Interoperability for Semantic Annotations

Stefano Paolozzi
Università Roma Tre, Italy
stefano.paolozzi@irpps.cnr.it

Paolo Atzeni
Università Roma Tre, Italy
atzeni@dia.uniroma3.it

Abstract

Semantic annotations represent a basic technology for the creation of meaningful content for the Semantic Web. Many tools have been developed to realize semantic annotations platforms also using ontologies. An important characteristic such tools should satisfy is the semantic interoperability with other systems or tools. In this paper we examine problems of interoperability between different semantic annotations systems proposing solutions based on Model Management technique, with particular attention to the ModelGen operator.

1. Introduction

Semantic annotation can play an important role in pursuing the goals of the Semantic Web. Indeed, most of the information currently available on the Web was produced to be used by humans and it is therefore hardly suitable for automatic processing, with “anonymous syntax” rather than “explicit semantics”. Semantic annotation refers to the use of metadata for the description of Web resources, and the term is used to indicate both the metadata itself and the process that produces the metadata [6]. Semantic annotations are used to enrich the informative content of Web documents and to express in more formal way the meaning of a resource.

The existence of various semantic annotation platforms is interesting but, at the same time, gives rise to a difficulty, as the various tools have been developed independently from one another, and so it can become difficult to take advantage of annotations produced in different environments. Indeed, we have here an “interoperability” problem, similar to those we encounter in various information systems or database areas. Here interoperability refers to the possibility of sharing and exchanging annotations. Interoperability is important in various scenarios, which include the translation of annotations from a framework to another (for example because they were produced in one and have to be used in another) or the integration of annotations coming from different, heterogeneous sources.

In this paper, we propose a novel approach to the management of interoperability of semantic annotation platforms, based on model management techniques [3] and, specifically, on the ModelGen operator. ModelGen was conceived in the database context, with the goal of translating database schemas and instances from a data model to another, in a model generic framework: within a universe of models, given two models (the “source” and the “target”) in the universe and a schema (with its instances) in the source model, the goal of ModelGen is to generate a schema (and its instances) in the target model that represents (according to a suitable notion of “representation”) the same information as the source one.

Specifically, we consider a ModelGen proposal [1] that adopts a “metamodel” approach: the various data models are defined in terms of a common set of metaconstructs and then translations between models are specified on the basis of translations over the metaconstructs. We propose to adopt such an approach to support interoperability of semantic annotations based on the ModelGen ideas. Specifically, we analyze the annotations used by some common tools and show how translations between them can be handled. As annotations often depend on the individual domain of interest, we consider a specific domain, referring to meetings.

The paper is organized as follows. In Section 2 we briefly describe some semantic annotations platforms. Section 3 is devoted to the description of the main concepts of ModelGen. In Section 4 we illustrate a possible metamodel for the integrated description of two models of semantic annotation. In Section 5 we show the way translations should be performed. In Section 6 we draw some conclusions.

2. Semantic annotation platforms

Semantic annotation is the creation of metadata and relations between them with the task of defining new methods of access to information and enriching the potentialities of the ones already existent.

The main goal is to have information on the Web, de-
fined in such a way that its meaning could be explicitly interpreted also by automatic systems, not understood just by humans. To reach such a target, it is necessary to associate meta-information with Web resources through semantic annotation. In particular, we want to annotate the resources with semantic information which give indications about the content of the resource itself. To do it, we need languages that can support the semantic representation of the meta-information such as RDF and RDF Schema. However, annotations specified with these languages do not significantly enrich the automatic processes, unless it could be possible to define methods which will allow to share the common knowledge about the meaning of the annotations themselves. An additional step is the adoption of ontologies, as “shared conceptualizations of a particular domain” [5], which allow for a machine-processable description of the semantics of the resources of interest.

Annotations based on ontological structures can be defined at two different levels: concepts or instances of concepts. Annotation through instances is the association of the instance with the associated element to increase the value of the properties that describe the instance itself. Annotation through concepts involves the association of some concept of the ontology with the annotated element [6]. In general, the structure of the data and the semantics inserted by ontologies improve the potentialities of the Web: for example, information retrieval systems can become more effective, if they can refer to a specific concept and search and find the pages which are really related to such a concept, not the ones which contain ambiguous or generic keywords.

In order to give a concrete idea of our approach, as ontologies describes individual domains, we need to refer to a particular domain. In the next subsections we will analyze two semantic annotation platforms, within the context of a specific domain, that of meetings, and with attention to multimedia objects.

2.1. NOMOS

NOMOS (aNnotation Of Media with Ontological Structure) is a platform for ontology-based semantic annotation of multimedia objects, developed at Stanford University Laboratories [10]. Ontologies are specified in a formal way in OWL [4]; this gives a rich meaning to the annotations and provides inference capabilities which enhance the annotation process as well as the annotations in many useful ways.

NOMOS ontologies (or more precisely the terminological components of these ontologies) are used as high level schemas for annotations. In particular NOMOS has a reference ontology (called corpora_2.0) which is predefined, but can be extended. The reference ontology is based on abstract data definition such as: event, entity, relations, etc., which represent basic unmodifiable concepts. The ontology concepts schema is shown in Fig. 1.

2.2. The NITE XML Toolkit

The NITE XML Toolkit (NXT) is an open source software tool for working with language corpora, within the meetings domain [8]. It is designed to support the needs of human analysts as they work with a corpus, for tasks such as hand-annotation, automatic annotation where that relies on complex match patterns, data exploration, etc. NXT data model and query language are oriented toward those users who build descriptive analysis of heavily annotated multimodal corpora in preparation for defining appropriate statistical models. In NXT, annotations are described by types and attribute-value pairs. Moreover, they can be related to signals via start and end times, to representations of the external environment, and to each other via a graph structure. NXT represents the data for one meeting as a related set of XML files, with a metadata file that expresses information about the structure and location of the files.

The main element of NXT data model is corpus. A corpus consists of a set of observations, each of which contains the data for one dialogue or interaction. Each of the dialogues or interaction types defined in a corpus involve a number of human or artificial agents. An agent is one actor in an observation. Each observation is recorded by one or more “signals” which consist of audio or video data characterizing the duration of the interaction. The NXT model is shown in Figure 2.
3. ModelGen

ModelGen has been developed in order to handle the translation of schemas and databases from a data model to another [1]. Let us summarize its main ideas. A metamodel is a set of constructs, called metaconstructs, which can be used in models definition. The approach is based on Hull and King observation [7] that the constructs used in most known models can be expressed by a limited set of generic (i.e. model-independent) metaconstructs: lexical, abstract, aggregation, generalization, function. Translations of schemas between two different models are defined as translation of the metamodel’s metaconstructs; in this way the proposed approach allows to use the same translation for a metamodel’s construct regardless of the model in which it is actually used.

In ModelGen, each model is defined by its constructs and the metaconstructs they refer to. For example, a simplified version of the Entity-Relationship model has (i) abstracts (entities); (ii) aggregations of abstracts (relationships); (iii) lexical attributes of abstracts (attributes of entities). At the same time, an object-oriented model would also have abstracts (the classes) and their lexical attributes of abstracts (fields of classes), but then it would have references between abstracts, rather than their aggregations.

An important concept in the approach is the supermodel which has the constructs corresponding to all known metaconstructs. Each model, in this approach, is a specialization of the supermodel, so a schema in any model is also a schema in the supermodel. In other words the supermodel is like a pivot model, so that it is sufficient to have translations from each model to and from the supermodel, rather than translations for any pairs of models. With this approach the number of translations is reduced to a linear number with respect to the total number of the models recognized by the system. Indeed the only needed translations are those within abstracts, rather than their aggregations.

The ModelGen operator, initially proposed by Bernstein [3], can be suitably extended to be used to solve the interoperability problem between semantic Web languages (see [2] for an example).

4. A metamodel for Semantic Annotation

In this initial proposal we show how our approach can be used to handle translations between NOMOS and NXT which are clearly representatives of the problems and allow us to illustrate the major features of our technique.

The data models of NOMOS and NXT have a number of similarities and a few major differences. The most important similarity is that, despite some minor differences, it is possible to claim that the basic components of the two models, Thing in NOMOS and Corpus in NXT are very similar to each other. They correspond to notions we have in data models, at the instance level, such as occurrences of entities in the ER model or objects, belonging to classes, in object oriented models.

The way an annotation is described in terms of properties and related to one another is indeed slightly different in the two models. Indeed, NOMOS has an ontology concept to define the relation between things, whereas in NXT, relations between different corpora are not explicitly defined.

The actors of an annotation can be represented in both models. In NXT there is the Agent object, while in NOMOS we can use a specialization of the concept Entity, the only difference is on the categorization of the agents.

The Event concept in NOMOS ontology can reflect the Observation data object of NXT model, because they both represent a particular event of a meeting that can be observed.

A complete development of the analysis sketched above would allow us to understand which are the constructs in the two data models that are in close correspondence and which are unique to one of the two. In order to achieve this task it is useful to define a suitable metamodel (in ModelGen sense) of NOMOS and NXT. Possible metamodels are reported in Figures 3 and 4 where each construct is identified by a unique OID (Object Identifier). This would lead to the definition of a supermodel for this framework: a set of constructs, some of which appear (usually with different names) in both NOMOS and NXT models, and the others in only one.

The metamodel we use includes a few of the basic features, which we use essentially as representative of properties. For example minimum cardinalities are represented by means of the boolean IsOptional, so that allowed cardinalities are 0 ("true") and 1 ("false") and maximum cardinalities are represented by IsFunctional, with 1 and N as possible cardinality values. Many other features are omitted here for the sake of space (for example optionality or nullability) but could be easily included.

Let us conclude the section by listing and commenting the metaconstructs of interest, with the correspondences in the two models (it should be noted that the correspondence is in some cases not really precise, but this could be dealt with by considering “variants” of constructs, which are omitted in this preliminary discussion):

SM_Abstract (we use the prefix SM for this and the other constructs, as they belong to the supermodel)
struct \textit{SM\_Abstract} is used to describe all the kinds of objects, concrete or abstract. It corresponds to constructs used in many models, such as ER model entities, semantic network nodes, NOMOS things and NXT corpora. Each object must have an identity to be uniquely identifiable. There are variants for this construct, corresponding to classes, instances, and properties, distinguished by means of attributes.

\textit{SM\_Collection} It models collections of objects of the same type, with variants corresponding to the specific form (such as set, list, or bag). This is not really essential here, but could be needed for other platforms, and we have included it here for the sake of completeness.

\textit{SM\_Object} It is the generalization of \textit{SM\_Abstract} and \textit{SM\_Collection}, described above.

\textit{SM\_AggregationOfAbstract} It allows the definition of a relationship on two or more \textit{SM\_Abstract} components, and corresponds to relations in NOMOS (as well as relationships in the ER model).

\textit{SM\_ComponentOfAggregation} This construct allows the specification of the participation of an object to a \textit{SM\_AggregationOfAbstract}, this includes the definition of the role in NOMOS relation (i.e. Subject or Object thing).

\textit{SM\_ParticipantOfAbstract} It allows the definition of the actors of a \textit{SM\_Abstract}. This models the Entity concept in NOMOS and the Agent one in NXT.

\textit{SM\_Signal} This construct correspond to the constructs used in many models of semantic annotations to represent the different multimedia sources that characterize the interaction that must be annotated.

\textit{SM\_AttributeOfSignal} It describes a property of a signal. It can even be represented by a literal, and also define signals features.

\textit{SM\_Annotation} This construct models the concept of the annotation. It can be either an abstract or even a literal.

\textit{SM\_Type} An assertion is related to a class that expresses its meaning.

\textit{SM\_Identity} It describes the correspondence between an \textit{SM\_Object} and a form of identification for it.

Figures 5 and 6 show the correspondence between the constructs in the two models of interest and those in the supermodel.

5. Translations between Semantic Annotation platforms

For the purposes of semantic annotations interoperability, translations from a model to another can be carried out with the same strategy of previous ModelGen work [1], namely: (i) import of the source data to the supermodel; (ii) translation within the supermodel; (iii) export of the data from the supermodel to the target model.

By definition of supermodel, the first and the third steps are essentially copy operations and so they can be automat-
ically managed once the correspondence between the constructs of the models and the constructs of the supermodel is built. In Fig. 5 the correspondences represented by a single arrow can be implemented by a plain copy operation from the metamodel to the supermodel tables. In the cases the figure shows a double arrow, the copy is slightly more complex: an element of NOMOS, Event has a correspondence with an element in SM_Object and an element in SM_Identity: the identifier is represented in SM_Object whereas in SM_Identity there are information on the identifying mechanism.

Translations within the supermodel can then be carried out by eliminating constructs that appear in the current set of data and are not allowed in the target model.

The translations we are considering here can be implemented by noting which are the constructs available in one model and not available in the other. Considering a translation from NOMOS to NXT we will have to eliminate SM_AggregationOfAbstract by replacing them with other constructs, for example new SM_Abstracts (see Figures 5 and 6). We can roughly state that this is a translation similar to the standard translation in conceptual models that replaces n-ary relationships by means of binary ones. A complete translation can then be built by specifying the basic translations needed to replace the constructs that are not available in the target model and then composing them.

In Fig. 7 a simple example of translation is sketched. Starting from an NXT schema model (Fig. 7(a)) the system performs a few basic translations in order to obtain a model compliant to NOMOS (Fig. 7(d)). The first step is the change of the connection of the AgentName as shown in Fig. 7(b), in other words we must change the reference to the original containing object. Fig. 7(c) shows intuitive sketches of the intermediate schemas in this process. The translation into the NOMOS environment requires the introduction of a new object (AudioRec) for the signal class of NXT model associated to the annotation (AnnotationRef) and the SignalRef becomes a child of this one. As we said before, translations are performed by composing elementary steps translating NXT metaclasses into NOMOS metaclasses (Fig. 7(d)) and then composing them.

6 Conclusions

We have shown how the ModelGen approach can be at the basis of a framework for the translation of annotations from a platform to another. In this paper we have addressed two semantic annotation platforms, NOMOS and NXT, but our approach can be extended to similar platforms, such as the Annotation Graph Toolkit [9], which have a schema-based structure.

The approach has turned out to be feasible, but, given the differences between this problem and standard database settings, it has been needed to define a new metamodel, with some constructs that are new, as they were not needed for database models. A future, interesting step would be the integration of the two frameworks, which would, however, present an additional problem: the different level of “semantic power”, as database models are inherently syntactic, whereas semantic annotation ones carry some “meaning”; therefore, translations will have to consider some forms of information loss.

References