S-CREAM — Semi-automatic CREAtion of Metadata

Siegfried Handschuh[1] and **Steffen Staab**[1] and **Fabio Ciravegna**[2]

Abstract. Richly interlinked, machine-understandable data constitute the basis for the Semantic Web. We provide a framework, S-CREAM, that allows for creation of metadata and is trainable for a specific domain. Annotating web documents is one of the major techniques for creating metadata on the web. The implementation of S-CREAM, OntoMat supports now the semi-automatic annotation of web pages. This semi-automatic annotation is based on the information extraction component Amilcare. OntoMat extract with the help of Amilcare knowledge structure from web pages through the use of knowledge extraction rules. These rules are the result of a learning-cycle based on already annotated pages.

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. To support the construction of relational metadata, we have provided an annotation [15] and authoring [16] framework (CREAM — manually CREAting Metadata) and a tool (OntoMat) that implements this framework. Nevertheless, providing plenty of relational metadata by annotation, i.e. conceptual mark-up of text passages, remained a laborious task.

Though there existed the high-level idea that wrappers and information extraction components could be used to facilitate the work [8, 15], a full-fledged integration that dealt with all the conceptual difficulties was still lacking. Therefore, we have developed S-CREAM (Semi-automatic CREAtion of Metadata), an annotation framework that integrates a learnable information extraction component (viz. Amilcare [1]).

Amilcare is a system that learns 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 are two major type of problems that we had to solve 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 4). In order to overcome this problem, we introduce a new processing component, viz. a lightweight module for discourse representation (Section 5).
2. Semantic tags do not correspond one-to-one to the conceptual description (Section 5 and 6).
   - 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.

In the remainder of the paper, we will first describe the existing frameworks, viz. CREAM (Section 2) and Amilcare (3). Second, we will focus on the integration problems (Section 4–5). Third, we will describe a usage scenario (Section 6). Eventually, we will discuss related works and conclude.

2 CREAM/OntoMat

CREAM is an annotation and authoring framework suited for the easy and comfortable creation of relational metadata. OntoMat is its concrete implementation. Before we sketch some of the capabilities of CREAM/OntoMat, we first describe its assumptions on its output representation and some terminology we use subsequently.

2.1 Relational Metadata

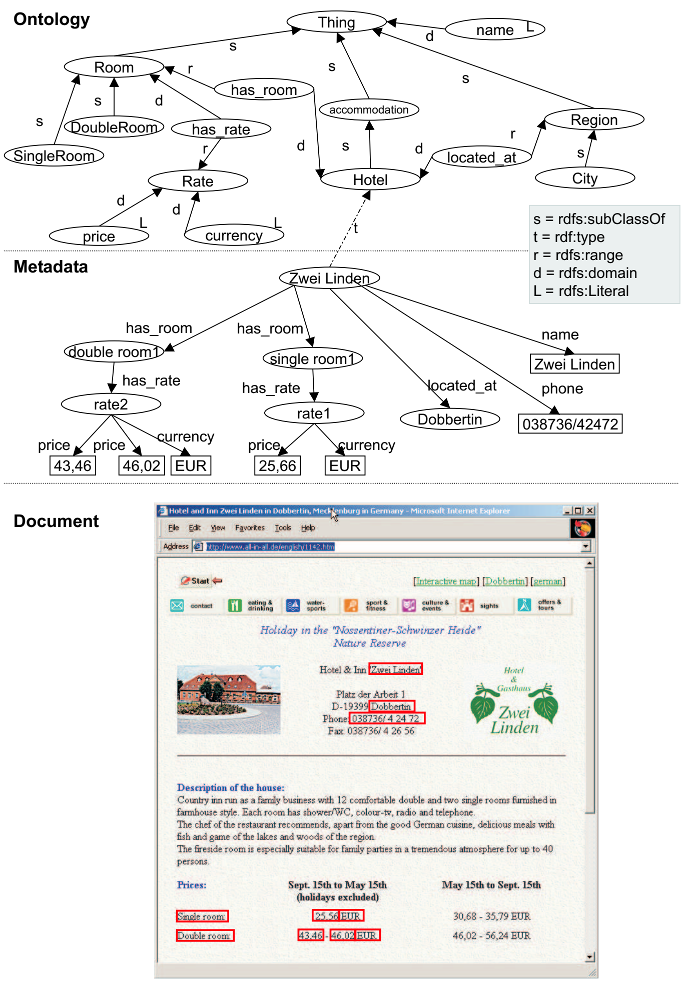
We elaborate the terminology here because many of the terms that are used with regard to metadata creation tools carry several, ambiguous connotations that imply conceptually important differences:

- **Ontology**: An ontology is a formal, explicit specification of a shared conceptualization of a domain of interest [13]. In our case it is constituted by statements expressing definitions of DAML+OIL classes and properties [11].
- **Annotations**: An annotation in our context is a set of instantiations attached to an HTML document. We distinguish *(i)* instantiations of DAML+OIL 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 'urn:rdf:936694d5ca907974ea16565de20c997a-0'.[3]
  They frequently come with attribute instances, such as a human-readable label like 'Dobbertin'.
- **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.

[1] AIFB, University of Karlsruhe, email: {sha,sst}@aifb.uni-karlsruhe.de
[2] Department of Computer Science, University of Sheffield, email: F.Ciravegna@dcs.shef.ac.uk

[3] In the OntoMat implementation we create the URIs with the createUniqueResource method of the RDF-API

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 modelling 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: The hotel "Zwei Linden" is located in the city "Dobbertin"., *i.e.* relational metadata contain relationships between class instances rather than only textual notes.



Holiday in the "Nossentiner-Schwinzer Heide"
Nature Reserve

Hotel & Inn 'Zwei Linden'

Platz der Arbeit 1
D-19399 Dobbertin
Phone: 038736/ 4 24 72
Fax: 038736/ 4 26 56

Description of the house:
Country inn run as a family business with 12 comfortable double and two single rooms furnished in farmhouse style. Each room has shower/WC, colour-tv, radio and telephone.
The chef of the restaurant recommends, apart from the good German cuisine, delicious meals with fish and game of the lakes and woods of the region.
The fireside room is especially suitable for family parties in a tremendous atmosphere for up to 40 persons.

Prices:	Sept. 15th to May 15th (holidays excluded)	May 15th to Sept. 15th
Single room:	25,56 EUR	30,68 - 35,79 EUR
Double room:	43,46 - 46,02 EUR	46,02 - 56,24 EUR

Figure 1. Annotation example

Figure 1 illustrates our use of the terms "ontology", "annotation" and "relational metadata". It depicts some part of a tourism ontology.[4] Furthermore it shows the homepage of the Hotel "Zwei Linden"(`http://www.all-in-all.de/ english/1142.htm`) annotated in RDF. For the hotel there is a instances denoted by corresponding URI (`urn:rdf:947794d5ca907974-ea16565de21c998a-0`). In addition, there is a relationship instance between the hotel and the city.

[4] currently only available in German at http://ontobroker.semanticweb.org/ontos/compontos/tourism_I1.daml

2.2 Modes of Interaction

The objective of CREAM is to allow for the easy generation of target representations such as just illustrated. This objective should be achieved irrespective of the mode of interaction. In the latest version of CREAM [16] there existed three major modes:

1. **Annotation by typing statements** involves working almost exclusively with the ontology browser and fact templates.
2. **Annotation by markup** involves reuse of data from the document editor in the ontology browser by first marking document parts and drag'n'dropping them onto the ontology.
3. **Annotation by authoring web pages** involves the reuse of data from the ontology and fact browser in the document editor by drag'n'drop.

OntoMat usually embeds the resulting annotation into the HTML document, but it can also be stored in a separate file or database.

3 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. 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 work in two modes: **training**, used to adapt 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 ([22], 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 [1] [3]. This is a wrapper induction methodology [19] 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 generalise 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. [3]). All the generalizations are tested in parallel by using a variant of the AQ algorithm [23] 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 recognising the location in holiday advertisements, but not to spot the hotel address. This strategy is quite effective for analysing documents with mixed genres, quite a common situation in web documents [2].

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 match also 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 to 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 and then it applies its rules and returns the original text with the added annotations. The Gate annotation schema is used for annotation [22].

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 [7] 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. Although the current interface for balancing P&R is designed for IE experts, we have plans for enabling also naive users [4].

4 Synthesizing S-CREAM

In order to synthesize S-CREAM out of the existing frameworks CREAM and Amilcare, we consider their core processes in terms of input and output, as well as the process of the yet undefined S-CREAM. Figure 2 surveys the three processes.

The first 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 sketched in Section 2 and described in detail in [16]) For instance, an annotator may use OntoMat to describe that on the homepage of hotel "Zwei Linden" (cf. Figure 1) the relationships listed in Table 1(a) show up.

The second process is indicated by a circled A1. It is information extraction, e.g. provided by Amilcare [1], which digests a document and produces either a XML tagged document or a list of XML tagged text snippets (cf. Table 1(b)).

The obvious questions that come up at this point are: Is the result of Table 1(b) equivalent to the one in Table 1(a)? How can Table 1(b) be turned into the result of Table 1(a)? The latter is a requirement for the Semantic Web.

The "Semantic Web answer" to this is: The difference between

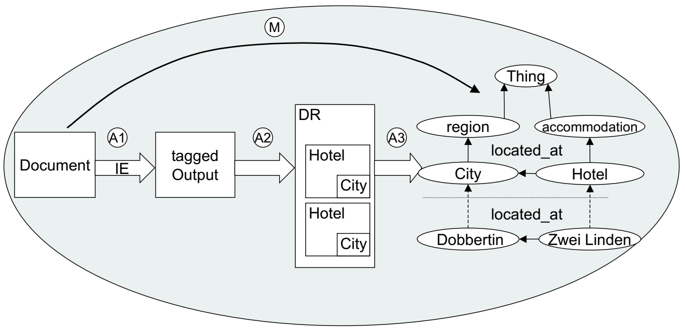


Figure 2. Two Ways to the Target: Manual and Automatic Annotation

Table 1(a) and Table 1(b) is analogous to the difference between an RDF structure and a very particular serialization of data in XML. This means that assuming a very particular serialization of information on Web pages, the Amilcare tags can be specified so precisely[5] that indeed Table 1(b) can be rather easily mapped into Table 1(a). 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 doubleroom of a hotel.

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[6] 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 do.

The third process given in Figure 2 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 [12, 24] and explained in the next section.

5 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 general, a document), there is always a logical description (e.g., a 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.

[5] We abstract here from the problem of correctly tagging a piece of text.
[6] A template is like a single tuple in an unnormalized relational database table, where all or several entries may have null values.

(a) OntoMat	(b) Amilcare
Zwei Linden INSTOF Hotel	<hotel>Zwei Linden</hotel>
Zwei Linden is LOCATED_AT Dobbertin	
Dobbertin INSTOF City	<city>Dobbertin</city>
Zwei Linden HAS_ROOM single_room_1	
single_room_1 INSTOF Single_Room	<singleroom>Single room</singleroom>
single_room_1 HAS_RATE rate2	
rate2 INSTOF Rate	
rate2PRICE 25,66	<price>25,66</price>
rate2CURRENCY EUR	<currency>EUR</currency>
Zwei Linden HAS_ROOM double_room_3	
double_room_3 INSTOF Double_Room	<doubleroom>Double room</doubleroom>
double_room_3 HAS_RATE rate4	
rate4 INSTOF Rate	
rate4PRICE 43,46	<lowerprice>43,46</lowerprice>
rate4PRICE 46,02	<upperprice>46,02</upperprice>
rate4CURRENCY EUR	<currency>EUR</currency>
...	...

Table 1. Comparison of Output: Manual OntoMat versus Amilcare

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

Grosz & Sidner [12] 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 are available at all 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 *forward-looking centers*, $C_f(U_k)$ for each utterance U_k. The forward-looking centers $C_f(U_k)$ constitutes a ranked list of what is available and what is prefered 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 forward-looking centers, $C_f(U_{k-1})$. For instance, when reading "The chef of the restaurant" in Figure 1 the centering model allows relationships with "Country inn", but not with "Dobbertin".

The drawback of the centering model is that, first, it has only been devised for full text and not for semi-structured text such as appears in Figure 1 and, second, it often needs more syntactic information than shallow information extraction can provide.

Therefore, we use only an *extremly 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, though adaptable, domains. It is not even necessary to have a complete model, because we analyse only a very small part of the text. For instance, we analyse only the part about hotels with rooms, prices, addresses and hotel facilities. Note that thereby, hotel facilities are found in full texts rather than tables and not every type of hotel facility is known beforehand.

We specify the discourse model by logical rules, the effects of which we illustrate in the following paragraphs. Thereby, we use the same inferencing mechanisms that we have already exploited for supporting annotation [14], viz. Ontobroker [5].

As our baseline model, we assume the "single template stragey", 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. E.g. "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 2(a).[7] This is probably the shallowest discourse representation possible at all, because it does not include ordering constraints or other 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 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 2 by numbers; this may be modelled as 4-arity predicates in F-Logic used by Ontobroker) 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., prices are only attached where they are allowed)
- rules to attach tag values to the nearest preceding, conceptually possible entity (thus, prices for single and double room may be distinguished without further ado).
- rules to create a new complex object when two simple ones are adjacent, e.g., to create a rate when it founds adjacent number and currencies.

Further information that could be included is, e.g., adjacency information, etc. Thus, one may produce Table 1(a) out of the discourse representation from a numbered Table 2(a).

The strategy that we follow here is to make simple things simple and complex tasks possible. The experienced user will be able to handcraft logical rules in order to define 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

[7] Results have been selected to be comparable with Table 1.

Zwei Linden DR:INSTOF Hotel	Zwei Linden INSTOF Hotel
Zwei Linden DR:CITY Dobbertin	Zwei Linden is LOCATED_AT Dobbertin
	Dobbertin INSTOF City
Zwei Linden DR:SINGLE_ROOM single room	Zwei Linden HAS_ROOM single_room1
	single_room1 INSTOF Single_Room
Zwei Linden DR:PRICE 25,66	
Zwei Linden DR:CURRENCY EUR	
Zwei Linden DR:DOUBLE_ROOM double_room	Zwei Linden HAS_ROOM double_room1
	double_room3 INSTOF Double_Room
Zwei Linden DR:PRICE 43,46	
Zwei Linden DR:PRICE 46,02	
Zwei Linden DR:CURRENCY EUR	
(a) Discourse Representation	(b) Target Graph Structure

Table 2. Template Strategy

defined by directly aligning relevant concepts and relations by drag and drop, while in the general case one must write logical rules.

6 Usage scenario

This section describe a usage scenario. The first step is the project definition. A domain ontology can be the basis for the annotation of different types of documents. Likewise a certain kind of documents can be annotated in reference to different ontologies. Therefore a project defines the combination of a domain ontology (e.g. about tourism) with a certain text type (e.g. hotel homepages). Further the user have do define which part of the ontology is relevant for the learning task, e.g. which attributes of the several concepts will be used for tagging the corpus. The mapping of the Ontology to the Amilcare tags works as follows:

- concepts: concepts are mapped by the name of the concept, e.g. the concept with the name "Hotel" results in a <hotel> tag.
- inheritance: the concepts of the ontology represents a hierarchical structure. To emulate the different levels of conceptualization OnO-Mat allows to map a concept in multiple tags, e.g. the concept "Hotel" in <company>, <accommodation>, and <hotel>.
- attributes: The mapping of attributes to tags is a tradeoff between an specific and a general naming. The specific naming ease the mapping to the ontology concepts but at the same time 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 have to decide for every attribute the adequate accuracy of the naming, because it influences the learning results.

After the definition of the project parameters one needs a corpus, a set of certain type of documents, e.g. hotel homepages.

If there exist already enough annotated documents in the web the user can perform a crawl with OntoMat and collect the necessary documents. The crawl can be limited here to documents which are annotated with the desired ontology. If necessary the ontology subset and the mapping to the Amilcare tags must be re-adjusted according to the existing annotations in the crawled documents. Afterwards the desired type of document must be checked still manually.

If there are no annotated documents, one can produce the necessary corpus with OntoMat themselves. The user have to collect and annotate documents of a certain type by the sub-set of the ontology that is chosen in the project definition phase. The document are annotated by OntoMat with RDF facts. These facts are linked by an XPointer description to the annotated text part. Because Amilcare needs as a corpus XML tagged files, these RDF annotations will be 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.

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 3. After the training Amilcare stores the annotation rules in a certain file which belongs to the project.

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 2). Here a mapping (A2) is done from OntoMat from the flat markup to the conceptual markup in order to create new RDF facts (A3). These mapping is undertaken by the discourse representation (cf. section 5).

These mapping results in several automatic generated proposals for the RDF annotation of the document. The user can interact with these annotation proposals in three different ways of automation: (i) a highlighting of the annotation candidates or (ii) interactive suggestion of each annotation or (iii) a first full automatic annotation of the document and a later refinement by the user.

highlighting mode: First of all the user opens a document he would like to annotate in the OntoMat document editor. Then the highlighting mode marks all annotation candidates by a colored underline. The user can decide on his own if he use this hint for an annotation or not.

interactive mode: This mode is also meant for the individual document processing. The interactive suggestion is a step by step process. Every 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 fully automatic approach is useful if there is a bunch of documents that needs to be annotated, so it can be done in batch mode. All selected documents are annotated automatically.

7 Related Work

S-CREAM can be compared along four dimensions: First, it is a framework for mark-up in the Semantic Web. Second, it may be considered as a particular knowledge acquisition framework very vaguely similar to Protégé-2000[9]. Third, it is certainly an annotation framework, though with a different focus than ones like Annotea [18]. Fourth, it produces semantic mark-up with support of information extraction.

7.1 Knowledge Markup in the Semantic Web

We know of three major systems that intensively use knowledge markup in the Semantic Web, viz. SHOE [17], Ontobroker [5] and WebKB [21]. All three of them rely on knowledge in HTML pages. They all start with providing manual mark-up by editors. However, our experiences (cf. [8]) have shown that text-editing knowledge mark-up yields extremely poor results, viz. syntactic mistakes, improper references, and all the problems sketched in the scenario section.

The approaches from this line of research that are closest to *S-CREAM* is the *SHOE Knowledge Annotator*[8] and the WebKB annotation tool.

The SHOE Knowledge Annotator is a Java program that allows users to mark-up webpages with the SHOE ontology. The SHOE system [20] defines additional tags that can be embedded in the body of HTML pages. The SHOE Knowledge Annotator is rather a little helper (like our earlier OntoPad [10], [5]) than a full fledged annotation environment.

WebKB uses conceptual graphs for representing the semantic content of Web documents. It embeds conceptual graph statements into HTML pages. Essentially they offer a Web-based template like interface like knowledge acquisition frameworks described next.

7.2 Comparison with Knowledge Acquisition Frameworks

The S-CREAM framework allows for creating class and property instances to populate HTML pages. Thus it has a target roughly similar to the instance acquisition phase in the Protégé-2000 framework [9] (the latter needs to be distinguished from the ontology editing capabilities of Protégé). The obvious difference between S-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 to distinguish between different ways that classes and properties are treated.

7.3 Comparison with Annotation Frameworks

There are a number of — even commercial — annotation tools like ThirdVoice[9], Yawas [6], CritLink [26] and Annotea (Amaya) [18]. 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 to an existing document. For instance, a user of these tools might attach a note like ”A really nice hotel!” to the name “Zwei Linden” on the Web page. In S-CREAM we would design a corresponding ontology that would allow to type the comment (an unlinked fact) “A really nice hotel” into an attribute instance belonging to an instance of the class comment with a unique XPointer at “Zwei Linden”.

Annotea actually goes one step further. It allows to rely on an RDF schema 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, no further help is provided by Amaya for syntactically correct statements with proper references.

[8] http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html
[9] http://www.thirdvoice.com

7.4 Semantic Markup with Support from Information Extraction

The only other system we know that produce semantic markup with support from information extraction is the annotation tool cited in [25]. It uses information extraction components (Marmot, Badger and Crystal) from the University of Massachusetts at Amherst (UMass). It allows the semi-automatic population of an ontology with metadata. We assume that this approach is more laborious than to use Amilcare for information extraction, e.g. they had to define their own verbs, nouns and abbreviations in order to apply Marmot for a domain. Also, they have not dealt with relational metadata or authoring concerns so far.

8 Conclusion

CREAM is a comprehensive framework for creating annotations, relational metadata in particular — the foundation of the future Semantic Web. The new version of S-CREAM presented here supports metadata creation with the help of information extraction in addition to all the other nice features of CREAM, like comprises inference services, crawler, document management system, ontology guidance/fact browser, document editors/viewers, and a meta ontology.

OntoMat is the reference implementation of the S-CREAM framework. It is Java-based and provides a plugin interface for extensions for further advancements, e.g. collaborative metadata creation or integrated ontology editing and evolution. The plugin interface has already been used by third parties, e.g. for creating annotation for Microsoft Word^{TM} documents. Along similar lines, we are now investigating how different tools may be brought together, e.g. to allow for the creation of relational metadata in PDF, SVG, or SMIL with OntoMat.

REFERENCES

[1] Fabio Ciravegna, ‘Adaptive information extraction from text by rule induction and generalisation’, in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)e*, Seattle, Usa, (August 2001).

[2] Fabio Ciravegna, ‘Challenges in information extraction from text for knowledge management’, *IEEE Intelligent Systems and Their Applications*, **16**(6), 88–90, (2001).

[3] Fabio 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).

[4] Fabio Ciravegna and Daniela Petrelli, ‘User involvement in adaptive information extraction: Position paper’, 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).

[5] S. Decker, M. Erdmann, D. Fensel, and R. Studer, ‘Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information’, in *Database Semantics: Semantic Issues in Multimedia Systems*, eds., R. Meersman et al., 351–369, Kluwer Academic Publisher, (1999).

[6] L. Denoue and L. Vignollet, ‘An annotation tool for web browsers and its applications to information retrieval’, in *In Proceedings of RIAO2000*, Paris, (April 2000). http://www.univ-savoie.fr/labos/syscom/Laurent.Denoue/riao2000.doc.

[7] Aaron 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/.

[8] M. Erdmann, A. Maedche, H.-P. Schnurr, and Steffen 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).

[9] H. Eriksson, R. Fergerson, Y. Shahar, and M. Musen, 'Automatic generation of ontology editors', in *Proceedings of the 12th Banff Knowledge Acquisition Workshop, Banff, Alberta, Canada*, (1999).

[10] D. Fensel, J. Angele, S. Decker, M. Erdmann, H.-P. Schnurr, S. Staab, R. Studer, and Andreas Witt, 'On2broker: Semantic-based access to information sources at the www', in *In Proceedings of the World Conference on the WWW and Internet (WebNet 99), Honolulu, Hawaii, USA*, (1999).

[11] Reference description of the DAML+OIL (March 2001) ontology markup language, March 2001. http://www.daml.org/2001/03/reference.html.

[12] B. J. Grosz and C. L. Sidner, 'Attention, intentions, and the structure of discourse', *Computational Linguistics*, **12**(3), 175204, (1986).

[13] T. R. Gruber, 'A Translation Approach to Portable Ontology Specifications', *Knowledge Acquisition*, **6**(2), 199–221, (1993).

[14] S. Handschuh and S. Staab, 'Authoring and annotation of web pages in cream', in *Proc. of WWW-2002*, (2002).

[15] S. Handschuh, S. Staab, and A. Maedche, 'CREAM — Creating relational metadata with a component-based, ontology driven framework', in *In Proceedings of K-Cap 2001*, Victoria, BC, Canada, (October 2001).

[16] Siegfried Handschuh and Steffen Staab, 'Authoring and annotation of web pages in cream.', in *Proceeding of the WWW2002 - Eleventh International World Wide Web Conferenceb (to appear)*, Hawaii, USA, (May 2002).

[17] J. Heflin and J. Hendler, 'Searching the web with shoe', in *Artificial Intelligence for Web Search. Papers from the AAAI Workshop. WS-00-01*, pp. 35–40. AAAI Press, (2000).

[18] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick, 'Annotea: An Open RDF Infrastructure for Shared Web Annotations', in *Proc. of the WWW10 International Conference. Hong Kong*, (2001).

[19] Nicholas Kushmerick, 'Wrapper induction for information extraction', in *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, (1997).

[20] S. Luke, L. Spector, D. Rager, and J. Hendler, 'Ontology-based Web Agents', in *Proceedings of First International Conference on Autonomous Agents*, (1997).

[21] 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*, pp. 1403–1419. Elsevier Science B.V., (1999).

[22] Diana Maynard, Valentin Tablan, Hamish Cunningham, Cristian Ursu, Horacio Saggion, Kalina Bontcheva, and Yorick 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.

[23] 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).

[24] M. Strube and U. Hahn, 'Functional centering — grounding referential coherence in information structure', *Computational Linguistics*, **25**(3), 309–344, (1999).

[25] M. Vargas-Vera, E. Motta, J. Domingue, S. Buckingham Shum, and M. Lanzoni, 'Knowledge Extraction by using an Ontology-based Annotation Tool', in *K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation*, Victoria, BC, Canada, (October 2001).

[26] Ka-Ping Yee. CritLink: Better Hyperlinks for the WWW, 1998. http://crit.org/˜ping/ht98.html.