Enriching OWL with Instance Recognition Semantics for Automated Semantic Annotation

Yihong Ding[1], David W. Embley[1], and Stephen W. Liddle[2]

[1] Department of Computer Science
[2] Information Systems Department
Brigham Young Univeristy, Provo, Utah 84602, U.S.A.

Abstract. Although OWL provides a solid basis for many semantic-web applications, it lacks sufficient declarative semantics for instance recognition. This omission prevents OWL from being a satisfactory ontology language for automated semantic annotation. We can resolve this problem by adding to OWL declarations, instance recognition semantics that include external representations and context recognition information for atomic, lexical ontology concepts. Our implementation shows that the new automated annotation prototype system using OWL ontologies with rich instance-recognition semantics not only has high precision and recall, but also overcomes the post-processing problem of linking extracted data to semantic-web ontologies. Our study also shows that the use of instance recognition semantics in ontologies can lead to enhanced knowledge sharing and reuse through the Semantic Web.

1 Introduction

Semantic annotation research is fundamental for the Semantic Web. A *semantic annotation* process adds formal metadata to web pages. This metadata links data in a web page to defined concepts in an ontology. Because machine agents are capable of interpreting data with respect to an ontology, annotated content becomes machine-processable.

Automated semantic annotation is the primary means of adding machine-processable metadata to existing web pages. Several researchers have suggested various ways to automate semantic annotation (e.g., [1, 3, 8, 9, 12]). Each of these approaches has adapted a data-extraction engine to wrap and annotate existing web pages. None of the adapted data-extraction engines, however, was originally designed to produce annotations linking extracted data to an ontology [9, 10]. To provide machine-processable semantics for annotated content, these approaches therefore need to do post-processing to map extracted data to an ontology. Some researchers identify this problem as the "main drawback" of current approaches to automating annotation and suggest the direct use of ontologies in the extraction process to help with semantic annotation [9].

Our ontology-based semantic annotation research shows that this suggested approach does indeed work [4]. There is, however, a hidden problem to address: any system that does not conform to Semantic Web standards will not

be interoperable and thus will not be generally accepted. The current standard (W3C-recommended) Semantic Web ontology language is OWL (Web Ontology Language) [13]. But OWL lacks sufficient declarative semantics for instance recognition, which are needed to extract data directly with respect to ontologies. Our ontology language [4] supports rich declarative instance-recognition semantics, but it is not a standard.

To resolve this issue, we propose an extension of OWL that contains enriched declarative instance-recognition semantics. Because this extension is specifically designed for automated annotation, we call it OWL-AA (OWL for Automated Annotation). Rather than just propose our OWL-AA syntax, the goal of this paper is to illustrate the essence of a sound solution to the problem.

The primary contribution of this work is that we have proposed OWL-AA as a way to extend OWL to provide for automated semantic annotation. With the use of OWL-AA, we present a new semantic annotation model that (1) embeds instance-recognition semantics declarations in ontologies and data-extraction tools and (2) provides enhanced knowledge sharing and reuse through the Semantic Web. Furthermore, as a significant consequence, OWL-AA separates the creation of domain knowledge from the implementation of a processor to use domain knowledge for the purpose of annotating web pages. With OWL-AA, domain experts need not know how to implement extraction and annotation programs, and system developers need not be domain experts.

To explain these contributions, Section 2 presents the details of our proposed instance-recognition semantics, and Section 3 explains their role within the automated semantic-annotation paradigm. Section 4 introduces OWL-AA—the motivation for our choices and its definition and usage. We discuss related work in Section 5 and make concluding remarks in Section 6.

2 Instance Recognition Semantics

Instance-recognition semantics, which we present as *instance semantics recognizers* (*ISR*),[3] are formal specifications that identify instances of a concept in ordinary text. The text may be unstructured, semi-structured, or fully structured. For Semantic Web applications, the concept should be a lexical element of a formal ontology (e.g. concepts such as *date*, *time*, *place*, *location*, *name*, *telephone number*, *email address*, various weights and measures, etc.). Thus, an ISR of an ontology concept (e.g. *Telephone Number*) interprets an instance in a text fragment (e.g. the contact number in "Call me at 222-1234.") to have the intensional meaning of the defined concept (e.g. *Telephone Number*).

We have used information-extraction (IE) ontologies that include ISRs to do data extraction [6] and semantic annotation [4]. In our IE ontologies, we use a *data frame* construct [5] that describes information about a concept—its external and internal representations, its contextual keywords or phrases that may

[3] We avoid the acronym IRS to avoid association with the U.S. Internal Revenue Service.

indicate the presence of an instance of the concept, operations that convert between internal and external representations, and other manipulation operations that can apply to instances of the concept along with contextual keywords or phrases that indicate the applicability of an operation. Thus, a data frame contains ISR declarations together with other elements that are not the focus of this paper.

BedroomNr
  external representation: [1-9]|10
  left context phrase: \b
  right context phrase: .*r(oo)?ms?
  exception phrase: \s.*ba(th)?s?\b.*r(oo)?ms?
  context keywords: b(r|d)s?|bdrms?|bed(rooms?)?
  ...
end

Feature
  external representation: ApartmentFeature.lexicon
  ...
end

Fig. 1. ISR Declarations for `BedroomNr` and `Feature`.

Figure 1 shows partial ISR declarations for two concepts: *BedroomNr* and *Feature*. We use Perl-style regular expressions to declare recognition patterns. Essentially, the ISR declaration of a concept contains two categories of information: self-recognition patterns and context patterns. The *external representation* clause defines self-recognition patterns, which are the typical signatures for any instantiations of a concept. In our example, a valid instantiation of *BedroomNr* must match the regular pattern that describes strings of digits from 1 to 10, and any valid instantiation of *Feature* must match one of the strings declared inside the lexicon file *ApartmentFeature.lexicon*.

Self-recognition patterns by themselves, however, are not enough to precisely recognize a valid instantiation. For example, an integer up to 10 can also be the number of bathrooms or the number of people who can share an apartment. To help resolve potential ambiguities, we use context information. There are two types of context declarations: context phrases and context keywords. A *context phrase* describes contextual information that is directly adjacent (either on the left or the right) to the self-recognition patterns. For example, a valid *BedroomNr* must have a word boundary for its direct left context, and its right context must contain the regular phrase "r(oo)?ms?" with possibly several other words in between. This pattern thus recognizes text content such as "3rms" or "1 large room". A *context keyword* is a word that typically appears close to a self-recognition pattern. In our example, the presence of keywords such as "bds", "bdrms", or "bedrooms" provides a strong hint that we should interpret a small number close to these keywords as a bedroom count.

Another part of an ISR declaration is the *exception* clause, which declares an exception from a previously declared pattern. An *exception* clause negates some external-representation or required-context clause by omitting candidates that match the exception pattern. In our example, the *BedroomNr* ISR declaration will recognize the number "1" in both "1 large room" and "1 bath room". But the exception phrase specifies that we should ignore bathrooms when attempting to recognize a bedroom count.

Although we only present one *external representation* and one set of context declarations for *BedroomNr* in our example, in general there could be many. Each concept can have zero or more self-recognition patterns and each self-recognition pattern can have zero or more context restrictions and exceptions.

3 Automated Semantic Annotation

We now show how we use declarative ISRs for automated semantic annotation. Our experiments have shown that by using ISR declarations, we can successfully annotate many web pages automatically. The following text is a real-world example taken from *Sale Lake City Weekly*.[4]

> **CAPTIAL HILL** Luxury 2 bdrm 2 bath, 2 grg, w/d, views, 1700 sq ft. $1250 mo. Call 533-0293

We illustrate the annotated results in Figure 2. Based on the ISR declarations in Figure 1, the annotation system can automatically recognize the first "2" to be a bedroom number, while the second and third "2" are not since neither of them satisfies the right-context requirement. Similarly, the system can automatically recognize the second "2" to be a bathroom number. The third "2" should be part of "2 grg"—a two-car garage. If we have an ISR regular-expression declaration for it in our *ApartmentFeature.lexicon*, it will be recognized along with "w/d"—washer and dryer—as an apartment feature. Based on the same ISR processing, we can annotate the remaining instances of the *aptrent* ontology—"1250" as a monthly rate, "533-0293" as a contact phone number.

Figure 3 shows a screen shot of annotated results from our ISR-based, automated semantic annotation demo.[5] After annotating the page using ISR declarations, our demo lets a user move the mouse over annotated text, such as the "2" in the text, and see its detailed annotation information—its ontology name, concept name, record number, and multiple-value index. As Figure 3 shows, the highlighted number "2" is about a bedroom number that has been defined in the *aptrent* ontology. The system also automatically produces a table that contains all the annotated data with respect to every record entry in a web page. For some concepts (such as *Feature*) that may contain multiple values in one record, there is a clickable button. By clicking the button we can retrieve a sub-table that contains detailed information. In our example, when we click the "Show" button under *Feature*, "2 grg" and "w/d" appear as Figure 3 shows.

[4] http://www.slweekly.com/
[5] http://www.deg.byu.edu/

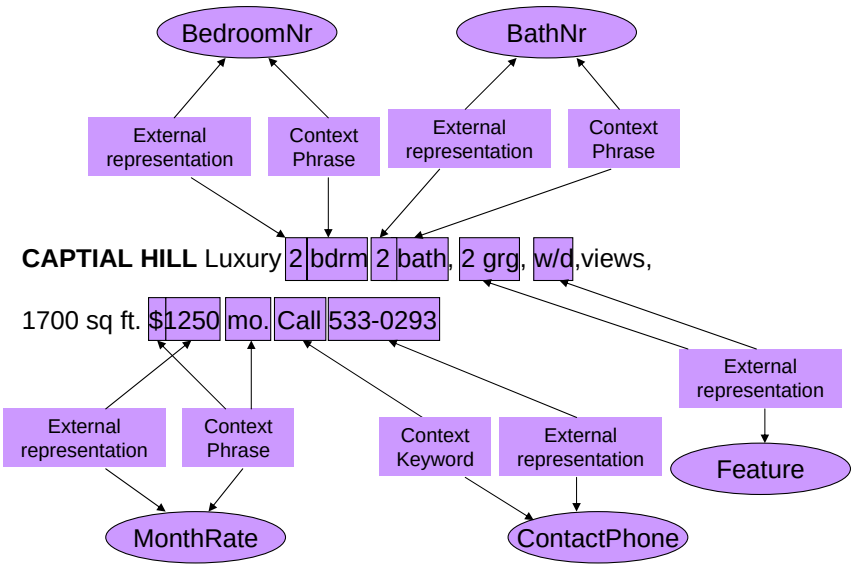

Fig. 2. Example of Apartment-Rental Annotation Results.

A significant feature of ISR-based annotation is the exclusion of document layout information from ISR declarations. Many current wrappers use web-page layout information, which allows the wrapper to do fast and accurate data recognition (when a target layout matches the defined layout) [10]. However, the primary purpose of layout is for displaying, not for describing data. More importantly, relying on layouts prevents an annotation system from working continually on a web page when its layout changes. A reality of the current web is that page layouts do change periodically, and rewriting or regenerating wrappers to accommodate these changes is costly. Hence, layout-independence, which is also called *resiliency* [10], is a preferred property of automated semantic annotation approaches. With the property of resiliency, an annotation system can automatically be applied to new web pages, independent of their specific layouts. Our premise is that the layout of information is not as important as its content. Our ontology-based, data-extraction engine is resilient, and it generally works well [6, 10].

4 OWL-AA: OWL for Automated Annotation

The importance of ISR declarations to semantic annotation, as well as to the Semantic Web in general, is still not widely recognized. Semantic Web researchers besides us (e.g, [2] and [11]) have also observed that the current Semantic Web ontology-language standard, OWL, is not sufficient to resolve all issues for the

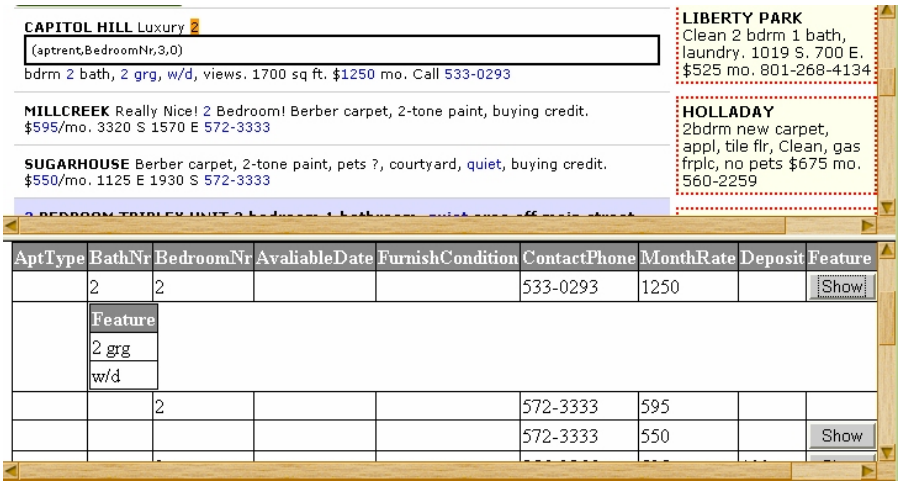

Fig. 3. Screen-shot of ISR-based Annotation Demo.

Semantic Web. We need various extensions of OWL for particular application scenarios. To leverage automated semantic annotation, we thus propose OWL-AA based on our experience with ISR declarations.

4.1 OWL-AA Description

Figure 4 shows the complete RDF schema of OWL-AA. The RDFS class *ISR* provides an abstraction mechanism for grouping different types of instance recognition semantics. Like RDF or OWL class declarations, every ISR declaration is associated with a set of individuals, each of which can have an *ISRvalue* that explicitly provides its content. Each *ISRvalue* has an XML string data type. An ISR individual has no conceptual meaning beyond the represented pattern itself. An individual obtains its conceptual meaning only after it is explicitly bound to an ontology concept. For example, the regular-expression pattern "[1-9]|10" does not have any conceptual meaning beyond the pattern itself. Only after it is bound to a concept such as *BedroomNr*, does it become conceptually meaningful. Two ISR individuals may have the same ISR value, but still mean something different by being assigned to different concepts.

There are three subclasses of *ISR*: *ExternalRepresentation*, *ContextualRepresentation*, and *Exception*. *ExternalRepresentation* declarations formalize the body of an ISR individual. *ContextualRepresentation* declarations formalize the contextual constraints to the body of an ISR individual. *Exception* declarations formalize exceptions to ISR declarations.

An *ExternalRepresentation* could be either a *RegularExpression* declaration or a *LexiconList* declaration. The reason for two forms is simple convenience. Sometimes it is easier to enumerate matching strings (e.g. country names), and

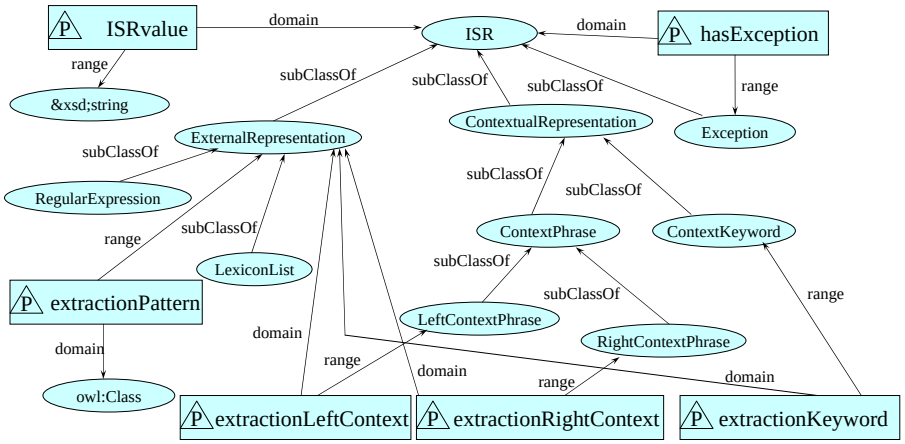

Fig. 4. RDFS Graph of OWL-AA.

other times it is easier to write a descriptive formula (e.g. the integers between 1 and 10,000,000).

ContextualRepresentation also has two subclasses, *ContextPhrase* and *ContextKeyword* that declare context phrases and context keywords respectively. As a further subdivision, a *ContextPhrase* could be either a *LeftContextPhrase* or a *RightContextPhrase*.

Five properties apply when declaring an ISR. An *extractionPattern* binds an *ExternalRepresentation* to an *owl:Class*. An *extractionLeftContext*, an *extractionRightContext*, and an *extractionKeyword* respectively bind a *LeftContextPhrase*, a *RightContextPhrase*, and a *ContextKeyword* to an *ExternalRepresentation*. The *hasException* property binds an *Exception* to an *ISR*.

We now show how to declare our example in Figure 1 using OWL-AA. We start with a standard OWL ontology that describes the apartment rental domain. We then attach OWL-AA statements to respective concepts, such as *BedroomNr*.

At the beginning of the apartment rental OWL ontology, we first add a new *owlaa* namespace.

```
xmlns:owlaa="http://www.deg.byu.edu/OWL-AA#"
```

Within the OWL Class *BedroomNr*, we next add a new property *owlaa:extractionPattern* that associates an ISR declaration to *BedroomNr*.

```
<owl:onProperty rdf:resource="owlaa:extractionPattern" />
<owl:hasValue rdf:resource="BedroomNr-1" />
```

Then we can declare the ISR value and contextual restriction associated to this external representation as follows.

```
<owlaa:RegularExpression rdf:ID="BedroomNr-1"/>
```

```
    <owlaa:ISRvalue rdf:datatype="&xsd;#string">[1-9]|10</owlaa:ISRvalue>
    <owlaa:extractionLeftContext rdf:resource="#leftContext-1"/>
    <owlaa:extractionRightContext rdf:resource="#rightContext-1"/>
    <owlaa:extractionKeyword rdf:resource="#contextKeyword-1"/>
  </owlaa:RegularExpression>
```

Finally, we add context phrases, context keywords, and exceptions as follows.

```
  <owlaa:LeftContextPhrase rdf:ID="leftContext-1"/>
    <owlaa:ISRvalue rdf:datatype="&xsd;#string">\b</owlaa:ISRvalue>
  </owlaa:LeftContextPhrase>
  <owlaa:RightContextPhrase rdf:ID="rightContext-1"/>
    <owlaa:ISRvalue rdf:datatype="&xsd;#string">
        .*r(oo)?ms?</owlaa:ISRvalue>
    <owlaa:hasException rdf:resource="#exception-1"/>
  </owlaa:RightContextPhrase>
  <owlaa:ContextKeyword rdf:ID="contextKeyword-1"/>
    <owlaa:ISRvalue rdf:datatype="&xsd;#string">
        b(r|d)s?|bdrms?|bed(rooms?)?</owlaa:ISRvalue>
  </owlaa:ContextKeyword>
  <owlaa:Exception rdf:ID="exception-1"/>
    <owlaa:ISRvalue rdf:datatype="&xsd;#string">
        \s.*ba(th)?s?\b.*r(oo)?ms?</owlaa:ISRvalue>
  </owlaa:Exception>
```

4.2 Discussion

Syntactically, OWL-AA is attachment-independent with respect to OWL. We add OWL-AA statements without modifying a single line of an existing OWL ontology. We can obtain the original OWL ontology by removing these attached OWL-AA statements.

Semantically, OWL-AA is also attachment-independent with respect to OWL. The addition of ISR declarations to ontology concepts only enriches their details without changing the meaning at the conceptual level. An OWL ontology enriched by OWL-AA can, for example, be used for reasoning as usual. A regular reasoning engine can simply skip attached OWL-AA statements since they are syntactically attachment-independent.

In order to process OWL-AA for automated annotation, we have implemented a syntax convertor that can convert an OWL-AA ontology to our IE ontology. We can then apply our ontology-based annotation tool using the converted ontology. In our implementation, we have used Jena, a standard Semantic Web framework, to help with this ontology conversion. Since we have captured all of an IE-ontology ISR declaration in OWL-AA, OWL-AA ontologies can work as well as the original data-extraction ontologies for automated annotation tasks.

5 Related Work

The idea of ISR declarations is not new. Many researchers have coded ISR declarations into their procedures, such as wrappers. IBM's UIMA (Unstruc-

tured Information Management Architecture)[6] is a typical approach that has addressed the issues of sharing and reusing ISRs in procedures. The text analysis engines (TAEs) in the UIMA project are typical examples of small reusable ISR processors. Each TAE does a particular data analysis on source documents, for example, pulling out chemical names and their interactions [7].

Although OWL is the current W3C-recommended web-ontology language, researchers have pointed out that OWL is not sufficient for all Semantic Web work. There have been several proposed extensions of OWL. Among them, two extensions are closer to our work than others: C-OWL and OWL-Eu. Context OWL (C-OWL) [2] provides an extension for ontology mapping that localizes ontology content to allow for limited and totally controlled forms of global visibility. Similar to ours, C-OWL declarations can be expressed independent of ordinary OWL ontologies. OWL-Eu [11] enriches OWL with customized datatypes. The authors point out that "many potential users will not adopt OWL unless [the data-type-support problem] is overcome." In our experience, support for customized datatypes could also improve the performance of data extraction. Unlike our extension, OWL-Eu modifies existing OWL constructs to support customized data types. OWL-AA, on the other hand, is an attachment-independent extension. As a result, OWL-AA appears to be fully compatible with OWL-Eu too, so we could use them together if desired. With enriched customized data types, we expect that we could declare richer ISR representations by attaching our OWL-AA extension to OWL-Eu ontologies.

6 Concluding Remarks

Automated semantic annotation is an important and fundamental problem for the Semantic Web. The key to automated annotation is the ISR declaration. Our OWL-AA extension augments OWL to formalize ISR declarations. OWL-AA is fully compatible with ordinary OWL, and fully attachable and detachable from standard OWL ontologies. OWL-AA does not introduce new complexity and decidability issues into OWL. Our prototype implementation demonstrates that the OWL-AA extensions work well for our automated semantic annotation system.

References

1. L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo, "Automatic annotation of data extracted from large web sites," *Proc. Sixth International Workshop on the Web and Databases (WebDB 2003)*, pp. 7-12, San Diego, California, June 2003.
2. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt, "Contextualizing ontologies," *Journal of Web Semantics*, vol. 1, no. 4, pp. 325–343, October 2004.

[6] http://www.research.ibm.com/UIMA/

3. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien, "A case for automated large scale semantic annotations," *Journal of Web Semantics*, vol. 1, no. 1, pp. 115–132, December 2003.
4. Y. Ding, D.W. Embley, and S.W. Liddle, "Automatic creation and simplified querying of Semantic Web content: An approach based on information-extraction ontologies," *Proc. First Asian Semantic Web Conference (ASWC 2006)*, pp. 400-414, Beijing, China, September 2006.
5. D.W. Embley, "Programming with data frames for everyday data items," *Proc. 1980 National Computer Conference*, pp. 301-305, Anaheim, California, May 1980.
6. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data & Knowledge Engineering*, vol. 31, no. 3, pp. 227-251, November 1999.
7. D. Ferrucci and A. Lally, "Building an example application with the Unstructured Information Management Architecture," *IBM Systems Journal*, vol. 43, No. 3, pp. 455–475, March 2004.
8. S. Handschuh, S. Staab, and F. Ciravegna, "S-CREAM Semi-automatic CREAtion of Metadata," *Proc. European Conference on Knowledge Acquisition and Management (EKAW-2002)*, pp. 358–372, Madrid, Spain, October, 2002.
9. A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic annotation, indexing, and retrieval," *Journal of Web Semantics*, vol. 2, no. 1, pp. 49–79, December 2004.
10. A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, and J.S. Teixeira, "A brief survey of web data extraction tools," *SIGMOD Record*, vol. 31, no. 2, pp. 84-93, June 2002.
11. J.Z. Pan and I. Horrocks, "OWL-Eu: Adding customised datatypes into OWL," *Journal of Web Semantics*, vol. 4, no. 1, pp. 29–39, January 2006.
12. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, "MnM: Ontology driven tool for semantic markup," *Proc. Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, pp. 43–47, Lyon, France, July 2002.
13. W3C (World Wide Web Consortium) *OWL web ontology language reference*, URL: http://www.w3.org/TR/owl-ref/.