we do — the complete mapping process involving the three process steps just listed (actually it also takes care of some more issues like evolution).

What makes deep annotation different from all these approaches is that they all depend on lexical agreement of part of the two ontologies/database schemata for the initial discovery of overlaps between different ontologies/schemata. Deep annotation only depends on the user understanding the presentation — the information within an information context — developed for him/her anyway. Concerning the mapping representation and execution, we follow a standard approach exploiting Datalog, giving us many possibilities for investigating, adapting and executing mappings as described in Section 6.7.

Wrapper Construction

Methods for wrapper construction achieve many objectives that we pursue with our approach of deep annotation. They have been designed to allow for the construction of wrappers by explicit definition of HTML or XML queries [Sahuguet and Azavant, 2001] or by learning such definitions from examples [Kushmerick, 2000; Ciravegna, 2001a]. Thus, it has been possible to manually create metadata for a set of structurally similar Web pages. The wrapper approaches come with the advantage that they do not require cooperation by the owner of the database. However, their disadvantage is that the correct scraping of metadata is dependent to a large extent of data layout rather than on the structures underlying the data.

Furthermore, when definitions are given explicitly [Sahuguet and Azavant, 2001], the user must directly cope with querying by layout constraints and when definitions are learned, the user must annotate multiple web pages in order to derive correct definitions. Also, these approaches do not map to ontologies. They typically map to lower level representations, e.g. nested string lists in [Sahuguet and Azavant, 2001], from which the conceptual descriptions must be extracted, which is a non-trivial task. In fact, we have integrated a wrapper learning method, *viz.* Amilcare [Ciravegna, 2001a], into our OntoMat-Annotizer. How to bridge between wrapper construction and annotation is described in detail in Section 5.1.

Annotation

Finally, we need to consider annotation proper as part of deep annotation. In the latter, we "inherit" the principal annotation mechanism for creating relational metadata as elaborated above.

10.2.2. Related Work for Pattern-based Annotation

In Chapter 5.2 we presented a novel paradigm: *the self-annotating Web* as well as an original method PANKOW which makes use of globally available structures

as well as statistical information to annotate Web resources. Though there are initial blueprints for this paradigm, to our knowledge there has been no explicit formulation of this paradigm as well as a concrete application of it as previous presented in Chapter 5.2.

On the other hand, there is quite a lot of work related to the use of linguistic patterns to discover certain ontological relations from text. Hearst [Hearst, 1992] and Charniak [Charniak and Berland, 1999] for example make use of a related approach to discover taxonomic and *part-of* relations from text, respectively.

Concerning the task of learning the correct class or ontological concept for an unknown entity, there is some related work, especially in the computational linguistics community. The aim of the Named Entity Task as defined in the MUC conference series ([Hirschman and Chinchor, 1997]) is to assign the categories *ORGANIZATION*, *PERSON* and *LOCATION*.

Other researchers have considered this more complex task such as Hahn and Schnattinger [Hahn and Schnattinger, 1998], Alfonseca and Manandhar [Alfonseca and Manandhar, 2002] or Fleischman and Hovy [Fleischman and Hovy, 2002].

Hahn and Schnattinger [Hahn and Schnattinger, 1998] create a *hypothesis space* when encountering an unknown word in a text for each concept to which the word could belong. These initial hypothesis spaces are then iteratively refined on the basis of evidence extracted from the linguistic context in which the unknown word appears. In their approach, evidence is formalized in the form of quality labels attached to each hypothesis space. At the end the hypothesis space with maximal evidence with regard to the qualification calculus used is chosen as the correct ontological concept for the word in question. Their approach is very closely related to ours and in fact they use similar patterns to identify instances from the text. However, the approaches cannot be directly compared. On the one hand they tackle categorization into an even larger number of concepts than we do and hence our task would be easier. On the other hand they evaluate their approach under clean room conditions as they assume accurately identified syntactic and semantic relationships and an elaborate ontology structure, while our evaluation is based on very noisy input — rendering our task harder than theirs.

Alfonseca and Manandhar [Alfonseca and Manandhar, 2002] have also addressed the problem of assigning the correct ontological class to unknown words. Their system is based on the distributional hypothesis, i.e. that words are similar to the extent to which they share linguistic contexts. Based on this premise, they adopt a vector-space model and exploit certain syntactic dependencies as features of the vector representing a certain word. The unknown word is then assigned to the category corresponding to the most similar vector. However, it is important to mention that it is not clear from their paper whether they are actually evaluating

their system on the 1200 synsets/concepts or only on a smaller subset of these.

Fleischman and Hovy [Fleischman and Hovy, 2002] address the classification of named entities into fine-grained categories. In particular, they categorize named entities denoting persons into the following 8 categories: *athlete*, *politician/-government*, *clergy*, *businessperson*, *entertainer/ artist*, *lawyer*, *doctor/scientist*, *police*. Given this categorization task, they present an experiment in which they examine 5 different Machine Learning algorithms: C4.5, a feed-forward neural network, k-nearest Neighbors, a Support Vector Machine and a Naive Bayes classifier. As features for the classifiers they make use of the frequencies of certain N-grams preceding and following the instance in question as well as topic signature features which are complemented with synonymy and hyperonymy information from WordNet. Fleischman and Hovy's results are certainly very high in comparison to ours – and also to the ones of Hahn et al. [Hahn and Schnattinger, 1998] and Alfonseca et al. [Alfonseca and Manandhar, 2002] – but on the other hand, however, they address a more complex task than the MUC Named Entity Task, they are still quite some way away from the number of categories we consider here.

10.3. Related Work for Applications of CREAM

The annotation application for the CREAM framework presented in this thesis are service annotation (10.3.1) and linguistic annotation (10.3.2).

10.3.1. Comparison with Service Annotation

In Chapter 7.2 we provide an original application of the CREAM framework, CREAM-Service, to embed the process of Web service discovery (here: by browsing Web pages and retrieving Web pages from search engines like Google), composition (here: by deep annotation and reasoning over logically possible configurations), and invocation (here: by OntoMat-Service-Surfer, and the mapping to a client ontology). The consideration of semantic heterogeneity is germane to CREAM-Service. It offers semantic translations as one of its core modules.

CREAM-Service does not aim at Web service discovery, composition and invocation that is intelligent in the sense that it completely automates the task that typically the user is supposed to do. Rather, it provides an interface, OntoMat-Service-Surfer, that supports the intelligence of the user and guides him to add semantic information to the extent that that only few logically valid paths remain to be chosen from by the user.

To fully pursue such an objective, one needs a large set of different modules. We have built on our existing experience and tool framework for Semantic Annotation

CREAM and for logical reasoning [Decker *et al.*, 1999]. We have not yet dealt with the issue of Web service flow execution and monitoring that is certainly needed to complement our current version of CREAM-Service.

Frameworks that facilitate the building of Web service flows come closest to our approach. A number of software systems are available to facilitate manual composition of programs, and more recently web services. Such programs, which include a diversity of workflow tools [van der Aalst, 1999; Ellis and Nutt, 1993], and more recently service composition aids such as BizTalk Orchestration [Lowe, 2001] enable a software engineer to manually specify a composition of programs to perform some task — though they typically neglect the aspect of semantic heterogeneity that is core to CREAM-Service[5].

Web Services Invocation Framework (WSIF) [Apache, 2004] is an open source framework to execute any Web service, that can be described by a WSDL document. However, it does not support the execution of a flow of web services.

Some technologies have been proposed that use some form of semantic markup of Web services in order to automatically compose Web services to perform some desired task (e.g. [Narayanan and McIlraith, 2002; Benjamins *et al.*, 1998; McIlraith and Son, 2002]). In [Narayanan and McIlraith, 2002], the authors use situation calculus for representing Web service description and Petri nets for describing the execution behaviors of Web services. In [Benjamins *et al.*, 1998], the authors present an architecture of intelligent brokers that offer problem solving methods that can be configured and used by the users according to their needs. In [McIlraith and Son, 2002] the authors propose an extended version of Golog for formalizing the provision of high-level generic procedures and customization of constraints. In [Ponnekanti and Fox, 2002], the authors propose a rule-based expert system to automatically compose Web services from existing Web services.

On the one hand most recent experiences from such advanced projects like IBrow, however, have shown that automatic composition techniques cannot yet be carried over to an open world setting. In such a scenario one needs to tightly integrate the user of a Web service — such as we do in CREAM-Service. On the other hand CREAM-Service can obviously be extended in the future to consider more types of automatic semantic matchmaking, service discovery [Paolucci *et al.*, 2002; Sycara *et al.*, 1999] and configuration of Web services into the Web service planning phase.

10.3.2. Comparison with Linguistic Annotation

There are a vast amount of frameworks and tools developed for the purpose of linguistic annotation. However, in Chapter 7.1 we focus on the discussion

[5] BizTalk even allows for XML-based (non-semantic) translations of data.

of frameworks for the annotation of anaphoric or discourse relations in written texts. In the annotation scheme proposed by [Müller and Strube, 2001] in the context of their annotation tool MMAX and in contrast to the one proposed in Chapter 7.1, anaphoric relations are restricted to coreferring expressions, while bridging relations are restricted to non-coreferring ones. In line with [Krahmer and Piwek, 2000] and [van Deemter and Kibble, 2000] this is, in our view, a too strict definition of anaphora, so that we propose a more relation-based classification of anaphoric and bridging relations. Furthermore, in [Müller and Strube, 2001], anaphoric relations are further differentiated according to the lexical items taking part in the relation. We have shown that under the assumption that the corresponding grammatical information is provided by the annotators, such a classification can be seen as a byproduct of a more semantic one such as outlined in Chapter 7.1. In addition, [Müller and Strube, 2001] propose to specify antecedence with regard to equivalence classes rather than with regard to particular antecedents. However, this has the disadvantage that the information about the actual antecedent which has been selected by an annotator is actually lost. Thus in our annotation proposal the fact that the *Coreference* relation forms equivalence classes is modeled by an underlying axiom system which can be exploited in the evaluation of a system against the annotation standard.

The annotation scheme proposed by Poesio et al. [Poesio and Vieira, 1998] is a product of a corpus-based analysis of definite description (DD) use showing that more than 50% of the DDs in their corpus are *discourse new* or *unfamiliar*. Thus in Poesio et al.'s annotation scheme definite descriptions are also explicitely annotated as discourse new.

The MUC coreference scheme [Hirschman and Chinchor, 1997] is restricted to the annotation of coreference relations, where coreference is also defined as an equivalence relation. Though this annotation scheme may seem quite simple, we agree with [Hirschman and Chinchor, 1997] that it is complex enough when taking into account the agreement of the annotators on a task. In fact, it has been shown that the agreement of subjects annotating bridging [Poesio and Vieira, 1998] or discourse [Cimiano, 2003] relations can be too low for tentative conclusion to be drawn [Carletta, 1996]. The motivation of the MUC coreference scheme was thus to develop an annotation scheme leading to a good agreement. On the other hand, our motivation is to show how our ontology-based framework can be applied to the annotation of anaphoric relations in written texts and from this perspective the MUC coreference annotation scheme would have been in fact too restricted to actually show all the advantages of our approach.

The UCREL [Fligelstone, 1992] and DRAMA [Passoneau, 1996] annotation schemes are more related to ours than the schemata above in the sense that they also provide a rich set of particular bridging relations that can be annotated. However, in contrast to the ontology-based framework presented here, these bridging relations are not constrained with regard to the conceptual types

of their arguments, so that erroneous annotations can not be avoided.

The coreference annotation scheme proposed within the MATE Workbench project consists of a core as well as an extended scheme [Davies *et al.*, 1998]. The core scheme is in principle identical to the MUC coreference scheme and is restricted to the annotation of coreference in the sense of [van Deemter and Kibble, 2000]. The extended scheme also allows the annotation of bound anaphors, of the relationship between a function and its values, of different set, part and possesion relations, of instantiation relations as well as of event relations. The MATE scheme is related to our ontology-based annotation scheme in the sense that relations are also annotated as triples via the *link*-tag [Davies *et al.*, 1998]. As in our framework, the MATE scheme also allows ambiguities of reference to be marked up. However, in contrast to the MATE scheme our framework has no means to specify a preference order on these ambiguous antecedents. On the other hand, the MATE scheme also includes a reasonable and complete taxonomy of markables as well as some features relevant for the annotation of coreference in dialogues such as the treatment of hesitations, disfluencies and misunderstandings.

10.4. Related Work for Evaluation

Our evaluation of inter–annotator agreement is a corresponding investigation to studies of consist inter–linking of hypertexts or inter-indexing in library science. On a rough view there is an analogy between indexing a document and identifying ontology objects for documents. There is also an analogy for the creating of links between hypertext nodes compared to creating relations between ontology objects. However, the goals of each approach are not quite comparable and the ontology structures are more complex than hypertext links and indices.

The survey [Leonard, 1977] describes the measurement of the extent to which agreement exists among different indexers on the sets of index terms to be assigned to individual documents. The study shows that there is a low level of agreement between the sets of index terms assigned to a document by different indexers. Even the levels of consistency identifiable in the work of a single indexer on a collection of documents are often comparably low. The results from the measurement of inter-linker consistency in hypertext databases as shown in [Ellis *et al.*, 1994] are similar. The work describes an experiment in which the degree of similarity is measured between a number of hypertext databases that share a common set of nodes but whose link-sets have been manually created by different people. In the result the inter–linker consistency is low and varying. The results of inter–indexing and inter–linking studies are comparable with our principal conclusion that high levels of agreement are rarely achieved.

A lot of work on evaluating information extraction systems has been done in the

Message Understanding Conferences (MUC). In [Lehnert *et al.*, 1994] it is described how the basic evaluation text corpus has been developed in a distributed manner. All contributing sites generated template representations for some specified segment of the 1300 texts. Pairwise combinations of sites were expected to compare overlapping portions of their results and work out any differences that emerged. The authors note that it takes an experienced researcher three days to cover 100 texts and produce good quality template representation for these texts. However the question, how quality is measured, remains open in their paper. The authors state that their "estimate also finesses the fact that two people will seldom agree on the complete representation which can then be compared, discussed and adjusted as needed."

[Noy *et al.*, 2000] describes an empirical evaluation of a knowledge acquisition tool with the target of building domain knowledge bases. Military experts have been taken as subjects that had no experience in knowledge acquisition or computer science in general. Evaluation criteria are defined along several dimensions, namely the knowledge-acquisition rate, the ability to find errors, the quality of knowledge entries, the error–covery rate, the retention of skills and the subjective opinion. The results document the ability of these subjects to work on a complex knowledge-entry task and highlight the importance of an effective user interface enhancing the knowledge– acquisition process.

11. Conclusion and Future Work

This thesis began with the observation that Semantic Annotation is a prerequisite for the Semantic Web. The thesis shows how the problem of metadata creation can be tackled by means of a comprehensive annotation framework. The dissertation discusses the requirements for the design, application and evaluation of an ontology-based annotation framework as well as the prototypical implementation.

The concluding chapter is divided into four sections describing the contributions of this thesis in more detail, followed by the insight gained into Semantic Annotation, the open questions and topics for future research.

11.1. Contributions

The contributions made by this thesis fall into the following categories:

- **Identification of the annotation problem and definition of the requirements**. Given the problems with syntax, semantics and pragmatics with annotation we identified the requirements of: consistency, proper reference, avoidance of redundancy, relational metadata, maintenance, ease of use and efficiency.
- **Design of a comprehensive and pioneering annotation framework** that reduces the complexity of Semantic Annotation for the annotator. The framework employs a comprehensive set of modules including inference services, crawler, document management system, ontology guidance/fact browser, and document editors/viewers. Process issues pertaining to the annotation/authoring task are modularized from content descriptions by a meta ontology. The framework has been **prototypically implemented** in the open source project OntoMat[1] hosted by the DARPA DAML program. OntoMat is the reference implementation of the CREAM framework. It is Java-based and provides a plug-in interface for extensions for further applications. It has been used in several cases, e.g. the annotation of paper abstracts for the International Conferences on Semantic Web (ISWC 2002,

[1] http://projects.semwebcentral.org/projects/ontomat/

2003, 2004) by each of the authors. Also it is in use on class room machines in an obligatory Semantic Web course[2] for informatics students in Prague,on which some 250 people enroll every year.

The annotation framework comprised methods especially for:

- **Manual Annotation**: The transformation of existing syntactical resources, viz. documents, into relational knowledge structures which represent the underlying information.
- **Authoring of Documents**: Authoring lets users create metadata — almost for free — while putting together the content of a page.
- **Semi-automatic Annotation**: Semi-automatic Annotation based on Information Extraction.
- **Deep Annotation**: Considers Web pages which are generated from a database by annotation of the underlaying database.

- Demonstration of the flexibility of the annotation framework by the application on **linguistic annotation** and **service annotation**. The thesis shows how to exploit the basic framework easily for linguistic annotation. Furthermore, it shows how to apply the deep annotation framework also to annotation and composition of Web services.

- The thesis shows how to**evaluate Semantic Annotation** using several different measures applied within a gold standard and inter-annotator setting. It provides a case study for evaluating human annotation in comparison with semi-automatic annotation evaluating the annotation framework.

11.2. Insights into Semantic Annotation

In addition to the major contributions listed above, this research provides several additional insights into Semantic Annotation. This section lists the most salient perspectives.

- **Comprehensive Annotation Format**: Semantic Metadata are, simply expressed, facts that are related to a domain ontology. Though this may appear trivial at first, however this easily conflicts with several other requirements. We also need a *meta ontology* describing how the domain ontology should be used by the annotation framework. Furthermore, there is the requirement for remote storage of annotation, which leads to the need for a robust referencing scheme, viz. *XPointer*. Also, there is the need for the provision of *metametadata*, e.g. author, date, time and location of

[2] http://nb.vse.cz/~svatek/modz.htm

annotation. In addition, different requirements exist for different *semantics* of Semantic Annotation as well as the need to express different aspects of the content in metadata viz. a *layering* of the annotation (e.g. Structural Annotation, Lexical Annotation, Semantic Annotation).

- **Automatization**: Automatization is vital to ease the knowledge acquisition bottleneck. To achieve this, the integration of knowledge extraction technologies into the annotation environment has been undertaken. This is used to semi-automatically identify entities in text that are instances of a particular class and relations between the classes. As the evaluation showed, HCI implications are also important here, so that a semi-automated tool can be used effectively by Web users without expertise in natural language processing methods.

- **User centric design**: Annotation is a potential knowledge acquisition bottleneck as discussed above. To ease the constriction, annotation has to be carried out by people who are not specialist annotators. To facilitate the annotation task is especially important for the success of the Semantic Web. The Annotation interfaces must, therefore, bridge the gap between formal descriptions of knowledge and Web users understanding their domains of interest. A good approach is therefore a semantic authoring environment, so that the environment in which users annotate documents is the same as the one in which they create, read and edit them.

11.3. Open Questions

In a topic like Semantic Annotation, which is novel and relatively unexplored it is natural that it raises new questions while it answers old ones. Hence, the general problem of metadata creation remains interesting. In this section the open questions that are not answered are identified:

- Firstly, the question of **scalability** to more and larger dimensions. Like "what happens if there are 100,000 people known in your annotation inference server?". Even for the evaluation we had to prune the ontology in order to make it feasible for the annotation task.

- Secondly, Semantic Annotation takes place within the Semantic Web. For the proper creation of relational metadata we need **unique identifiers** of persons, institutes or companies. While crawling of existing metadata helps to reduce this problem it is not solved and possibly may never be.

- Thirdly, we are still in the early stages with respect to providing **methodological guidelines** for the purposes of Semantic Annotation.

- Fourthly, probably the most important for the Semantic Web. How to create **incentives** for annotation?

11.4. Future Research

A number of future research topics and challenges have already been listed at the end of each chapter, e.g.

- **Improvement of semi-automatic Annotation**: The process to generate the discourse representation is not automatic but relies on hand crafted rules. In the ideal scenario, the information extraction system would be ontology-aware and therefore learn the discourse structure of the given documents as well. To extend the information extraction system in this way was not possible for us because we were able to use Amilcare only as a black box component. We envisage therefore further research to specify and implement this type of an ontology-aware information extraction component. In addition, to use information extraction one needs a rather large amount of previously annotated documents, which is only useful if one has to annotate a group of similar documents. As regards to relational metadata, we have only covered the relationships within one document, but not relationships which points towards another document, viz. semanticly enriched hyperlinks. The semi-automatic annotation of interrelated documents is therefore a topic for further research.

- **Improvement of Self-Annotating Web**: The self-annotating Web is not possible with the kind of implementation that we provided. The reason for this is that we have issued an extremely large number of queries against the Google API — to an extent that would not scale towards thousands or even millions of Web pages in a reasonable time-frame. However, we envision an optimized indexing scheme and API that would reduce this number to acceptable load levels. Furthermore, in order to reduce the amount of queries sent to the Google Web service API, a more intelligent strategy should be devised, which takes into account the ontological hierarchy between concepts.

- **Improvement of Deep Annotation**: Granularity and automatic derivation of server-side Web page markup. Granularity: So far we have only considered atomic database fields. For instance, one may find a string "Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002." as a title of a book whereas one might rather be interested in separating this field into title, location and date. Automatic derivation of server-side Web page markup:

A content management system like Zope could provide the means for automatically deriving server-side Web page markup for deep annotation.

- **Ontology Evolution and Annotation**: There exists a tight inter linkage between evolving ontologies and Semantic Annotation of documents. In a realistic Semantic Web scenario, incoming information that is to be annotated does not only require more annotating, but also continuous adaptation to new semantic terminology and relationships. Stojanovic et. al. (cf., [Stojanovic *et al.*, 2002b]) gives first insights in this topic. However, further research must elaborate in great detail how ontology re-visioning may influence Semantic Annotation and the appropriate strategies to cope with that.

- **Multimedia Annotation**: Multimedia annotation requires a different approach to content and semantics, e.g. one has to deal with the video or audio content given in multimedia standards. One challenge here is to combine multimedia standards (like, MPEG-7, MPEG-21, SMIL) with recently established Semantic Web technologies (RDF, OWL) to describe explicit relational metadata with formal semantics. The semantic description needs to represent aspects of time and space. Another research challenge is semi-automatic annotation of multimedia content which needs a transformation from relatively easily extractable low-level audiovisual features (e.g. color, shapes, movements) into high-level semantic concepts (e.g. goalkeeper, score a goal) represented by a domain ontology (e.g. football ontology). This research question is currently tackled by the 6th Framework EU IST project "aceMedia"[3].

11.5. Summary

Documents created by Semantic Annotation bring the advantages of semantic search and interoperability. These benefits, however, come at the cost of an increased authoring effort. In this thesis we have, therefore, presented a comprehensive framework which support users in dealing with the documents, the ontologies and the annotations that link documents to ontologies.

Future research challenges include further improvements to automatic annotation components, such as relation extraction, and developing support systems for ontology evolution. There are also important human computer interaction challenges inherent in building integrated systems of this complexity.

[3] http://www.acemedia.org/

Part V.

Appendix

A. Metadata Standards

Coverage	Content	Scope of the content of the resource, e.g. location, time period or jurisdiction
Description	Content	Content of the resource
Relation	Content	Reference to a related resource
Source	Content	Reference to a resource from which the present resource is derived from
Subject	Content	Topic of the resource
Title	Content	Name given to resource
Type	Content	Nature of the content of the resource
Contributor	Intellectual Property	Name of institutions and persons contributing to resource
Creator	Intellectual Property	Name of creator of resource
Publisher	Intellectual Property	Name of entity responsible for making the resource available
Rights	Intellectual Property	Information about rights held in and over the resource
Date	Instantiation	Data associated with an event in the life cycle of the resource recommended for the encoding is ISO 8601
Format	Instantiation	Physical or digital manifestation of the resource recommended is to follow http://www.isi.edu/in-notes/iana/assignments/media-types/media-types
Identifier	Instantiation	unambiguous reference to the resource within a given context, e.g. URI
Language	Instantiation	Language the resource is in, according to RFC1766 (obsuleted by RFC3066)

Table A.1.: Dublin Core Elements

B. Glossary

Overview

an alphabetical list of technical terms in some specialized field of knowledge; usually published as an appendix to a text on that field

- *Annotation*: a comment or instruction (usually added); "his notes were appended at the end of the article"; "he added a short notation to the address on the envelope". *WordNet (r) 2.0*
- *HTML*: A markup language that is a subset of SGML and is used to create hypertext and hypermedia documents on the World Wide Web incorporating text, graphics, sound, video, and hyperlinks. *Merriam-Webster Online*
- *Markup*: In computerised document preparation, a method of adding information to the text indicating the logical components of a document, or instructions for layout of the text on the page or other information which can be interpreted by some automatic system. *The Free On-line Dictionary of Computing*
- *Metadata*: Metadata is data about data. A good example is a library catalog card, which contains data about the nature and location of a book: it is data about the data in the book referred to by the card.
- *Ontology*:
  - *Artificial intelligence*: (From philosophy) An explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them.

    For AI systems, what "exists" is that which can be represented. When the knowledge about a domain is represented in a declarative language, the set of objects that can be represented is called the universe of discourse. We can describe the ontology of a program by defining a set of representational terms. Definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions or other objects) with human-readable text describing what the names

mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory.

A set of agents that share the same ontology will be able to communicate about a domain of discourse without necessarily operating on a globally shared theory. We say that an agent commits to an ontology if its observable actions are consistent with the definitions in the ontology. The idea of ontological commitment is based on the Knowledge-Level perspective. *The Free On-line Dictionary of Computing*

- *Information science*: The hierarchical structuring of knowledge about things by subcategorising them according to their essential (or at least relevant and/or cognitive) qualities. See subject index. This is an extension of the previous senses of "ontology" (above) which has become common in discussions about the difficulty of maintaining subject indices. *The Free On-line Dictionary of Computing*

- *Relational Metadata*: Metadata is data about data. A good example is a library catalog card, which contains data about the nature and location of a book: it is data about the data in the book referred to by the card.

- *Semantic*: In general, Semantics (from the Greek semantikos, or "significant meaning," derived from "sema," sign) refers to the study of meaning, in some sense of that term. Semantics is often opposed to syntax, in which case the former pertains to what something means while the latter pertains to the formal structure/patterns in which something is expressed (e.g. written or spoken). *From Wikipedia, the free encyclopedia*

- *Semantic Annotation*: The term Semantic Annotation describes a process as well as the outcome of the process (cf. with term "drawing"). Hence it describes i) the process of addition of semantic information or metadata to the content given an agreed ontology and ii) it describes the semantic metadata itself as a result of this process.

- *Semantic Web*:

  - The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [Berners-Lee *et al.*, 2001]
  - A term coined by Tim Berners-Lee which views the future Web as a web of data, like a global database. The infrastructure of the Semantic Web would allow machines as well as humans to make deductions and organize information. The architectural components include semantics (meaning of the elements), structure (organization of the elements),

and syntax (communication). *from* http://www.w3.org/DesignIssues/Semantic.html

- *Software Agent*: In computer science, a software agent is a piece of autonomous, or semi-autonomous proactive and reactive, computer software. Many individual communicative software agents may form a multi-agent system. *From Wikipedia, the free encyclopedia*

- *URI* A Uniform Resource Identifier (URI), is an Internet protocol element consisting of a short string of characters that conform to a certain syntax. The string indicates a name or address that can be used to refer to an abstract or physical resource.

  URIs are a superset of the more commonly-known Uniform Resource Locator used for website addressing. A URI can be classified as a locator, a name, or both. URLs are the subset of URIs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location"). *From Wikipedia, the free encyclopedia*

- *URL* Uniform Resource Locator, URL (pronounced as "earl"), or Web address, is a standardized address for some resource (such as a document or image) on the Internet. URLs are classified by the "scheme" which typically identifies the network protocol used to retrieve the resource over a computer network. *From Wikipedia, the free encyclopedia*

- *World Wide Web*: (the "Web" or "WWW" for short) is a hypertext system that operates over the Internet. Hypertext is browsed using a program called a Web browser which retrieves pieces of information (called "documents" or "Web pages") from Web servers (or "Web sites") and displays them on your screen. You can then follow hyperlinks on each page to other documents or even send information back to the server to interact with it. The act of following hyperlinks is often called "surfing" the Web. *From Wikipedia, the free encyclopedia*

Bibliography

[Agarwal *et al.*, 2003] S. Agarwal, S. Handschuh, and S. Staab. Surfing the Service Web. In Fensel et al. [2003], pages 211–226.

[Agirre *et al.*, 2000] E. Agirre, O. Ansa, E. Hovy, and D. Martinez. Enriching Very Large Ontologies Using the WWW. In *Proceedings of the First Workshop on Ontology Learning OL'2000 Berlin, Germany, August 25, 2000*, 2000. Held in Conjunction with the 14th European Conference on Artificial Intelligence ECAI'2000, Berlin, Germany.

[Alfonseca and Manandhar, 2002] E. Alfonseca and S. Manandhar. Extending a Lexical Ontology by a Combination of Distributional Semantics Signatures. In Gómez-Pérez and Benjamins [2002], pages 1–7.

[Ankolekar *et al.*, 2002] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. McDermott, D. Martin, S. A. McIlraith, S. Narayanan, M. Paolucci, and T. Payne. DAML-S: Web Service Description for the Semantic Web. In Hendler and Horrocks [2002].

[Antoniou and van Harmelen, 2004] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, Handbooks in Information Systems, pages 67–92. Springer, 2004.

[Apache, 2004] Apache. Web Services Invocation Framework, 2004.

[Appelt *et al.*, 1993] D. Appelt, J. Hobbs, J. Bear, D. Israel, M. Kameyama, and M. Tyson. FASTUS: A Finite State Processor for Information Extraction from Real World Text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 1993.

[Asher and Lascarides, 1999] N. Asher and A. Lascarides. Bridging. *Journal of Semantics*, 15:83–113, 1999.

[Bechhofer and Goble, 2001] S. Bechhofer and C. Goble. Towards Annotation using DAML+OIL. In *Proceedings of the Knowledge Markup and Semantic Annotation Workshop 2001 (at K-CAP 2001)*, pages 13–20, Victoria, BC, Canada, October 2001.

[Bechhofer *et al.*, 2002] S. Bechhofer, L. Carr, C. Goble, S. Kampa, and T. Miles-Board. The Semantics of Semantic Annotation. In *ODBASE: First International Conference on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, pages 1152 – 1167, Irvine, California, October 2002.

[Benjamins *et al.*, 1998] V. R. Benjamins, E. Plaza, E. Motta, D. Fensel, R. Studer, B. Wielinga, G. Schreiber, and Z. Zdrahal. IBROW3 - An intelligent brokering service for knowledge-component reuse on the World Wide Web. In *Proc.11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW98)*, 1998.

[Benjamins *et al.*, 1999] V. R. Benjamins, D. Fensel, and S. Decker. KA2: Building Ontologies for the Internet: A Midterm Report. *International Journal of Human Computer Studies*, 51(3):687–713, 1999.

[Bergamaschi *et al.*, 2001] S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic Integration of Heterogeneous Information Sources. In *Special Issue on Intelligent Information Integration, Data & Knowledge Engineering*, volume 36, pages 215–249. Elsevier Science B.V., 2001.

[Berners-Lee *et al.*, 2001] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, pages 30–37, Mai 2001.

[Blythe and Gil, 2004] J. Blythe and Y. Gil. Incremental Formalization of Document Annotations through Ontology-Based Paraphrasing. In Feldman et al. [2004], pages 17–22.

[Bray *et al.*, 2004] T. Bray, J. Paoli, and C.M. Sperberg-McQueen. Extensible markup language (XML) 1.0. Technical report, W3C, 2004. http://www.w3.org/TR/REC-xml.

[Brickley and Guha, 2004] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C, Feb. 2004. W3C Working Draft. http://www.w3.org/TR/rdf-schema/.

[Broekstra *et al.*, 2001] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling Knowledge Representation on the Web by Extending RDF Schema. In *Proceedings of WWW 2001*, pages 467–478. ACM Press, 2001.

[Carletta, 1996] J. Carletta. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22(2):249–254, 1996.

[Charniak and Berland, 1999] E. Charniak and M. Berland. Finding Parts in Very Large Corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.

[Chen *et al.*, 2003] Y.-F.R. Chen, L. Kovács, and S. Lawrence, editors. *WWW-2002 — Proceedings of the 12th International Conference on the World Wide Web*. ACM Press, 2003. Budapest, Hungary, May, 2003.

[Cimiano and Handschuh, 2003] P. Cimiano and S. Handschuh. Ontology-based Linguistic Annotation. In *Proceedings of the Workshop on Linguistic Annotation at the ACL*, 2003.

[Cimiano *et al.*, 2004] P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In Feldman et al. [2004], pages 462–471.

[Cimiano, 2003] P. Cimiano. Ontology-driven discourse analysis in GenIE. In *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems*, pages 77–90, 2003.

[Ciravegna *et al.*, 2003] F. Ciravegna, A. Dingli, D. Guthrie, and Y. Wilks. Integrating Information to Bootstrap Information Extraction from Web Sites. In *IJCAI 2003 Workshop on Information Integration on the Web, Workshop in Conjunction with the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, August, 9-15*, pages 9–14, 2003.

[Ciravegna, 2001a] F. Ciravegna. Adaptive Information Extraction from Text by Rule Induction and Generalisation. In Bernhard Nebel, editor, *Proceedings of the seventeenth International Conference on Artificial Intelligence (IJCAI-01)*, pages 1251–1256, San Francisco, CA, August 4-10 2001. Morgan Kaufmann Publishers, Inc.

[Ciravegna, 2001b] F. Ciravegna. Challenges in Information Extraction from Text for Knowledge Management. *IEEE Intelligent Systems and their Applications*, 16(6):88–90, 2001.

[Ciravegna, 2001c] F. Ciravegna. $(LP)^2$, an Adaptive Algorithm for Information Extraction from Web-Related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining held in Conjunction with 17th International Joint Conference on Artificial Intelligence (IJCAI)*, Seattle, USA, August 2001.

[Clark and DeRose, 1999] J. Clark and S. DeRose. XML Path Language (XPath), W3C Recommendation 16 November 1999, 1999. http://www.w3.org/TR/xpath.

[Clauß and Ebner, 1995] G. Clauß and H. Ebner. *Grundlagen der Statistik für Psychologen, Pädagogen und Soziologen.* Harri Deutsch Verlag, Januar 1995.

[Cohen, 1998] W. Cohen. The WHIRL Approach to Data Integration. *IEEE Intelligent Systems*, pages 1320–1324, 1998.

[Collier *et al.*, 2004] N. Collier, A. Kawazoe, A. A. Kitamoto, T. Wattarujeekrit, T. Y. Mizuta, and A. Mullen. Integrating Deep and Shallow Semantic Structures in Open Ontology Forge. *Special Interest Group on Semantic Web and Ontology. JSAI (Japanese Society for Artificial Intelligence)*, 2004.

[Collier, 2001] N. Collier. Machine Learning for Information Extraction from XML Markup-up Text on the Semantic Web. In *Proceedings of the Semantic Web Workshop at the Tenth International Conference on the World Wide Web (WWW'10)*, pages 22–36, Hong Kong, May 2001.

[Constantopoulos *et al.*, 2000] P. Constantopoulos, V. Christophides, and D. Plexousakis, editors. *SemWeb 2000 — Proceedings of the First International Workshop on the Semantic Web*, 2000. Workshop at ECDL-2000, Lisbon, Portugal, September, 2000.

[Critchlow *et al.*, 1998] T. Critchlow, M. Ganesh, and R. Musick. Automatic Generation of Warehouse Mediators Using an Ontology Engine. In *Proceedings of the 5 th International Workshop on Knowledge Representation meets Databases (KRDB'98)*, pages 8.1–8.8, 1998.

[Davies *et al.*, 1998] S. Davies, M. Poesio, F. Bruneseaux, and L. Romary. *Annotating Coreference in Dialogues: Proposal for a Scheme for MATE*, 1998. http://www.cogsci.ed.ac.uk/p̃oesio/anno_manual.html.

[de Cea *et al.*, 2002] G. Aguado de Cea, I. varez de Mon, A. Gez-P ez, A. Pareja-Lora, and R. Plaza-Arteche. A Semantic Web Page Linguistic Annotation Model. In *Semantic Web Meets Language Resources. Papers from the AAAI Workshop*, 2002. http://www.cs.vassar.edu/ ide/events/ParejaLora.pdf.

[Decker *et al.*, 1998] S. Decker, D. Brickley, J. Saarela, and J. Angele. A Query and Inference Service for RDF. In *Proceedings of the W3C Query Language Workshop (QL-98), Boston, MA, December 3-4*, 1998. http://www.w3.org/TandS/QL/QL98/.

[Decker *et al.*, 1999] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.

[Decker, 2002] S. Decker. *Semantic Web Methods for Knowledge Management*. PhD thesis, University of Karlsruhe, 2002.

[Decker, 2004] S. Decker. Semantic Web Community Portal. Web page, 2004. http://www.semanticweb.org.

[Denoue and Vignollet, 2000] L. Denoue and L. Vignollet. An Annotation Tool for Web Browsers and its Applications to Information Retrieval. In *Proceedings of RIAO2000*, Paris, April 2000. http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/riao2000.doc.

[DeRose *et al.*, 2001] S. DeRose, R. Daniel Jr., E. Maler, J. Marh, and N. Walsh. XML Pointer Language (XPointer). Technical report, W3C, 2001. Working Draft 16 August 2002.

[DeRose *et al.*, 2002] S. DeRose, E. Maler, and R. Daniel Jr. XPointer xpointer() Scheme, 2002. http://www.w3.org/TR/xptr-xpointer/.

[DeRose *et al.*, 2003] S. DeRose, R. Daniel Jr., E. Maler, and J. Marsh. XPointer xmlns() Scheme, 2003. http://www.w3.org/TR/xptr-xmlns/.

[DeRoure and Iyengar, 2002] D. DeRoure and A. Iyengar, editors. *WWW-2002 — Proceedings of the 11th International Conference on the World Wide Web*. ACM Press, 2002. Honolulu, Hawaii, May, 2002.

[Dill *et al.*, 2003] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. In *Proceedings of the 12th International Conference on World Wide Web*, pages 178–186. ACM Press, 2003.

[Doan *et al.*, 2002] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In DeRoure and Iyengar [2002], pages 662–673. Honolulu, Hawaii, May, 2002.

[Douthat, 1998] A. Douthat. The Message Understanding Conference Scoring Software User's Manual. In *7th Message Understanding Conference Proceedings, MUC-7*, 1998. http://www.itl.nist.gov/iaui/894.02/related_projects/muc/.

[dub, 2001] Dublin Core Metadata Initiative, April 2001. http://purl.oclc.org/dc/.

[Dublin Core Metadata Template, 2001] Dublin Core Metadata Template, 2001. http://www.ub2.lu.se/metadata/DC_creator.html.

[Ellis and Nutt, 1993] C.A. Ellis and G.J. Nutt. Modelling and enactment of Workflow Systems. *Application and Theory of Petri Nets*, LNCS 691:Modelling and enactment of workflow systems, 1993.

[Ellis *et al.*, 1994] D. Ellis, J. Furner-Hines, and P. Willett. On the Measurement of Inter-Linker Consistency and Retrieval Effectiveness in Hypertext

Databases. In *Proceedings of the 17th annual International ACM SIGIR Conference on Research and Development in Information*, pages 51–60, Berlin, 1994. Springer.

[Erdmann *et al.*, 2000] M. Erdmann, A. Maedche, H.-P. Schnurr, and S. Staab. From Manual to Semi-Automatic Semantic Annotation: About Ontology-Based Text Annotation Tools. In *P. Buitelaar & K. Hasida (eds). Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, Luxembourg, August 2000.

[Eriksson *et al.*, 1999] H. Eriksson, R. Fergerson, Y. Shahar, and M. Musen. Automatic Generation of Ontology Editors. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modelling and Management (KAW'99), Banff, Canada, October*, 1999.

[Euzenat, 2002] J. Euzenat. Eight Questions about Semantic Web Annotations. *IEEE Intelligent Systems*, 17(2):55–62, Mar/Apr 2002.

[F-Logic Tutorial, 2004] How to Write F-Logic Programs. A Tutorial for the Language F-Logic, (covers OntoBroker Version 3.7.x-3.8.x), 2004. http://www.ontoprise.de/documents/tutorial_flogic.pdf.

[Farquhar *et al.*, 1996] A. Farquhar, R. Fickas, and J. Rice. The Ontolingua Server: A Tool for Collaborative Ontology Construction. In *Proceedings of the 10th Banff Knowledge Acquisition for KnowledgeBased System Workshop (KAW'96)*, Banff, Canada, November 1996.

[Feldman *et al.*, 2004] Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors. *Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA, May 17-20, 2004*. ACM, 2004.

[Fensel *et al.*, 1999] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, and A. Witt. On2broker: Semantic-Based Access to Information Sources at the WWW. In *Proceedings of the World Conference on the WWW and Internet (WebNet 99), Honolulu, Hawaii, USA*, pages 366–371, 1999.

[Fensel *et al.*, 2002] D. Fensel, H. Lieberman, J. Hendler, and W. Wahlster, editors. *Spinning the Semantic Web*, Cambridge, MA, USA, 2002. MIT Press.

[Fensel *et al.*, 2003] D. Fensel, K. P. Sycara, and J. Mylopoulos, editors. *ISWC-2003 — Proceedings of the Second International Semantic Web Conference*, LNCS 2870. Springer, 2003.

[Fensel, 2001] D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, Berlin - Heidelberg - New York, 2001.

[Fillies and Weichhardt, 2002] C. Fillies and F. Weichhardt. Graphische Entwicklung und Nutzung von Ontologien mit SemTalk in MS Office. In R. Tolksdorf and R. Eckstein, editors, *XML Technologien für das Semantic Web — XSW 2002*, GI-Edition — Lecture Notes in Informatics (LNI), P-14. Bonner Köllen Verlag, 2002. Workshop 24. – 25. Juni 2002 in Berlin.

[Flake *et al.*, 2002] G.W. Flake, S. Lawrence, C.L. Giles, and F.M. Coetzee. Self-Organization and Identification of Web Communities. *IEEE Computer*, 35(3):66–70, March 2002.

[Fleischman and Hovy, 2002] M. Fleischman and E. Hovy. Fine Grained Classification of Named Entities. In *Proceedings of the Conference on Computational Linguistics, Taipei, Taiwan, August 2002*, 2002.

[Fligelstone, 1992] S. Fligelstone. Developing a Scheme for Annotating Text to Show Anaphoric Relations. In G. Leitner, editor, *New Directions in Corpus Linguistics*, pages 153–170. Mouton de Gruyter, 1992.

[Frank *et al.*, 2002a] M. Frank, N. Fridman-Noy, and S. Staab, editors. *Proceedings of the International Workshop on the Semantic Web*, CEUR Workshop Proceedings Vol. 55. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-55/, 2002. Workshop at WWW-2002, Honolulu, Hawaii, USA, May, 2002.

[Frank *et al.*, 2002b] M. Frank, P. Szekely, R. Neches, B. Yan, and J. Lopez. WebScripter: World-Wide Grass-Roots Ontology Translation via Implicit End-User Alignment. In M. Frank, N. Noy, and S. Staab, editors, *Proceedings of the Semantic Web Workshop 2002 (at WWW-2002)*, CEUR Proceedings, Volume 55, http://www.ceur-ws.org, pages 22–28, 2002.

[Genesereth and Fikes, 1992] M.R. Genesereth and R.E. Fikes. Knowledge Interchange Format Version 3.0 Reference Manual. Report Logic 92-1, Stanford Logic Group, June 1992. http://logic.stanford.edu/sharing/papers/kif.ps.

[Gil and Ratnakar, 2002] Y. Gil and V. Ratnakar. Trellis: an interactive tool for capturing information analysis and decision making. In Gómez-Pérez and Benjamins [2002], pages 37–42.

[Glover *et al.*, 2002] E.J. Glover, K. Tsioutsiouliklis, S. Lawrence, D.M. Pennock, and G.W. Flake. Using Web Structure for Classifying and Describing Web Pages. In *Proceedings of the Eleventh International Conference on World Wide Web*, pages 562–569. ACM Press, 2002.

[Goble *et al.*, 2001] C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual Open Hypermedia = The Semantic Web? In S. Staab, S. Decker, D. Fensel, and A. Sheth, editors, *The Second International Workshop on*

the Semantic Web, CEUR Proceedings, Volume 40, http://www.ceur-ws.org, pages 44–50, Hong Kong, May 2001.

[Gómez-Pérez and Benjamins, 2002] Asunción Gómez-Pérez and V. Richard Benjamins, editors. *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, volume 2473 of *Lecture Notes in Computer Science*. Springer, 2002.

[Googlism, 2003] Googlism, 2003. http://www.googlism.com.

[Grefenstette, 1999] G. Grefenstette. The WWW as a Resource for Example-Based MT Tasks. In *Proceedings of ASLIB'99 Translating and the Computer 21*, 1999.

[Grosso *et al.*, 2002] P. Grosso, E. Maler, J. Marsh, and N. Walsh. XPointer Framework, W3C Recommendation 25 March 2003, 2002. http://www.w3.org/TR/2003/REC-xptr-framework-20030325/.

[Grosso *et al.*, 2003] P. Grosso, E. Maler, J. Marsh, and N. Walsh. XPointer element() Scheme, 2003. http://www.w3.org/TR/xptr-element/.

[Grosz and Sidner, 1986] B.J. Grosz and C.L. Sidner. Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[Guarino, 1998] N. Guarino. Formal Ontology and Information Systems. 1998.

[Hahn and Schnattinger, 1998] U. Hahn and K. Schnattinger. Towards Text Knowledge Engineering. In *AAAI'98/IAAI'98 Proceedings of the 15th National Conference on Artificial Intelligence and the 10th Conference on Innovative Applications of Artificial Intelligence*, pages 524–531, 1998.

[Handschuh and Staab, 2002] S. Handschuh and S. Staab. Authoring and Annotation of Web Pages in CREAM. In DeRoure and Iyengar [2002], pages 462–473. Honolulu, Hawaii, May, 2002.

[Handschuh and Staab, 2003] S. Handschuh and S. Staab, editors. *Annotation in the Semantic Web*. IOS Press, 2003.

[Handschuh *et al.*, 2001] S. Handschuh, S. Staab, and A. Maedche. CREAM — Creating Relational Metadata with a Component-Based, Ontology-Driven Annotation Framework. In *Proceedings of K-Cap 2001*, pages 76–83. ACM Press, 2001.

[Handschuh *et al.*, 2002] S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM — Semi-Automatic CREAtion of Metadata. In Gómez-Pérez and Benjamins [2002], pages 358–372.

[Handschuh *et al.*, 2003] S. Handschuh, S. Staab, and R. Volz. On Deep Annotation. In Chen et al. [2003], pages 431–438. Budapest, Hungary, May, 2003.

[Hearst, 1992] M.A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, 1992.

[Heflin and Hendler, 2000a] J. Heflin and J. Hendler. Dynamic Ontologies on the Web. In *AAAI-2000 — Proceedings of the National Conference on Artificial Intelligence. Austin, TX, USA, August 2000*, 2000.

[Heflin and Hendler, 2000b] J. Heflin and J. Hendler. Searching the Web with SHOE. In *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, pages 35–40. AAAI Press, 2000.

[Hendler and Horrocks, 2002] J. Hendler and I. Horrocks, editors. *ISWC-2002 — Proceedings of the First International Semantic Web Conference*, LNCS 2342. Springer, 2002.

[Hillmann, 2001] D. Hillmann. Using Dublin Core, 2001. http://dublincore.org/documents/2001/04/12/usageguide/.

[Hirschman and Chinchor, 1997] L. Hirschman and N. Chinchor. Muc-7 coreference task definition. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, 1997.

[Horrocks, 1998] I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998*, pages 636–649. Morgan Kaufmann, 1998.

[Ide, 2002] N. Ide. Linguistic Annotation Framework. Technical Report ISO TC37/SC4/WG1 N11, ISO TC37 SC4 Language Resource Management, 08 2002.

[Jasper and Uschold, 1999] R. Jasper and M. Uschold. A Framework for Understanding and Classifying Ontology Applications. Banff, Canada, 1999. http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html.

[Jin *et al.*, 2001] Y. Jin, S. Decker, and G. Wiederhold. OntoWebber: Model-Driven Ontology-Based Web Site Management. In *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA, August 2001.

[Kahan *et al.*, 2001] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 623–632. ACM Press, 2001.

[Kashyap and Sheth, 1996] V. Kashyap and A. Sheth. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. In M. Papazoglou and G. Schlageter, editors, *Cooperative Information Systems: Current Trends and Directions*. Academic Press, 1996.

[Keller *et al.*, 2002] F. Keller, M. Lapata, and O. Ourioupina. Using the Web to Overcome Data Sparseness. In *Proceedings of EMNLP-02*, pages 230–237, 2002.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object oriented and frame-based languages. *Journal of the ACM*, 42(4):741–843, 1995.

[Klarity, 2001] Klarity – Automatic Generation of Metadata Based on Concepts within the Document, 2001. Klarity White Paper. http://www.klarity.com.au.

[Kogut and Holmes, 2001] P. Kogut and W. Holmes. AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. 2001.

[Krahmer and Piwek, 2000] E. Krahmer and P. Piwek. Varieties of Anaphora: Introduction. In *ESSLLI 2000 Reader*. 2000.

[Kunze, 1999] J. Kunze. Encoding Dublin Core Metadata in HTML, 1999. http://www.ietf.org/rfc/rfc2731.txt.

[Kushmerick, 1997] N. Kushmerick. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[Kushmerick, 2000] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[Lascarides and Asher, 1991] A. Lascarides and N. Asher. Discourse Relations and Defeasible Knowledge. In *Meeting of the Association for Computational Linguistics*, pages 55–62, 1991.

[Leech, 1997] G. Leech. Introducing Corpus Annotation. In R. Garside, G. Leech, and A.M. McEnery, editors, *Corpus Annotation: Linguistic Information from Computer Text Corpora*, 1997.

[Lehnert *et al.*, 1994] W.G. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. Evaluating an Information Extraction System. *Journal of Integrated Computer-Aided Engineering*, 1(6), 1994.

[Lei *et al.*, 2002] Y. Lei, E. Motta, and J. Domingue. An Ontology-Driven Approach to Web Site Generation and Maintenance. In Gómez-Pérez and Benjamins [2002].

[Lenat and Guha, 1990] D.B. Lenat and R.V. Guha. *Building Large Knowledge-Based Systems. Representation and Inference in the CYC Project.* Addison-Wesley, Reading, Massachusetts, 1990.

[Leonard, 1977] L. Leonard. Inter-Indexer Consistence Studies, 1954-1975: A Review of the Literature and Summary of the Study Results. Graduate School of Library Science, University of Illinois. Occasional Papers No.131, 1977.

[Levenshtein, 1966] I.V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[Lowe, 2001] D. et al. Lowe. *BizTalk(TM) Server: The Complete Reference.*, November 2001.

[Luke *et al.*, 1997] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-Based Web Agents. In *Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, USA, February 5-8, 1997*, pages 59–66, 1997.

[MacGregor, 1991] R. MacGregor. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, 1991.

[Madhavan *et al.*, 2001] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *Proceedings of the 27th International Conferences on Very Large Databases*, pages 49–58, 2001.

[Maedche and Staab, 2001] A. Maedche and S. Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72–79, March/April 2001.

[Maedche and Staab, 2003] A. Maedche and S. Staab. Services on the Move — Towards P2P-Enabled Semantic Web Services. In *Proceedings of the 10th International Conference on Information Technology and Travel & Tourism, ENTER 2003, Helsinki, Finland, 29th-31st January 2003.* Springer, 2003.

[Maedche and Zacharias, 2002] A. Maedche and V. Zacharias. Clustering Ontology-based Metadata in the Semantic Web. In *Proceedings of the Joint Conferences 13th European Conference on Machine Learning (ECML'02) and 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02).* Springer, 2002.

[Maedche *et al.*, 2002] A. Maedche, B. Motik, N. Silva, and R. Volz. MAFRA - A Mapping Framework for Distributed Ontologies. In *Proceedings of EKAW 2002*, LNCS 2473, pages 235–250. Springer, 2002.

[Manola and Miller, 2004] F. Manola and E. Miller. RDF Primer, Feb. 2004. http://www.w3.org/TR/rdf-primer/.

[Marcus *et al.*, 1993] M. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[Markert *et al.*, 2003] K. Markert, N. Modjeska, and M. Nissim. Using the Web for Nominal Anaphora Resolution. In *EACL Workshop on the Computational Treatment of Anaphora*, 2003.

[Martin and Eklund, 1999] P. Martin and P. Eklund. Embedding Knowledge in Web Documents. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*, pages 1403–1419. Elsevier Science B.V., 1999.

[Maynard *et al.*, 2002] D. Maynard, V. Tablan, H. Cunningham, C. Ursu, H. Saggion, K. Bontcheva, and Y. Wilks. Architectural Elements of Language Engineering Robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*, 2002. forthcoming.

[McGuinness *et al.*, 2000] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. The Chimaera Ontology Environment. In *Proc. of AAAI-2000*, pages 1123–1124, 2000.

[McIlraith and Son, 2002] S. McIlraith and T. Son. Adapting Golog for composition of semantic Web services. In *Proc 8th International Conference on Principles of Knowledge Representation and Reasoning*, 2002.

[Mickalski *et al.*, 1986] R.S. Mickalski, I. Mozetic, J. Hong, and H. Lavrack. The Multi Purpose Incremental Learning System AQ15 and its Testing Application to three Medical Domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, USA, 1986.

[Mitra *et al.*, 2000] P. Mitra, G. Wiederhold, and M. Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Proceedings of Conference on Extending Database Technology (EDBT 2000)*. Konstanz, Germany, 2000.

[MUC7, 1998] *MUC-7 — Proceedings of the 7th Message Understanding Conference*, 1998. http://www.muc.saic.com/.

[Müller and Strube, 2001] C. Müller and M. Strube. Annotating Anaphoric and Bridging Relations with MMAX. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, pages 90–95, 2001.

[Narayanan and McIlraith, 2002] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In DeRoure and Iyengar [2002]. Honolulu, Hawaii, May, 2002.

[Nejdl *et al.*, 2001] W. Nejdl, B. Wolf, S. Staab, and J. Tane. EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network. Technical report, Learning Lab Lower Saxony / Institute AIFB, 2001.

[Nejdl *et al.*, 2002] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. EDUTELLA: A P2P Networking Infrastructure Based on RDF. In DeRoure and Iyengar [2002], pages 604–615. Honolulu, Hawaii, May, 2002.

[Noy and McGuinness, 2001] N.F. Noy and D.L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics, 2001.

[Noy and Musen, 2000] N. F. Noy and M. A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. of AAAI-2000*, pages 450–455, 2000.

[Noy *et al.*, 2000] N.F. Noy, W.E. Grosso, and M.A. Musen. Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protege-2000. In *Proceedings of the 12th Internal Conference on Software and Knowledge Engineering. Chicago, USA, July, 5-7, 2000*, 2000.

[OWL Reference, 2004] OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref, 2004.

[Paolucci *et al.*, 2002] M. Paolucci, T. Kawmura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *First Int. Semantic Web Conf.*, 2002.

[Papakonstantinou and Vassalos, 2002] Y. Papakonstantinou and V. Vassalos. Architecture and Implementation of an XQuery-based Information Integration Platform. *IEEE Data Engineering Bulletin*, 25(1):18–26, 2002.

[Park *et al.*, 1997] J. Y. Park, J. H. Gennari, and M. A. Musen. Mappings for Reuse in Knowledge-based Systems. In *Technical Report, SMI-97-0697, Stanford University*, 1997.

[Passoneau, 1996] R. Passoneau. *DRAMA: Instructions for Applying Reference Annotation for Multiple Applications*, 1996.

[Patil *et al.*, 2004] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEOR-S Web Services Annotation Framework. In Feldman et al. [2004].

[Phelps and Wilensky, 2000] T. Phelps and R. Wilensky. Robust Intra-Document Locations. *Proceedings of WWW9 / Computer Networks*, 33(1-6):105–118, 2000.

[Poesio and Reyle, 2001] M. Poesio and U. Reyle. Underspecification in Anaphoric Reference. In *Proceedings of the IWCS-4*, 2001.

[Poesio and Vieira, 1998] M. Poesio and R. Vieira. A Corpus-Based Investigation of Definite Description Use. *Computational Linguistics*, 24(2):183–216, 1998.

[Ponnekanti and Fox, 2002] S. R. Ponnekanti and A. Fox. SWORD: A Developer Toolkit for Building Composite Web Services. In DeRoure and Iyengar [2002]. Honolulu, Hawaii, May, 2002.

[Popov *et al.*, 2003] B. Popov, A Kiryakov, D. Ognyanoff, D. Manov, A. Kirilov, and M. Goranov. Towards Semantic Web Information Extraction. In Fensel et al. [2003].

[Rahm and Bernstein, 2001] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[Resnik and Smith, 2003] P. Resnik and N. Smith. The Web as a Parallel Corpus. *Computational Linguistics*, 29(3), 2003.

[Resnik, 1997] P. Resnik. Selectional Preference and Sense Disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, 1997.

[Rinaldi *et al.*, 2003] F. Rinaldi, J. Dowdall, M. Hess, J. Ellman, G. P. Zarri, A. Persidis, L. Bernard, and H. Karanikas. Multilayer annotations in Parmenides. In *Proceedings of the Knowledgte Markup and Semantic Annotation Workshop, Sanibel, Flodia , USA, October 26, 2003*, pages 33–40, 2003.

[Sahuguet and Azavant, 2001] A. Sahuguet and F. Azavant. Building intelligent Web applications using lightweight wrappers. *Data and Knowledge Engineering*, 3(36):283–316, 2001.

[Schmid, 1994] H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, 1994.

[Siegel, 2001] S. Siegel. *Nichtparametrische statistische Methoden.* Klotz, Eschborn, 2001.

[Sowa, 1984] J.F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.

[Staab and Maedche, 2001] S. Staab and A. Maedche. Knowledge Portals — Ontologies at Work. *AI Magazine*, 22(2):63–75, Summer 2001.

[Staab *et al.*, 2000a] S. Staab, J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure. Semantic Community Web Portals. *Proceedings of WWW9 / Computer Networks*, 33(1-6):473–491, 2000.

[Staab *et al.*, 2000b] S. Staab, M. Erdmann, A. Maedche, and S. Decker. An Extensible Approach for Modeling Ontologies in RDF(S). In *Proceedings of the ECDL-2000 Workshop "Semantic Web: Models, Architectures and Management", Lisbon, September 21, 2000*, 2000.

[Staab *et al.*, 2001a] S. Staab, S. Decker, D. Fensel, and A. Sheth, editors. *SemWeb 2001 — Proceedings of the Second International Workshop on the Semantic Web*, CEUR Workshop Proceedings Vol. 40. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-40/, 2001. Workshop at WWW-2001, Hong Kong, China, May 1, 2001.

[Staab *et al.*, 2001b] S. Staab, A. Maedche, and S. Handschuh. Creating Metadata for the Semantic Web: An Annotation Framework and the Human Factor. Technical Report 412, Institute AIFB, University of Karlsruhe, 2001.

[Staab *et al.*, 2001c] S. Staab, H.-P. Schnurr, R. Studer, and Y. Sure. Knowledge Processes and Ontologies. *IEEE Intelligent Systems*, 16(1), 2001.

[Stojanovic *et al.*, 2001] N. Stojanovic, A. Maedche, S. Staab, R. Studer, and Y. Sure. SEAL: A Framework for Developing SEmantic PortALs. In *Proceedings of K-CAP 2001*, pages 155–162. ACM Press, 2001.

[Stojanovic *et al.*, 2002a] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic. User-Driven Ontology Evolution Management. In Gómez-Pérez and Benjamins [2002], pages 285–300.

[Stojanovic *et al.*, 2002b] L. Stojanovic, N. Stojanovic, and S. Handschuh. Evolution of Metadata in Ontology-based Knowledge Management Systems. In *Experience Management 2002, German Workshop on Experience Management, Berlin*, March 7-8 2002.

[Stojanovic *et al.*, 2002c] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating Data-Intensive Web Sites into the Semantic Web. In *Proceedings of the ACM Symposium on Applied Computing SAC-02, Madrid, 2002*, pages 1100–1107. ACM Press, 2002.

[Strube and Hahn, 1999] M. Strube and U. Hahn. Functional Centering — Grounding Referential Coherence in Information Structure. *Computational Linguistics*, 25(3):309–344, 1999.

[Studer *et al.*, 2002] R. Studer, Y. Sure, and R. Volz. Managing User Focused Access to Distributed Knowledge. *Journal of Universal Computer Science (J.UCS)*, 8(6):662–672, 2002.

[Sure *et al.*, 2002a] Y. Sure, J. Angele, and S. Staab. Guiding Ontology Development by Methodology and Inferencing. In K. Aberer and L. Liu, editors, *ODBASE-2002 — Ontologies, Databases and Applications of Semantics. Irvine, CA, USA, Oct. 29-31, 2002*, LNCS, pages 1025–1222. Springer, 2002.

[Sure *et al.*, 2002b] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In Hendler and Horrocks [2002], pages 221–235.

[Sure, 2003] Y. Sure. *Methodology, Tools and Case Studies for Ontology based Knowledge Management*. PhD thesis, University of Karlsruhe, 2003.

[Sycara *et al.*, 1999] K. P. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic Service Matchmaking Among Agents in Open Information Environments. *SIGMOD Record*, 28(1):47–53, 1999.

[Tallis *et al.*, 2001] M. Tallis, N. Goldman, and R. Balzer. The Briefing Associate: A Role for COTS Applications in the Semantic Web. In *Semantic Web Working Symposium (SWWS)*, Stanford, California, USA, August 2001.

[van Deemter and Kibble, 2000] K. van Deemter and R. Kibble. On Coreferring: Coreference in MUC and Related Annotation Schemes. *Computational Linguistics*, 26(4), 2000.

[van der Aalst, 1999] W.M.P. van der Aalst. Woflan: A Petri-net-based workflow analyzer, Systems Analysis - Modelling - Simulation. *Systems Analysis - Modelling and Simulation*, 35(3):345–357, 1999.

[van Heijst, 1995] G. van Heijst. *The Role of Ontologies in Knowledge Engineering*. PhD thesis, Universiteit van Amsterdam, 1995.

[Vargas-Vera *et al.*, 2001] M. Vargas-Vera, E. Motta, J. Domingue, S. Buckingham Shum, and M. Lanzoni. Knowledge Extraction by Using an Ontology-Based Annotation Tool. In *Proceedings of the Knowledge Markup and Semantic Annotation Workshop 2001 (at K-CAP 2001)*, pages 5–12, Victoria, BC, Canada, October 2001.

[Vargas-Vera *et al.*, 2002] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In Gómez-Pérez and Benjamins [2002], pages 379–391.

[W3C, 2003] W3C. Web Service Description Language (WSDL) Version 1.2, March 2003.

[Wiederhold, 1993] G. Wiederhold. Intelligent integration of information. *Proceedings of the ACMSIGMOD International Conference on Management of Data*, pages 434–437, 1993.

[Yee, 1998] K.-P. Yee. CritLink: Better Hyperlinks for the WWW, 1998. http://crit.org/~ping/ht98.html.