A Comprehensive Framework for Semantic Annotation of Web Content

Manuel Fiorelli[1], Maria Teresa Pazienza[2] and Armando Stellato[2]
[1]Department of Civil Engineering and Computer Science, University of Rome, Tor Vergata, Rome, Italy
[2]Department of Enterprise Engineering,University of Rome, Tor Vergata, Rome, Italy

Keywords: Semantic Annotation, Semantic Web, User Interface, Software Engineering.

Abstract: Studies on Semantic Annotation reveal how trying to match heterogeneous requirements leads to divergent methodologies, models and processes for annotation management and exchange. Community efforts towards the development of shared solutions are important to reduce the "entropy" of the field; nonetheless, any agreement on the ultimate annotation system is unlikely to be achieved. We propose a solution to this problem by defining a comprehensive framework, unbound to any specific design/annotation model, and instantiable into concrete system implementations, to meet different requirements. Towards this goal, we commit to fairly general assumptions, valid across disparate systems and not excessively constraining. Firstly, most systems deal with combined management of ontologies and Web content. Secondly, these systems can be described through a common behavioural model, in terms of an assignment of handlers to predetermined events. This behavioural model can be then enriched through progressive levels of specification, thus fostering a convention-over-configuration approach in detailing its characteristics. Then, recurring design fragments can be identified, in order to provide abstractions and specifications for the definition of concrete handlers.

1 INTRODUCTION

In the envisioned Semantic Web (Berners-Lee et al., 2001) the meaning of resources, possibly including services (Payne and Lassila, 2004), is captured through annotations with respect to well-defined ontologies. Formalized knowledge is believed to allow software agents to better interact with Web resources and perform intelligent tasks on behalf of humans, such as buying a vacation package from a virtual travel agency.

The actual deployment of the Semantic Web required further investigation on pragmatic aspects related to the publication and the reuse of disparate knowledge on the Web. This line of development eventually flowed into the Linked Open Data movement which elaborated a collection of best-practices (Heath and Bizer, 2011) aimed at better connecting the Semantic Web to the architecture of the Web. About this topic (Heath, 2009) states that "Linked Data isn't about rebranding the Semantic Web, it's about clarifying its fundamentals".

The interest on data publication and integration is complementary to the idea of annotating traditional information resources (documents, images, audio and video material), since the former provides a sound technological and methodological framework supporting the latter. In line with this idea, the W3C defines the SKOS vocabulary (W3C, 2009) as a means to establish a link between the Linked Open Data cloud and the world of Knowledge Organization Systems (Hodge, 2000), historically employed by museums, libraries and other large organizations to better manage and use their large body of resources.

So far, systems for annotating information content with respect to formalized knowledge have followed different and occasionally contrasting theories. These theories differentiated in many aspects: the primary focus of the annotation (e.g. is the traditional content which needs to be annotated with respect to a generic category, or are specific ontological resources to be grounded on existing documentation?), the granularity of the information to be reported, and the nature of the annotated elements. Therefore, even the offer of Semantic Annotation applications is variegated, and it is often difficult to see all of the requirements for a particular

Fiorelli M., Pazienza M. and Stellato A..
A Comprehensive Framework for Semantic Annotation of Web Content.
DOI: 10.5220/0004545702450252
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2013), pages 245-252
ISBN: 978-989-8565-81-5

usage scenario satisfied by a single system.

We propose here a framework for supporting the development of systems for combined management of ontological knowledge and Web content, including, but not limited to Semantic Annotation Systems. The framework is a subsystem of Semantic Turkey (Pazienza et al., 2012), a fully-fledged environment for knowledge management and acquisition based on RDF technologies (W3C, 2004), with a user interface deployed as a browser extension. Such an offer guarantees to end applications a high level of integration among browsing capabilities, ontology editing and cross-boundary features concerning both.

The rest of the paper is organized as follows. In section 2, we review the state-of-the-art in the field. In section 3, we motivate our work. In section 4, we elicit requirements for our framework, while we discuss its architecture in section 5. Finally, in section 6, we conclude and outline future works.

2 BACKGROUND

We can shortly state that an annotation establishes a link between two resources, asserting that one is "somewhat" about the other. The nature of this association is heavily domain and application dependent. For instance, informal free-text annotations are usually found as comments in a document to drive its edition, while structured annotations are the output of numerous NLP tasks, including named entity recognition and relation extraction. These scenarios depend on different assumptions regarding the nature of the annotations, their granularity, their level of formality and the use, if any, of formal ontologies.

Early works on the annotation of Web resources include Annotea (Kahan and Koivunen, 2001), which aimed at establishing a framework for the collaborative annotation of Web resources. Initially thought for supporting the collaborative development of specifications within the W3C, the project aimed at establishing standards for textual annotations of marked-up documents.

Later initiatives within the bioinformatics community, Annotation Ontology (Ciccarese et al., 2011) and Open Annotation Model (Sanderson and Van de Sompel, 2010), had a wider breath, aimed at the annotation of any media type possibly with respect to a supplied ontology. Those projects flowed into the Open Annotation W3C community project, whose mission is to develop an RDF based model for the annotation of digital artefacts. The Domeo annotation system developed by (Ciccarese, et al., 2012) supports the Annotation Ontology and it is expected to adopt the results of the novel W3C Community Group. With respect to early attempts, it is worth of notice that a shared data model is deemed sufficient, whereas dedicated protocols for querying and manipulating the annotations are no longer considered necessary, thanks to the availability of standards for performing such tasks developed meanwhile (e.g. SPARQL (Prud'hommeaux and Seaborne, 2008)).

In the context of these RDF models an annotation is established though the assertion of at least a statement relating a resource (the target) to another (the body) which represents the desired attachment. In case of Semantic Annotation the latter is found within a formally defined ontology. The choice of a domain/application ontology should reflect the particular point of view behind the annotation process. (Ma et al., 2011) introduced a higher order semantics for capturing the meaning of semantic annotations with respect to the ontological nature of the attached resource and how it is related to the target. They also show how different levels of analysis (i.e. linguistic and semantic) can cooperate, for example to suggest annotations or highlight possible errors.

Beyond the problems inherent to the representation of annotations, there is need for a clear process to create and maintain them. According to (Staab et al., 2000), this process should cope with the evolution of the domain ontology and the presence of mirrors or altered version of the annotated resources.

The production of annotations by human users is often regarded as the bottleneck limiting the scale of the annotation process. (Kiryakov et al., 2004) discussed the design issues related to an holistic system integrating semantic annotations, indexing and (semantically powered) retrieval. They propose the reengineering of state-of-the-art NLP tools for automatically producing semantic annotations with respect to a lightweight upper ontology, called KIMO. The existence of a reference upper ontology, possibly extensible to address domain and application specific needs, is a distinctive feature, since most works assume that semantic annotations are taken against any arbitrary domain ontology. This idea was implemented in the platform KIM (Popov et al., 2003), which was heavily tested for the automatic annotation of news stories.

Finally, (Uren et al., 2006) provided an overview of Semantic Annotation systems by comparing them on the basis of a set of requirements that the authors

consider key-features for the annotation task.

3 MOTIVATION

In the previous section we outlined the main research lines in the field of Semantic Annotation, showing how conflicting requirements demand diverging design decisions, making the definitive annotation system unlikely to appear.

Even a strong agreement on a universal data model for annotations is difficult to achieve: recent proposals focus on widely accepted usage scenarios, therefore failing to satisfy very specific needs.

Divergent methodologies have been proposed to support manual annotation rather than automatic generation of annotations. The latter can benefit, as shown by KIM, of the reuse of state-of-the-art IE tools; this entailing complex integration challenges.

Incompatible design decisions tend to cumulate, leading to very different system architectures and implementations. Therefore, pursuing the goal of realizing the ultimate annotation system appears to be fruitless, while it appears reasonable to aim at the definition of a comprehensive framework supporting alternative designs.

Analogously, designing a framework unbound to any prior assumption makes no sense as well, because an architecture is always based on some grounding which characterizes its offer to the user.

Therefore, our contribution narrows its scope to Semantic Web annotation systems and, in general, any application combining ontological knowledge with Web content. This is a fairly general model which avoids any commitment to specific goals, interaction patterns, methodology (e.g. human labour vs machine learning) or presentation mechanisms.

For what concerns the scope of our architecture (RDF and Web Documents), RDF is by no means the only formalism to capture semantics, though it is now widely spread and there are different W3C recommended vocabularies supporting different modelling needs. The choice for supporting Web documents is mostly a starting point (which does not contradict the generality of the approach), and future evolutions may foresee extensions for other kind of sources, different in format or media type.

4 SYNTHESIS OF REQUIREMENTS

In order to design the architecture of a comprehensive framework for Semantic Annotation, we have both analysed state-of-the-art systems, and taken into consideration principles for their design acquired from literature.

While we take into account the results of the discussed standardization efforts (see section 2), we decided not to commit to a specific model, and have instead an agnostic approach, which starts from the mere annotation acts and allows for the adoption of arbitrary models.

We have thus adopted and incremented the feature classification provided by (Uren et al., 2006), and positioned the class of systems that can be realized with our framework, with respect to those requirements:

- *Standard Formats*: RDF(S), OWL and SKOS for the representation of semantic descriptors; pluggable models for Semantic Annotation (most notable models provided by default as libraries); concrete implementations for different ranges should be provided as component libraries (e.g. offset or XPointer (DeRose et al., 2002) based ranges).
- *User Centred/collaborative Design*: the UI for ontology editing/annotation should be deployed as a web browser extension, while the browser itself hosts the web content. This approach exploits an environment the user is well acquainted with (the browser), while providing new functionalities.
- *Ontology Support*: the framework should support the editing of arbitrary ontologies to be used as domain for annotations;
- *Support of Heterogeneous Document Formats*: it is indeed a desirable feature, though currently our framework is tailored to Web documents; however, this is a technological limitation of the current implementation and not a theoretical choice.
- *Document Evolution*: different choices in the annotation format and in data preservation may be more or less prone to degradation with respect to the evolution of the annotated content; the framework should permit to retain metadata about the target document to be able to detect changes. Option for XPointers guarantees better resilience to changes than plain offsets;

- *Annotation Storage*: as noted in the (Uren et al., 2006), there is no universally winning choice for storing the annotation content: the framework should thus allow annotations to be stored separately from the annotated resources (offline annotations), or to be embedded into them.
- *Automation*: hosting of components for automatic annotation of content should be supported, as well as productive exploitation of their results and suitable interaction with the user for validating and refining these results.
- *Granularity*: both coarse grain and fragment level;

5 ARCHITECTURE

This section is organized as follows: we introduce by first the concepts that have driven the synthesis of the architecture; we then detail specific design choices; finally, we describe the end-user customizability.

5.1 Concept

The proposed framework has to support applications interacting with Web content. (Kahan and Koivunen, 2001) distinguish two strategies to meet this requirement: whether dedicated capabilities are injected into the browser, or into the content provided by a proxy. Our research effort focuses on the first approach, by relying on the extensibility of modern Web browsers to develop the additional capabilities. The user experience with the browser does not change in traditional web navigation, and is only minimally affected when users explicitly trigger one of the extended annotation capabilities. Despite being tightly coupled with the Web, a browser extension is under all aspects a desktop application, with all the advantages deriving in terms of robustness, integration with the local system, and customizability.

In our usage scenario (see), the traditional browser frame for visualizing the web content is complemented with a dedicated panel showing the reference domain model (e.g. an OWL ontology or a SKOS concept scheme).

Possible interactions fall into three main categories, with respect to the resources they affect. The first category comprises the interactions devoted to the navigation of the Web, for instance, activating a hyperlink to reach another Web page. As discussed in section 3, those interactions are completely managed by the hosting browser. The user might as well modify the domain model through interactions falling into the second category. Finally, there are interactions that encompass both realms: for instance, when the user drags a selection of text from a Web page and drops it onto a resource, as common in most annotation systems.

Our work develops from this scenario, by identifying a framework for realizing applications tied to both ontological resources and Web content, and not necessarily limited to semantic annotation.

In our setting, we envision unlimited binding possibilities between annotated content fragments, their originating sources and the resources belonging to the domain model. This should allow, for instance, to generate new ontology individuals *while* annotating their occurrences within web pages, to create and annotate relationships between individuals, etc..

The framework abstracts a collection of events out of gestures involving concrete user interface elements. These events are, to an extent, independent from the underlying presentation mechanism and the supporting technology. The framework dispatches events to suitable handlers, which implement application dependent logic. Event handlers must implement a given signature, whereas there is no prescription on their internal structure.

Within this framework, collections of event handlers define concrete applications, which might be characterized through a variety of (possibly orthogonal) dimensions, including, but not limited to, the following:

- annotation model;
- presentation mechanism;
- relevant ontological resources.

While two applications might differ along a few of those dimensions, they could be very close to each other along others. Therefore, applications are rarely completely orthogonal and in most cases share part of their user interface, behaviour and data management.

The paradigm based on the assignment of handlers to events meets the requirement of minimum commitment to the application goals. Nonetheless, the fact that most applications have overlapping designs would force the developers to implement the same user interfaces and behaviours multiple times. Therefore, a collection of ready-to-use components for common design fragments is required.

5.2 Design

In the forthcoming we refer to a combination of an annotation model, events and related handlers as an *annotation family*, and by a slight abuse of language we will identify possible applications with distinct annotation families. We discuss here three different levels for characterizing a family.

Currently, the framework (see) declares the following events:

- selectionOverResource: when a selection from a Web page is dropped onto an ontological resource
- resourceOverContent: upon gestures for the association of Web content with an ontological resource regardless of their occurrence in the text
- contentLoaded: triggered when Web content is loaded, in order to execute presentation related activities, e.g. highlighting the annotated fragments

So far, this basic set of events provides a core specification, which is sufficient to implement the entire machinery for an annotation system: handlers for the first two events encapsulate the logic for the creation of new annotations, whilst a handler for the third event is in charge of retrieving and properly visualizing annotations for a Web content (and for injecting the code to manage them). For instance, operations such as the deletion of annotations can actually be invoked by code which is injected into the content by handlers intercepting the contentLoaded event, thus leaving the specification of these functions opaque to the framework.

The framework treats different genres of RDF resources (e.g. classes, individuals, and properties) in a uniform manner, by declaring events concerning only generic resources. The uniform treatment of resources entails that the same event might be handled differently on the basis of the target resource. Moreover, applications might foresee the binding of multiple distinct handlers (see) to an event related to a single resource, each handler implementing a distinct way for consuming that event. A mechanism based on *preconditions* allows guarding the execution of handlers on the basis of contextual information. Standard preconditions are also defined and provided with the framework. The basic preconditions include filters based on the role of the resource (e.g. a class, individual etc..), so that a given handler may be activated only for certain resources. The discussion above might be more accessible through an example concerning the event selectionOverResource. As stated previously this event is fired when a selection from a Web page is dropped onto a resource, regardless of its type. Actually, this event might be processed in several ways. By first, a handler may simply annotate an occurrence of that resource within the Web page. Other handling strategies include more complex activities, which are valid only on a subset of the events. For instance, when the target is a class, a handler might create and annotate a new instance for that class, basing on the selected content; otherwise, if the target is a SKOS concept, another handler might create and annotate a narrower concept.

Figure 1: Overview of the Annotation Framework. The Web page is annotated with concepts (insects, plants and pesticides) and relations (isPestOf) from the thesaurus AGROVOC.

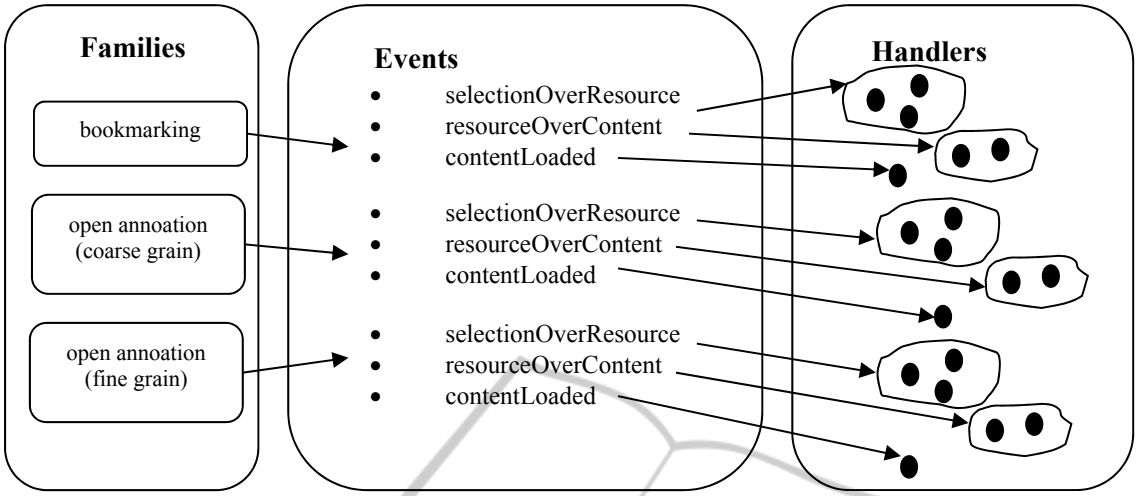

Figure 2: Event Based Architecture.

By following a *convention-over-configuration* approach to design, we provided a further level of specification, consisting in a set of interfaces which, if implemented, can be exploited by the framework on the basis of the previously defined events. The following abstract services can thus be implemented for each family:

- checkAnnotationsForContent(*contentID*)

  This function checks whether a given content source has been annotated. By default, this function is invoked by a framework predefined handler, upon triggering of the contentLoaded event
- getAnnotationsForContent(*contentID*)

  This function returns the annotations taken over a specific content source. Actually, it returns proxies for the annotations (which depend on the model) exposing some framework mandatory fields, such as the id and range of the annotations. The implementation/serialization of these annotation elements is left to the specific family, and must be *consistent* with the other services implemented in the family.

  This function is automatically invoked by the framework after a positive (returned value = true) check performed by the previous function (in the context of a contentLoaded event).
- getAnnotationsForResource(*RDFResource*)

  Analogous to the previous one, this function retrieves all annotations associated to a given RDF resource.

  When constructing a description for a RDF resource in the UI, the framework may exploit this function to produce a list of actionable links to annotated content sources.
- decorateContent(*annotations*)

  This is a client function for injecting elements inside the content, usually to show the annotations which have been previously taken over it.

  A standard text highlighting mechanism for web documents is provided by the system and invoked on the result of a getAnnotationsForContent(), in the context of a contentLoaded event. This mechanism can be overridden by implementing this function with custom content decorators.
- deleteAnnotation(*annotID*)

  This function takes care of removing all the information related to a given annotation. The *standard highlighter* injects calls to this function for each annotation shown on the web document.

The system thus, in line with the *convention-over-configuration* paradigm, allows for high flexibility, while reducing effort and need for detailed specification through massive availability of conventions (and in some cases, implementations).

5.3 Implementation

The annotation framework we presented is embedded in the Knowledge Management and Acquisition Platform Semantic Turkey (Pazienza et

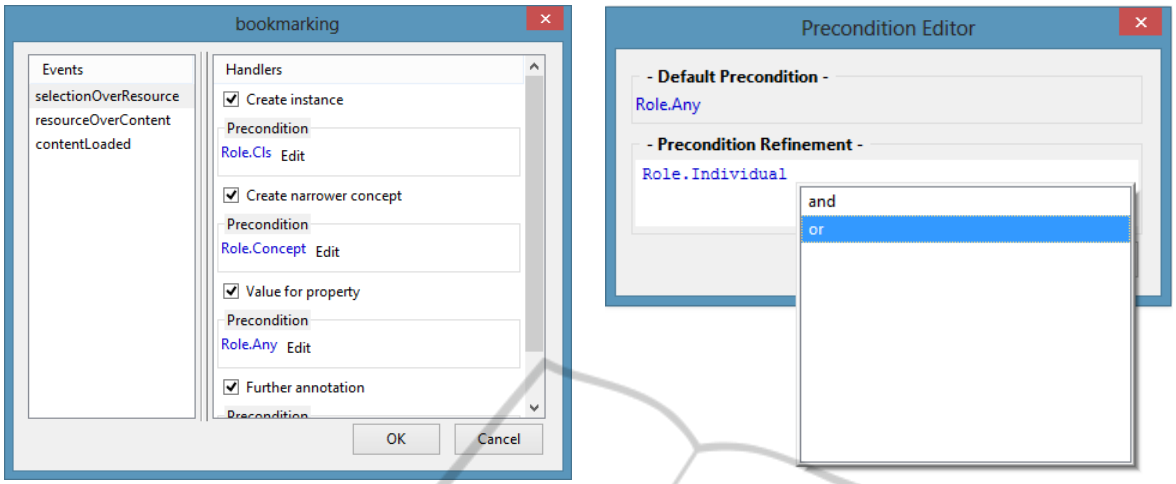

Figure 3: End-user Customization: handlers are enabled for a given event and can then be filtered – by editing their preconditions – when that event is fired.

al., 2012), and comes out-the-box with a few annotation families which differ in the underlying annotation model and, notably, in the tasks they support. The default handlers take into consideration the annotation of atomic ontological resources, and complex activities that are provided as macros, e.g. the creation of new instances, the definition of new subclasses in OWL, or of narrower concepts in SKOS. Semantic Turkey works on a per-project basis, and by default, annotations are stored as further RDF metadata inside the RDF repository of the managed project.

Semantic Turkey allows the deployment of third-party annotation families, additional preconditions, or the enrichment of existing ones by the addition of further handlers. The hosting platform offers to the implementers a wide choice of reusable capabilities. The browser provides technologies for the definition of user interfaces, the manipulation of information resources and the interaction with the Web. An annotation family might exploit them to support inline annotations (included in the document itself), which can then be saved in a updated copy of the web page. An annotation family may depend on core services provided by Semantic Turkey as well as define new ones for dealing with the specifics of its annotation mechanism. There is however no limit to the features that can be provided by adding new services, e.g. dedicated export mechanisms and ontology evolution management.

5.4 End-user Customizability

For any given family of annotations, even final users (i.e. human annotators) may customize their experience to some extent, with no need of coding intervention nor of performing complex configuration on the system.

Concretely, a user can customize a family (see) by enabling only a portion of the annotation functionalities associated to each event, or by refining the preconditions of its associated handlers. Most usage scenarios in fact, only concern with a subset of the possible interactions which a given family may offer, and users may want to enable only those actions which they are using in their setting. Users are normally prompted with the list of suitable handlers (obviously, well presented through appropriated descriptors) after they trigger an event as a consequence of performing an action; as an automatic shortcut, when such a list reduces to a single handler, it is executed without prompting the user.

6 CONCLUSIONS AND FUTURE WORKS

The proposed framework has been experimented in its evolution, through the development of several concrete applications for semantic annotation (Fallucchi et al., 2008; Pazienza et al., 2009; Pazienza et al., 2012). These experiences have helped us in understanding the features which a core framework for semantic annotation should exhibit, and the right trade-off in flexibility which should be granted to system developers, while still benefiting them with concrete support from the software.

Evaluation of frameworks in general is difficult to perform and is based on non-standard

considerations (e.g. the set of features must be decided arbitrarily), which are inherently highly biased by the aspects being put into examination. However, we plan for the future to offer an overall view of the features offered by most notable annotation systems at the current state of the art, and observe if these can be enabled in our framework. By emphasizing the amount of development effort necessary when developing a system with specific features, and the effort that is required to master our framework and build those same features over it, we can obtain a fair map of the improvements and benefits in adopting it. Regarding further evolutions, while the framework seems to us general enough in its basic assumptions, we want to improve it in terms of concrete support to developers. We will thus increment the set of available conventions and create template libraries for recurring annotation patterns. These libraries will provide partial implementations, which can be bound to specific needs through dedicated extension points. Our interest in semi-supervised processes for knowledge acquisition (Fiorelli et al., 2010) motivates our attention to integrating automatic extraction engines and to combining them with proper human interaction, into more virtuous acquisition workflows. We have already explored this approach in (Pazienza et al., 2012), with the development of a text analytics system for the discovery of new semantic relations among concepts belonging to the AGROVOC thesaurus (Caracciolo et al., 2012). We plan to integrate this system to the proposed framework and, in the meanwhile, extend its scope to the projection of arbitrary information onto an ontology.

REFERENCES

Berners-Lee, T., Hendler, J. A. & Lassila, O., 2001. The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American,* 279(5), pp.34-43.

Caracciolo, C. et al., 2012. Thesaurus Maintenance, Alignment and Publication as Linked Data - The AGROVOC Use Case. *International Journal of Metadata, Semantics and Ontologies (IJMSO),* 14 August, 7(1), pp. 65-75.

Ciccarese, P., Ocana, M. & Clark, T., 2012. Open semantic annotation of scientific publications using DOMEO. *Journal of Biomedical Semantics,* Volume 3, pp. 1-14.

Ciccarese, P. et al., 2011. An open annotation ontology for science on web 3.0. *Journal of Biomedical Semantics,* Volume 2, pp. 1-24.

DeRose, S. et al., 2002. *XML Pointer Language (XPointer).* [Online] Available at: http://www.w3.org/TR/xptr/

Fallucchi, F. et al., 2008. *Semantic Bookmarking and Search in the Earth Observation.* s.l., Springer, pp. 260-268.

Fiorelli, M. et al., 2010. *Computer-aided Ontology Development: an integrated environment.* La Valletta, Malta, s.n.

Heath, T., 2009. *Linked Data? Web of Data? Semantic Web? WTF?,* s.l.: s.n.

Heath, T. & Bizer, C., 2011. Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology,* 1(1), pp. 1-136.

Hodge, G., 2000. *Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files.* Washington, DC: Council on Library and Information Resources.

Kahan, J. & Koivunen, M.-R., 2001. *Annotea: an open RDF infrastructure for shared Web annotations.* Hong Kong, Hong Kong, ACM, pp. 623-632.

Kiryakov, A. et al., 2004. Semantic annotation, indexing, and retrieval. *Web Semant.,* #dec#, 2(1), pp. 49-79.

Ma, Y., Lévy, F. & Ghimire, S., 2011. *Reasoning with Annotations of Texts.* s.l., s.n.

Payne, T. R. & Lassila, O., 2004. Semantic web services. *IEEE Intelligent Systems,* 19(1), pp. 14-15.

Pazienza, M. T., Scarpato, N. & Stellato, A., 2009. *STIA: Experience of Semantic Annotation in Jurisprudence Domain.* s.l., IOS Press, pp. 156-161.

Pazienza, M. T., Scarpato, N., Stellato, A. & Turbati, A., 2012. Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. *Semantic Web Journal,* 3(3), pp. 279-292.

Pazienza, M. T. et al., 2012. An Architecture for Data and Knowledge Acquisition for the Semantic Web: The AGROVOC Use Case. In: P. Herrero, H. Panetto, R. Meersman & T. Dillon, eds. *On the Move to Meaningful Internet Systems: OTM 2012 Workshops.* s.l.:Springer Berlin Heidelberg, pp. 426-433.

Popov, B. et al., 2003. *KIM Semantic Annotation Platform.* Florida, USA, Springer-Verlag Berlin Heidelberg, pp. 834-849.

Prud'hommeaux, E. & Seaborne, A., 2008. *SPARQL Query Language for RDF.* [Online] Available at: http://www.w3.org/TR/rdf-sparql-query/

Sanderson, R. & Van de Sompel, H., 2010. *Making web annotations persistent over time.* New York, NY, USA, ACM, pp. 1-10.

Staab, S., Maedche, A. & Handschuh, S., 2000. *Creating Metadata for the Semantic Web—An Annotation Environment and the Human Factor,* s.l.: s.n.

Uren, V. et al., 2006. Semantic annotation for knowledge management: requirements and a survey of the state of the art. *Journal of Web Semantics,* 4(1), pp. 14-28.

W3C, 2004. *Resource Description Framework (RDF).* [Online] Available at: http://www.w3.org/RDF/

W3C, 2009. *SKOS Simple Knowledge Organization System Reference.* [Online] Available at: http://www.w3.org/TR/skos-reference/[Accessed 22 March 2011].