A Core Ontology of Knowledge Acquisition

José Iria

Department of Computer Science, The University of Sheffield, UK
jiria@dcs.shef.ac.uk

Abstract. Semantic descriptions of knowledge acquisition (KA) tools and resources enable machine reasoning about KA systems and can be used to automate the discovery and composition of KA services, thereby increasing interoperability among systems and reducing system design and maintenance costs. Whilst there are a few general-purpose ontologies available that could be combined for describing knowledge acquisition, albeit at an inadequate abstraction level, there is as yet no KA ontology based on Semantic Web technologies available. In this paper, we present OAK, a well-founded, modular, extensible and multimedia-aware ontology of knowledge acquisition which extends existing foundational and core Semantic Web ontologies. We start by using a KA tool development scenario to illustrate the complexity of the problem, and identify a number of requirements for OAK. After we present the ontology in detail, we evaluate it with respect to the identified requirements.

1 Introduction

The goal of Knowledge Acquisition (KA) is to develop methods and tools that make the arduous task of capturing and validating an expert's knowledge as efficient and effective as possible. Of special relevance to the fulfillment of the Semantic Web vision is automating KA from text and image resources. Automated KA systems take as input multimedia documents originally intended for human consumption only and provide as output knowledge that machines can reason about.

Despite the growing need for KA and recent advances in the field, the development of KA systems remains a complex and costly affair. Previous research on KA for the Semantic Web has followed three main directions: 1) increasing the accuracy of the extracted knowledge [9], 2) scaling up KA systems to address the knowledge acquisition bottleneck in populating large knowledge bases [12], and 3) providing semantically rich and well-founded means of representing multimedia data and annotations thereof [1]. Unfortunately, none of these research directions addresses the core problem of reducing the complexity in developing and maintaining these systems.

Research on Service-Oriented Architectures (SOA) addresses this problem. Frequently realized through (semantic) Web services concepts and technologies, SOA encourage following modularization principles for dealing with the complexity of software systems. Ontological descriptions of services serve as a basis

L. Aroyo et al. (Eds.): ESWC 2009, LNCS 5554, pp. 233–247, 2009.

for interoperability between systems at various levels of abstraction, and this enables important functionalities such as (automatic) semantic service discovery and composition. In this context, ontologies like OWL-S [8] and the Core Software Ontology [11] provide generic top-level constructs to formally describe software systems. However, to the best of our knowledge, there is currently no specialized ontology available for semantically describing KA tasks and systems.

The advantages of an ontology of KA are threefold. Firstly, it provides an agreed-upon means of referring to and describing KA tasks and KA systems, their inputs and outputs and their internal constituents (components, subsystems, auxiliary resources, and so on). Secondly, making such descriptions available greatly increases the interoperability between KA systems, by enabling both humans and machines to search for and reason upon them. Lastly, it cuts down system development and maintenance costs, and promotes optimal decisions both at design time and at runtime, because systems, subsystems and components of a KA system can more easily be found and put together.

Our contribution in this paper is thus to provide a core Ontology of Acquisition of Knowledge (OAK). In the design of OAK we have reused and extended existing foundational and core Semantic Web ontologies. Concretely, we have used DOLCE [5], the ontology of Descriptions & Situations, the Ontology of Plans, and the Ontology of Information Objects; and the core ontologies of multimedia (COMM) [1] and software (CSO) [11]. The relationship between OAK and these other ontologies is depicted in Figure 1.

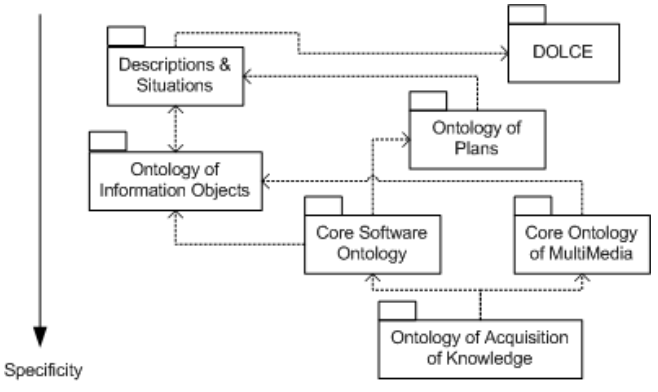

Fig. 1. Positioning the Ontology of Acquisition of Knowledge (OAK) in the framework of existing foundational and core ontologies. OAK extends the Core Software Ontology (CSO) and the Core Ontology of MultiMedia (COMM).

The rest of the paper is structured as follows. In the next section, we motivate the need for semantically describing KA systems. We center our discussion on a simple KA task to populate a knowledge base. In section 3, we review related work. In section 4, we identify a number of requirements that an ontology of knowledge acquisition should satisfy. We then present OAK using a collection of

UML diagrams and describe it in detail. In section 6, we show how the simple KA task is encoded with the proposed ontology. Finally, we present our conclusions and outline future work.

2 Developing a Simple Knowledge Acquisition System

Let us imagine that a team of developers is working on a new KA system. The system's goal is to extract facts about sports celebrities from online multimedia documents. Available to the developers is a set of sample facts extracted from a few hundred documents that can be used as training examples in a machine learning-based approach to the problem. The system should make use of both text and image features, as both are potentially valuable – for example, the presence of images about sports increase the likelihood that the surrounding text contains the facts of interest (expressed in natural language).

Developing a new KA system involves a number of decisions regarding subsystems and components to use. Even the development of a simple system that works in the above scenario requires addressing at least the following concerns:

Decomposition: Both text and images need to be decomposed into finer-grained media segments, which constitute the building blocks for deriving Information Extraction (IE) patterns and models, be it manually or via induction algorithms. Text is typically decomposed into sentences, noun phrases and tokens. Images are typically segmented into regions of interest.

Tagging/Annotation: Media segments need to be tagged or annotated, that is, extra information, obtained through some form of analysis and/or use of external resources, is attached to the segment, with the purpose of enriching IE patterns and models. Typical text tagging tools are part-of-speech and orthography taggers, which attach tags at the token level. Typical image analysis tools are color and texture analyzers, and edge detectors.

Data Modeling: A predictive model, capable of extracting facts from the decomposed and tagged input media, needs to be constructed, either manually or through automatic induction methods. Manually built models consist mainly of text patterns, carefully created, tested and maintained by domain and linguistic experts. Induced models are created by machine learning-based systems, which tap into a wealth of machine learning literature for a choice of algorithms and meta-algorithms, and are able to work on both text and images.

Semantic Annotation: Models need to be applied over unseen media to extract novel facts, typically expressed in the form of semantic annotations. The application of predictive models may sometimes not directly yield the facts of interest, but some form of validation, consistency checking, and merging of intermediate facts is required as a last step. This is typically achieved through reasoning on the output of the models.

2.1 Problems with Existing Solutions

State-of-the art middleware for the development of IE and KA systems do not offer facilities for semantically describing the subsystems and components that make a KA system, nor is there a specialized ontology of KA available that can be used to do that (see Section 3). This means that the developers in our example will find it hard to integrate the required tools and components obtained from several parties. Furthermore, they will not be able to publish the semantic descriptions of their system, to ease the effort by third parties in finding and using them in the future. Given the sheer number and diversity of subsystems involved, it would be desirable to automate service discovery and composition, not only to speed up development, but also to improve systemic qualities such as robustness to failure of one of the subsystems.

Our developers will use the XML plugin descriptor files or equivalent mechanisms (e.g., Java language annotations) which are offered by the current middleware solutions. Unfortunately, these descriptors present several problems:

Scope: The descriptors tend to mimic the native programming language, thus describing the components in terms of their inputs and outputs, and design time and runtime parameters. There is seldomly a way of describing aspects of the KA subsystems and components that fall outside the scope of those constructs. For instance, it would prove difficult to represent that the output of the component is a relationship between a given decomposition of text and images and a statement in the knowledge base.

Semantics: The descriptors lack formal semantics. While appropriate for describing the components within the context of their respective middleware, it is not possible to guarantee that the descriptors generated by different agents will be mutually understood – potentially, the same descriptor syntax can be used to mean a number of semantically distinct things.

Web Interoperability: The descriptors are not interoperable with existing (semantic) web standards. For example, there is no formal way of referring to a concept in a domain-specific ontology. This limits the interoperability with other web systems.

3 Related Work

In recent years, new directions in software engineering have started to be explored to combat the increasing complexity and rapid rate of change in modern systems development. Among these new paradigms is Semantic Web Enabled Software Engineering (SWESE), which tries to apply Semantic Web technologies (such as ontologies and reasoners) in mainstream software engineering [7]. SWESE hopes to provide stronger logical foundations and precise semantics for software models and other development artifacts. The work presented in this paper can be viewed as a contribution to this goal, with applications in the domain of knowledge acquisition systems engineering.

There are currently several state-of-the-art software frameworks available that are suitable for developing tools for knowledge acquisition from text and/or multimedia documents. The Unstructured Information Management Architecture (UIMA) [4] is an open-source platform for integrating components that analyze unstructured sources such as multimedia documents. UIMA-based systems define type systems (i.e., ontologies with extremely limited semantic commitments) to specify the kinds of information that they manipulate [6]. UIMA type systems include no more than a single-inheritance type/subtype hierarchy, thus to do substantive reasoning over the results of UIMA-based extraction, one needs to convert results into a more expressive representation. GATE [2] is a framework and graphical development environment for NLP tools and applications that has been extended over the years to include support for ontologies, multimedia data, and machine learning. Unlike UIMA, GATE does not provide a specialized data layer, using instead native data structures (the current version of GATE is implemented in Java). It features a plugin-based architecture in which components are described using so-called CREOLE (XML) descriptor files, which suffer from the problems identified in the previous section. In our survey of the state-of-the-art, we found that several other frameworks, e.g. [15], [16], presented identical limitations. In face of this, we believe that the formalization of the knowledge acquisition domain presented here is an important first step in the direction of interoperability between systems built on top of these middleware platforms.

Early research approaches to using ontologies to model the knowledge acquisition problem, e.g. [13], would focus on the interaction between a KA user interface and the domain experts. The work in this paper belongs to the class of more recent approaches that attempt to automate KA from documents written by the experts.

Our work is targeted at the fields of Knowledge Acquisition and Ontology-based Information Extraction [14]. The design of OAK was greatly inspired by working with a number of systems in these fields and by designing our own systems. For example, the system in [3] takes the approach that a large class of data sources on the web can be viewed as semantically related in the context of a given knowledge acquisition task, and design a general strategy for learning classifiers from those sources. This and other systems, such as those described in [10], [9] and [12], which at aim populating small ontologies from (sometimes very large) repositories of text or multimedia data, are the kinds of systems that would benefit from our work. None of these systems have been explicitly semantically described, and therefore machine reasoning about their functionality is not possible – and it is not uncommon for even humans to have difficulty in replicating the systems from their natural language descriptions found in the literature.

Other, more general, ontologies have been developed which could be used together to describe KA tasks and systems, albeit at an inadequate abstraction level. The Core Ontology of MultiMedia [1] is a well-founded ontology and API that provides constructs to semantically describe multimedia assets available on the Web. The Core Ontology of Software [11] formalizes the most fundamental

concepts which are required to model both software components and Web services, including concepts such as software, data, users, access rights or interfaces. Roughly speaking, COMM covers the data part of our ontology, while CSO covers the processing part. We have extended both ontologies and provided more specialized semantic constructs for the domain of automated KA from multimedia data.

4 Requirements for the Knowledge Acquisition Ontology

Here we compile a list of requirements that an ontology of knowledge acquisition should satisfy, given our discussion in the previous sections:

Multimedia: The proposed ontology should describe KA systems that work with multimedia data. In particular, it should support text and images, and be designed in such a way that it can be extended to other types of media as well.

Semantic Interoperability: An ontology of KA, much like any other ontology, must ensure that the intended meaning of the captured semantics can be shared among different systems. Reasoning processes about concepts and relations in different environments can only be guaranteed to yield identical results if the semantics is sufficiently explicitly described. Only when the captured semantics can be shared among multiple systems and applications are the KA system descriptions truly re-usable.

Syntactic Interoperability: The semantics of the KA system descriptions are only shareable among different systems if there is some agreed-upon syntax in which to convey it. In our case, the descriptions should be expressed in a semantic web language, such as OWL or RDF/XML, because our goal is to take advantage of other Semantic Web technologies such as semantic service discovery and composition.

Separation of Concerns: Domain knowledge should be kept separate from knowledge about the KA system. Moreover, as indicated in section 2, the development of a KA system should address at least the concerns of decomposition, tagging, data modeling, and semantic annotation. These should also be kept separate, if possible, in the design of the ontology, in form of patterns.

Modularity: Modularity is a key engineering principle which arises whenever dealing with large systems, be it software systems or ontologies. Since the ontology of knowledge acquisition can become very large, its design should be made modular from its inception.

Extensibility: Just like we have extended COMM and CSO in designing our KA ontology, we also expect OAK to be eventually extended and adapted to specific domains and applications. As ontology development methodologies show, ontologies are inherently incomplete, and for that reason extensibility is a key and pervasive requirement in ontology engineering.

5 The OAK Ontology

In this section we present OAK, a core ontology of knowledge acquisition from multimedia data. The ontology is organized into several patterns, described below with the aid of UML diagrams.

5.1 Foundational and Core Ontologies Used

The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [5] is used as a modeling basis. DOLCE is a well-founded ontology which models three ontological patterns which we require: the Descriptions & Situations (D&S) ontology, the Ontology of Information Objects (OIO) and the Ontology of Plans (OoP). The first pattern is used to formalize contextual knowledge. The second pattern implements a semiotics model of communication theory. The third pattern characterizes planning concepts.

As previously mentioned, we extend the COMM and CSO core ontologies. Please consult Section 3 for the references to the publications which describe them, a reading of which may prove necessary to understand some of the concepts discussed in this section.[1]

5.2 KA Task and KA System

We start by characterizing core KA concepts by answering the questions: What is a knowledge acquisition task? What is a knowledge acquisition system? What is a knowledge base (KB) statement? We define these concepts in terms of the Descriptions & Situations pattern, which contextualizes an interplay of COMM and CSO concepts. This is depicted in Figure 2.

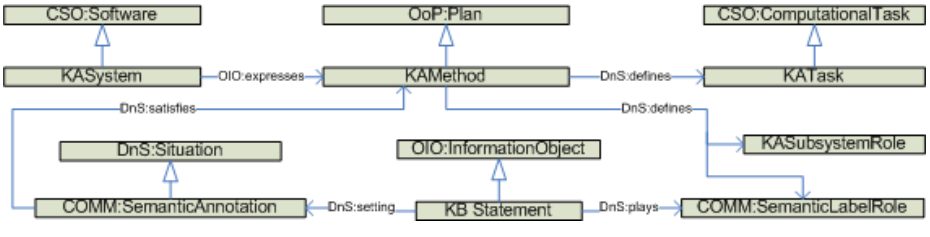

Fig. 2. Ontology pattern for describing a knowledge acquisition system

A `KASystem` is a `CSO:Software` that expresses a `OoP:Plan`, namely that of a `KAMethod`. A `KATask` is a `CSO:ComputationalTask` defined by the method. A statement in a knowledge base, `KBStatement`, is a `DnS:Role`,

[1] Throughout the paper, concepts and associations are written in `sans serif` and are labelled in a namespace-like manner. Namespace prefixes indicate the ontology where those concepts and associations are defined. If no namespace is used, they are assumed to be defined in OAK.

namely the COMM:SemanticLabelRole, played by an OIO:InformationObject. In other words, the statement is an information object expressing a fact that provides a semantic label to some media segment – the segment which contributed to that fact being extracted or derived. A KAMethod is a DnS:Description of a situation. In the KA domain, the situation it describes is a COMM:SemanticAnnotation. This provides a DnS:setting where the statements and the CSO:ComputationalActivity sequenced by the KATask (not shown in the figure) exist.

Additionally, the KAMethod defines the KASubsystemRole, the meaning of which will become clear in the next section.

5.3 KA Subsystem

Knowledge acquisition systems are complex systems composed of many subsystems. Our ontology should capture these as well. This is depicted in Figure 3, for natural language processing (NLP) tools, arguably the most commonly used type of KA subsystem.

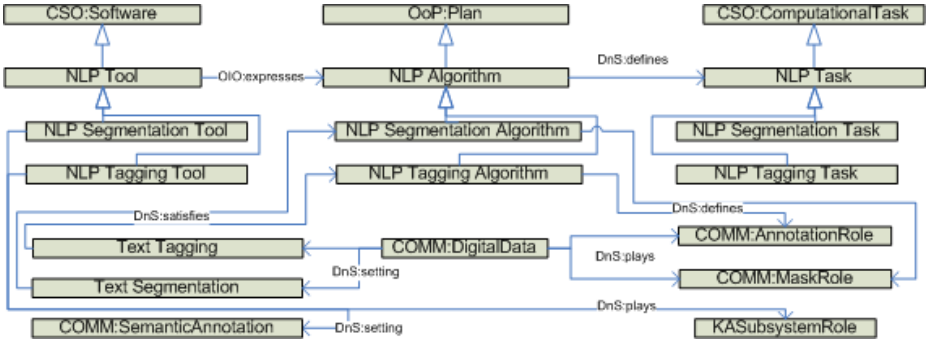

Fig. 3. Ontology pattern for describing one of the possible types of KA subsystems: NLP tools

A NLPTool is a software that OIO:expresses a NLPAlgorithm which DnS:defines an NLPTask. Following COMM patterns of decomposition and annotation, NLP tools can be specialized into segmentation and tagging tools. Examples of the former include the sentence splitter, the chunker and the tokenizer, while examples of the latter include tools such as the part-of-speech tagger or the orthography tagger. Both NLPTaggingTools and NLPSegmentationTools play the KASubsystemRole in the setting of a COMM:SemanticAnnotation (defined above). A NLPSegmentationTool expresses a NLPSegmentationAlgorithm that defines a NLPSegmentationTask and the COMM:MaskRole, which is played by some COMM:DigitalData in the setting of a TextSegmentation situation satisfied by the algorithm. Conversely, a NLPTaggingTool expresses a NLPTaggingAlgorithm which defines a NLPTaggingTask and the

COMM:AnnotationRole, played by some COMM:DigitalData in the setting of a TextTagging situation. The TextSegmentation and TextTagging situations are specializations of the COMM:SegmentDecomposition and COMM:Annotation situations, respectively (see Figure 4).

Many other tools may form part of a KA system. Of particular relevance to the simple KA system described in Section 2 are image analysis tools, which, together with NLP tools, enable handling multimedia documents. Due to space constraints, we do not show figures about these tools, but it should be mentioned that their characterization follows a similar pattern. Thus, a IATool is a software that OIO:expresses a IAAlgorithm which DnS:defines an IATask. We define two kinds of image analysis tools, the IADecompositionTool, e.g., a region of interest classifier, and the IAAnnotationTool, e.g., an edge detector . Like NLP tools, both play a KASubsystemRole in the setting of a COMM:SemanticAnnotation.

5.4 Decomposition

The subsystems of a KA system are typically regarded as blackboxes, and this lack of detail makes it impossible for machines to reason about them. Hence, we need semantic constructs to declaratively specify the data resources and internal processes used by these subsystems. Here, we specialize a number of generic COMM constructs to the KA domain. This is depicted in Figure 4.

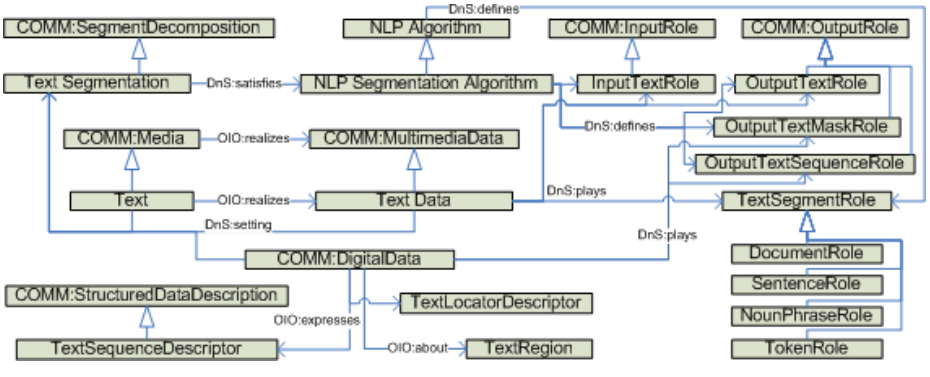

Fig. 4. Specializing the COMM decomposition pattern to the KA domain

Text is a type of COMM:Media that realizes TextData (a specialization of COMM:MultimediaData). A NLPAlgorithm defines a TextSegmentRole which TextData can play in the setting of some situation. In the case of the TextSegmentation situation depicted, TextData also plays the InputTextRole and OutputTextRole roles defined by the NLPSegmentationAlgorithm. In other words, the segmentation algorithm can, for example, take a document and split it into sentences (sentence splitter), or it can take a sentence and split

it into tokens (sentence-level tokenizer), or it can take a document and split it into tokens (document-level tokenizer), and so on. Two other types of outputs are defined by a segmentation algorithm: the `OutputTextMaskRole` and the `OutputTextSequenceRole`. The former is played by a `COMM:DigitalData` object that expresses a `TextLocatorDescriptor` and is about some `TextRegion`, e.g. a descriptor about a sentence that provides a means of locating it in the document. The latter is played by a `COMM:DigitalData` object that expresses a `TextSequenceDescriptor`, e.g. a descriptor about the order in which tokens appear in the document[2].

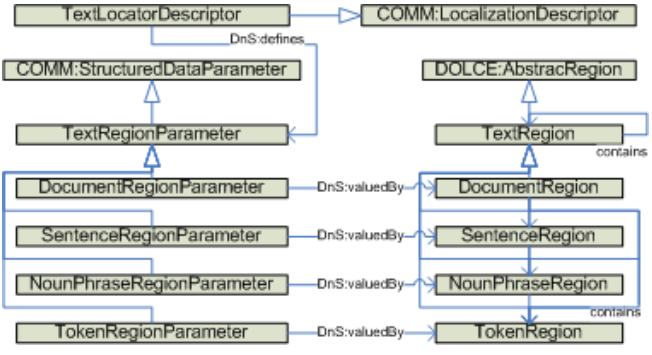

Fig. 5. Specializing COMM locators for text data

Figure 5 shows the `TextLocatorDescriptor` and the `TextRegion`. The former is a specialization of `COMM:LocalizationDescriptor` for text data. This descriptor enables specifying regions in the `TextData` manipulated by the algorithm. It defines a number of types of `TextRegionParameters` valued by their respective `TextRegions`. This is by no means an exhaustive list of possible subclasses, but merely illustrative of a pattern that suits a number of different concrete implementations and can easily be extended.

For the decomposition of image data, the constructs introduced in [1] by the authors of COMM are sufficient for the purposes of OAK, and thus do not need to be extended.

5.5 Annotation

Addressing the annotation concern requires a very similar pattern to the decomposition pattern. Hence, we just outline the differences with respect to the semantic constructs depicted in Figure 4.

[2] In practice, this descriptor is often expressed implicitly in the native programming language. For example, returning a `List` of `Token` objects in Java implicitly encodes the sequence information, since lists are ordered collections.

In a `TextTagging` situation satisfying a `NLPTaggingAlgorithm`, the latter can take as input and return as output `TextData` that play different `TextSegmentRoles`. For instance, a part-of-speech tagger tags tokens, while a sentiment classifier may tag sentences. A tagging algorithm defines a `InputTextRole` and a `OutputTextTagRole`. The latter is played by a structured `COMM:DigitalData` object that `OIO:expresses` a `TextTagDescriptor` (a subclass of `COMM:StructuredDataDescription`). The `COMM:DigitalData` object mentioned is `OIO:about` a `DOLCE:Particular`.

As with the decomposition pattern, we found that there was no need to extend the constructs introduced in [1] for the annotation of image data, as they are sufficient for the purposes of OAK. Moreover, for the same reasons we also directly adopt the semantic annotation pattern of COMM, as we find it requires no specialization in order to address the semantic annotation concern mentioned in Section 2.

5.6 Data Modeling

Another identified concern was that of managing models of the data. An ontology pattern that addresses this concern is depicted in Figure 6.

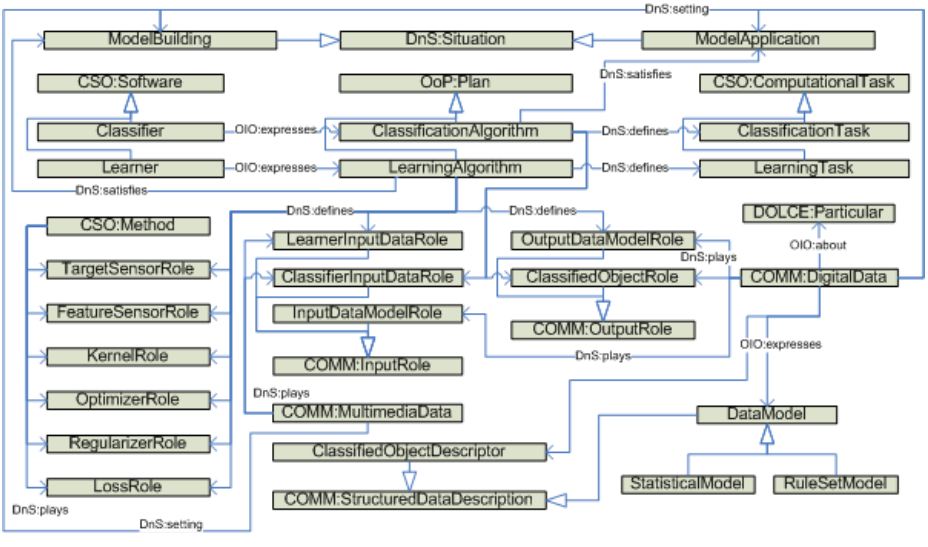

Fig. 6. Ontology pattern for describing building and applying data models

There are two situations to describe in data modeling: `ModelBuilding` and `ModelApplication`. A `Learner` is a `CSO:Software` that expresses a `LearningAlgorithm`, which defines a `LearningTask` and satisfies the `ModelBuilding` situation. Conversely, a `Classifier` is a `CSO:Software` that

expresses a ClassificationAlgorithm, which defines a ClassificationTask and satisfies the ModelApplication situation.

For the purposes of KA from multimedia documents, a LearningAlgorithm takes COMM:MultimediaData as input and outputs a DataModel (expressed by some COMM:DigitalData). Data models are often automatically induced via machine learning methods, resulting in StatisticalModels, or often manually crafted by domain experts in the form of a set of rules, resulting in RuleSetModels. Learning algorithms tend to be highly configurable, and we tried to capture that by defining a number of roles for well-known basic functions in the machine learning literature (here semantically described by a CSO:Method). The KernelRole is played by a similarity metric or kernel function, e.g. cosine similarity, radial kernel. The OptimizerRole is played by an optimization algorithm, e.g. quasi-Newton method. The RegularizerRole is played by a regularization function used to keep model complexity low and prevent over-fitting. The LossRole is played by a loss function that defines the penalty of miss-prediction, e.g. hinge loss. The FeatureSensorRole is played by a function that extracts features from the data, while the TargetSensorRole is played by a function that determines the class which the data belongs to by consulting some oracle, e.g. manually annotated data.

A ClassificationAlgorithm takes COMM:MultimediaData and a DataModel as input, and outputs COMM:DigitalData that expresses a structured description of the classified object, a ClassifiedObjectDescriptor.

5.7 Fulfillment of the Requirements

We now discuss how the requirements outlined in Section 4 are satisfied by our proposed modeling of the knowledge acquisition ontology.

By carefully choosing to model OAK on top of COMM, we ensure that OAK supports multimedia data. We have shown how to specialize the core COMM constructs to support text. OAK can easily be extended to accommodate more types of media in the same way.

All the ontologies chosen – DOLCE, COMM and CSO – provide a rich axiomatization of each pattern using first order logic. Moreover, through the semantic annotation pattern, our ontology can be linked to any Web-based domain ontology. These fulfill the semantic and syntactic interoperability requirements, respectively.

The use of ontology patterns ensures a clear separation of concerns. The KA system and subsystem patterns define the core concepts. The decomposition pattern, the annotation pattern and the data modeling pattern each address the homonymous concerns mentioned in Section 2.

The modularity requirement is satisfied, since these patterns form modules in the core of the architecture of the KA ontology. The extensibility requirement is fulfilled in several ways. First, OAK allows accommodating further media types as mentioned above. Second, it is also straightforward to define new types of KA subsystems. Third, it is equally easy to define new types of segment roles and regions. Finally, new types of learning and classification algorithms

can also be added without much effort. The modularity and extensibility of OAK is in great part due to our patterns being grounded in the D&S pattern, which enables adding further contextual knowledge in such a way that will not change the patterns (mainly by defining new roles or parameters), so that legacy descriptions remain valid.

6 Describing the Simple KA System with OAK

Let us revisit the scenario described in Section 2. The developers have decided to throw away any documents that do not contain sports pictures as a prior filtering step. On the remaining documents, a sentence splitter and a named entity recognizer are run to split each document into sentences and tag occurrences of people and location names. Each sentence is then passed to a classifier which generates statements of the type *born_in(person, location)* as output. Figure 7 shows how to semantically describe such a simple KA system.

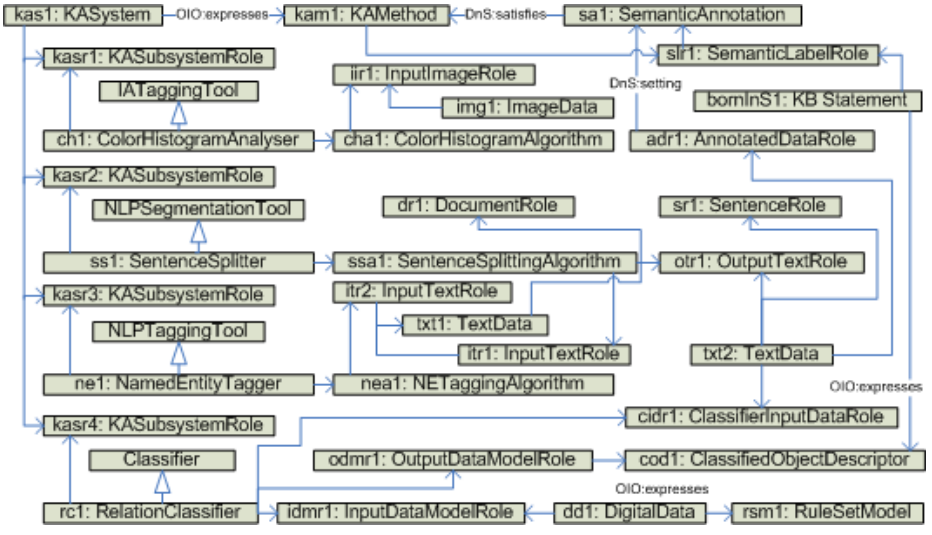

Fig. 7. Semantic description of a simple KA system that populates the KB with *born_in* relations. All the `DnS:plays`, `DnS:defines` and some `DnS:setting` and `OIO:expresses` labels were omitted for clarity. Due to space limitations, the outputs of the color histogram analyser and the named entity tagger were also omitted.

The system defines four subsystem roles, played by a color histogram analyser (an image analysis tagging tool), a sentence splitter (a nlp segmentation tool), a named entity tagger (a nlp tagging tool) and a relation classifier (a classifier). The image analyser takes the image data from the multimedia document and outputs a color histogram, which is used to decide whether the document should be discarded. The input to both the sentence splitter and the named entity tagger

is the text from the whole document (hence playing a `DocumentRole`), whereas the output of the sentence splitter is the text from a sentence (hence playing a `SentenceRole`). The sentence text also plays the `ClassifierInputDataRole` defined by the relation classifier. The latter also takes a set of manually built text extraction rules as input (via the `InputDataModelRole`). The generated KB statements express the `ClassifiedObjectDescriptor` output by the classifier. Finally, the sentence text plays the `COMM:AnnotatedDataRole` and the KB statement the `COMM:SemanticLabelRole` in this `COMM:SemanticAnnotation` setting, which satisfies the KA method expressed by the system.

7 Conclusion and Future Work

In this paper we presented OAK, a well-founded, modular, extensible and multimedia-aware ontology of knowledge acquisition which extends existing foundational and core Semantic Web ontologies. The ontology addresses concrete problems with current state-of-the-art middleware for KA systems development. The advantages of its use include enhanced interoperability among systems, enabling machine reasoning on what the KA system does and how it does it, and enabling semantic discovery and composition of KA services.

The work presented in this paper just scratches the surface on what is possible to represent about KA systems. Nevertheless, as future work, we intend to keep on focusing on those semantic constructs and ontology patterns which are generic enough to constitute the core of KA. One particular pattern that we found lacking in the current version of OAK is that of knowledge fusion, i.e., to be able to describe the situation in which a KB statement is not directly related to a media segment (because it was extracted from it), but rather receives a indirect contribution from the segment towards its existence and validity (e.g., via merging of several extracted facts). We also intend to work on a more detailed characterization of the data modelling pattern, by including constructs such as dataset, instance and pattern that are missing in the current version.

Acknowledgments

This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978.

References

1. Arndt, R., Troncy, R., Staab, S., Hardman, L., Vacura, M.: COMM: Designing a Well-Founded Multimedia Ontology for the Web. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 30–43. Springer, Heidelberg (2007)

2. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to Meet New Challenges in Language Engineering. Natural Language Engineering 10(3/4), 349–373 (2004)
3. Caragea, D., Bao, J., Honavar, V.: A General Strategy for Knowledge Acquisition from Semantically Heterogeneous Data Sources. In: AAAI 2006 Fall Symposium on Semantic Web for Collaborative Knowledge Acquisition (2006)
4. Ferrucci, D., Lally, A.: UIMA: an architectural approach to unstructured information processing in the corporate research environment. Natural Language Engineering 10(3/4), 327–348 (2004)
5. Gangemi, A., Borgo, S., Catenacci, C., Lehmann, J.: Task Taxonomies for Knowledge Content. Metokis Deliverable D 2007 (September 2004)
6. Götz, T., Suhre, O.: Design and implementation of the UIMA Common Analysis System. IBM Systems Journal 43(3), 476–489 (2004)
7. Happel, H.-J., Seedorf, S.: Applications of Ontologies in Software Engineering. In: Second International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), held at the Fifth International Semantic Web Conference (November 2006)
8. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services (November 2004), `http://www.w3.org/Submission/OWL-S/`
9. Maynard, D.: Benchmarking ontology-based annotation tools for the Semantic Web. In: AHM 2005 Workshop on Text Mining, e-Research and Grid-enabled Language Technology (2005)
10. Maynard, D., Yankova, M., Kourakis, A., Kokossis, A.: Ontology-based information extraction for market monitoring and technology watch. In: ESWC 2005 Workshop on End User Aspects of the Semantic Web (May 2005)
11. Oberle, D., Lamparter, S., Grimm, S., Vrandecic, D., Staab, S., Gangemi, A.: Towards Ontologies for Formalizing Modularization and Communication in Large Software Systems. Journal of Applied Ontology 1(2), 163–202 (2006)
12. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM - A Semantic Platform for Information Extraction and Retrieval. Journal of Natural Language Engineering 10(3/4), 375–392 (2004)
13. Puerta, A.R., Neches, R., Eriksson, H., Szekely, P., Luo, P., Musen, M.A.: Toward Ontology-Based Frameworks for Knowledge-Acquisition Tools. In: Eighth Knowledge Acquisition Workshop for Knowledge-Based Systems (1994)
14. Yildiz, B., Miksch, S.: Motivating Ontology-Driven Information Extraction. In: International Conference on Semantic Web and Digital Libraries (2007)
15. `http://t-rex.sourceforge.net`
16. `http://alias-i.com/lingpipe`