Talking OWLs: Towards an Ontology Verbalizer

Graham Wilcock

University of Helsinki
00014 Helsinki, Finland
graham.wilcock@helsinki.fi

Abstract. The paper describes on-going work on an ontology verbalizer which can provide spoken summaries and explanations of the information specified in ontologies. The approach combines semantic web techniques with natural language generation and text-to-speech.

1 Previous work on generation from ontologies

Referring to ontologies formalized in Ontolingua, Aguado *et al.* say:

> Our experience shows that domain experts and human final users do not understand formal ontologies codified in such languages even if such languages have a browser and a graphic user interface to display the ontology content. [1]

They describe a system that translates the ontology into natural language to help users understand it. To map from domain concepts to linguistic representations they use the Generalized Upper Model, based on the Penman Upper Model [2], as a "linguistic ontology". Surface realization is done with KPML.

Frohlich and Riet [4] describe domain independent tools for generation based on "using different ontologies to represent the domain knowledge for different tasks of the generation process." Like [1], they have a domain-specific layer at the top and a domain-independent layer based on the Penman Upper Model at the bottom, with KPML for surface realization.

These earlier projects used languages and tools such as Ontolingua, Penman Upper Model, LISP, LOOM and KPML. Although we can now use Java, XML, RDF and OWL, we still need to help users to understand the ontologies.

2 Generating summaries from RDF

In XML-based natural language generation [9], [10], [11], a pipeline of XSLT transformations implements the sequence of processing stages in an orthodox pipeline architecture for natural language generation. At the start of the pipeline, XSLT template-based generation creates an XML text plan tree whose leaves are domain concept messages. The text plan tree is transformed by the microplanning stages into an XML text specification tree whose leaves are linguistic phrase

specifications. The XSLT processors are embedded in Java, using SAX events to pass XML content efficiently down the pipeline.

XML-based generation has been used in a spoken dialogue system [6]. For spoken output, the final realization stage produces JSML (Java Speech Markup Language) [7] which is XML-based. The JSML is passed to FreeTTS [8], a speech synthesizer implemented entirely in Java.

XML-based generation can naturally be used for generation from RDF. A prototype implementation uses Jena [5] to feed content from RDF into the XSLT pipeline. Jena includes an RDF parser (ARP), an RDF query language (RDQL), and support for persistent storage of RDF models in relational databases.

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'>
  <rdf:Description rdf:about='http://somewhere/JohnSmith/'>
    <vcard:FN>John Smith</vcard:FN>
    <vcard:N rdf:nodeID='A0'/>
    <vcard:EMAIL rdf:nodeID='A1'/>
  </rdf:Description>
  <rdf:Description rdf:nodeID='A1'>
    <rdf:value>John@somewhere.com</rdf:value>
    <rdf:type rdf:resource='http://www.w3.org/2001/vcard-rdf/3.0#internet'/>
  </rdf:Description>
  <rdf:Description rdf:nodeID='A0'>
    <vcard:Family>Smith</vcard:Family>
    <vcard:Given>John</vcard:Given>
  </rdf:Description>
</rdf:RDF>
```

Fig. 1. Example RDF description from the Jena tutorial

The simple RDF description in Figure 1 is taken from the Jena tutorial [5]. It describes a specific person (John Smith), not the general class of persons, and it uses the RDF encoding for vCard (visiting card) personal information [13]. If the natural language generator were limited to the information given explicitly in the RDF representation, it might produce something like Example 1.

Example 1.
This is a description of 'http://somewhere/JohnSmith/'. The description includes 3 items: 'vcard:FN', 'vcard:N' and 'vcard:EMAIL'.

The value of 'vcard:FN' is 'John Smith'.

The description of 'vcard:N' includes 2 items: 'vcard:Family' and 'vcard:Given'. The value of 'vcard:Family' is 'Smith'. The value of 'vcard:Given' is 'John'.

The description of 'vcard:EMAIL' includes a value and a type. The value is 'John@somewhere.com'. The type is 'http://www.w3.org/2001/vcard-rdf/3.0#internet'.

However, the generator can exploit the use of vCard by providing predefined XSLT text plan templates for vCard, following the domain-specific approach of *shallow generation* [3], [10]. The values from the RDF representation are copied into the slots in the text plan template. By using knowledge about vCard, the generator can create a much better text plan equivalent to Example 2.

Example 2.
This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. John Smith's given name is 'John'. John Smith's family name is 'Smith'. John Smith's email address is 'John@somewhere.com'. John Smith's email address is type 'internet'.

Of course, natural language generation can produce something more natural than this. The referring expressions stage of the generator can convert the text plan into a text specification equivalent to Example 3.

Example 3.
This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. His given name is 'John'. His family name is 'Smith'. His email address is 'John@somewhere.com'. It is 'internet' type.

Further, by performing sentence aggregation, the microplanning stages of the generator can produce a text specification equivalent to Example 4.

Example 4.
This is a description of John Smith identified by 'http://somewhere/JohnSmith/'. His given name is 'John' and his family name is 'Smith'. His email address, which is 'internet' type, is 'John@somewhere.com'.

3 Generating explanations from DAML+OIL

The approach described in Section 2 is a form of shallow generation. One of the ideas in shallow generation [3] is to build domain-specific and task-specific generators, and not to attempt general solutions.

Naturally, shallow generation is compatible with a domain-specific ontology, but at first sight it seems incompatible with more general ontologies. However, Aguado *et al.* [1] claim that their rhetorical schemas represent standard patterns of scientific discourse, and they identified a number of stereotypical paragraph templates including definitions, comparisons, examples and classifications. If a small number of explanation schemas are sufficient to generate explanations from ontologies, then shallow generation can be used. This is an important point, to be investigated further.

The approach used for RDF can again be extended to process DAML+OIL. Jena [5] provides Java methods to read a DAML+OIL ontology and load it as a Jena model. There are also Jena methods to list all the ontology classes and to list all the properties. This provides a starting point for verbalising the ontology

contents, but raw lists of classes and properties are very difficult to understand. In order to generate something which is an *explanation* of the ontology, the classes and properties need to be organised into meaningful groups.

This is on-going work. The RDF examples, the DAML+OIL processing, and the use of ontologies in spoken dialogue systems are discussed further in [12]. The current prototype uses RDF and DAML+OIL with Jena 1. Future work will use RDF and OWL[1] with Jena 2.

References

1. G. Aguado, A. Bañón, J. Bateman, S. Bernardos, M. Fernández, A. Gómez-Pérez, E. Nieto, A. Olalla, R. Plaza, and A. Sánchez. Ontogeneration: Reusing domain and linguistic ontologies for Spanish text generation. In *Proceedings of ECAI-98 Workshop on Applications of Ontologies and Problem-solving Methods*, pages 1–10, Brighton, 1998.
2. J. Bateman, R. Kasper, J. Moore, and R. Whitney. A general organization of knowledge for natural language processing: the PENMAN upper model. Technical report, USC/ISI, 1990.
3. S. Busemann and H. Horacek. A flexible shallow approach to text generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 238–247, Niagara-on-the-Lake, Ontario, 1998.
4. M. Fröhlich and R. van de Riet. Using multiple ontologies in a framework for natural language generation. In *Proceedings of ECAI-98 Workshop on Applications of Ontologies and Problem-solving Methods*, pages 67–77, Brighton, 1998.
5. HP Labs. Jena Semantic Web Toolkit. http://www.hpl.hp.com/semweb/jena.htm, 2003.
6. K. Jokinen and G. Wilcock. Confidence-based adaptivity in response generation for a spoken dialogue system. In *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, pages 80–89, Aalborg, Denmark, 2001.
7. Sun Microsystems. Java Speech Markup Language Specification, version 0.6. http://java.sun.com/products/java-media/speech/, 1999.
8. Sun Microsystems. FreeTTS: A speech synthesizer written entirly in the Java programming language. http://freetts.sourceforge.net/, 2002.
9. G. Wilcock. Pipelines, templates and transformations: XML for natural language generation. In *Proceedings of the 1st NLP and XML Workshop*, pages 1–8, Tokyo, 2001.
10. G. Wilcock. XML-based Natural Language Generation. In *Towards the Semantic Web and Web Services: XML Finland 2002 - Slide Presentations*, pages 40–63, Helsinki, 2002.
11. G. Wilcock. Integrating Natural Language Generation with XML Web Technology. In *Proceedings of the Demo Sessions of EACL-2003*, pages 247–250, Budapest, 2003.
12. G. Wilcock and K. Jokinen. Generating responses and explanations from RDF/XML and DAML+OIL. In *Knowledge and Reasoning in Practical Dialogue Systems*, pages 58–63. IJCAI-2003, Acapulco, 2003.
13. World Wide Web Consortium. Representing vCard Objects in RDF/XML. http://www.w3.org/TR/vcard-rdf, 2001.

[1] I thank Lauri Carlson for the phrase "Talking OWLs" in the title.