Integrating Natural Language Generation with XML Web Technology

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

Abstract

The paper describes a software demo integrating Natural Language Generation (NLG) techniques with recent developments in XML web technology. The NLG techniques include a form of template-based generation, transformation of text plan trees to text specification trees, and a multi-stage pipeline architecture. The web technology includes XSLT transformation processors, an XML database, a Java servlet engine, the Cocoon web publishing framework and a Java speech synthesizer. The software is all free, open-source.

1 Introduction

XML-based techniques for natural language generation are described by Wilcock (2001), based on practical experience in developing an XML-based generation component for a spoken dialogue system (Jokinen and Wilcock, 2001).

The basic approach in the earlier work was to construct a pipeline of XSLT transformations corresponding to the different NLG processing tasks. The current work extends this by putting the XSLT pipelines inside an Apache Cocoon XML server, and by putting the lexicon in an Apache Xindice native XML database. Also, the earlier speech synthesizer is replaced by FreeTTS, a speech synthesizer implemented in Java. Some of these extensions are described by Wilcock (2002).

The paper is organised as follows. Section 2 briefly summarizes the main tasks of NLG and Section 3 then describes how these tasks can be implemented using recent XML web technology. Section 4 gives details of a software demo based on this approach, showing a concrete example.

2 Brief Summary of NLG

The most common design for NLG systems is a pipeline architecture. The version described by Reiter and Dale (2000) has the following modules and tasks:

- Text Planning
  – content determination
  – discourse structuring
- Microplanning
  – lexicalization
  – referring expression generation
  – aggregation
- Realization
  – linguistic realization
  – structure realization

The interface between text planning and microplanning is a text plan, a tree whose leaves are domain-specific concept messages. The interface between microplanning and realization is a text specification, another tree whose leaves are linguistic phrase specifications. The various stages of the NLG pipeline perform transformations on the text plan tree and the text specification tree.

The status of *template-based generation* has been debated by NLG researchers (Becker and Busemann, 1999), but if it means "making extensive use of a mapping between semantic structures and representations of linguistic surface structures that contain gaps" (van Deemter et al., 1999), then it is a good way to create initial text plan trees that contain gaps to be filled later when the concept messages are turned into phrase specifications in the text specification tree.

3 XML Web Technology

As described by Wilcock (2001), XML-based NLG can be performed by a sequence of XSLT transformations. Template-based generation of the initial text plan tree can be done by XSLT templates. Subsequent transformations of the text plan tree during microplanning can be done by specialised XSLT transformations to produce the text specification tree.

Java-based XSLT processors such as Xalan (Apache XML Project, 2003c) can be embedded in Java servlets and executed in a servlet engine such as Tomcat (Apache Jakarta Project, 2003). Moreover, it is now possible to embed complete sequences of XSLT transformations, organised as pipelines, inside Cocoon (Apache XML Project, 2003b). Cocoon runs as a web application inside Tomcat, and offers high-performance XSLT processing and scalability.

As Cocoon supports re-configurable pipelines of XSLT transformations, pipelines for different NLG requirements can be set up. For example, Finnish and English generation pipelines can include the same XSLT transforms for text planning stages which are domain-specific, but have different sequences of transforms in the microplanning and realization stages which are language-specific (Wilcock, 2002).

The Apache Xindice native XML database (Apache XML Project, 2003a) also runs as a web application in Tomcat, and can be used together with Cocoon. Xindice provides suitable support for a lexicon in XML form, which can be indexed to meet different processing requirements. For generation, words can be indexed by concepts instead of by spelling.

FreeTTS (Sun Microsystems, 2002) is a speech synthesizer implemented in Java, which accepts JSML, Java Speech Markup Language (Sun Microsystems, 1999). Because it is Java-based FreeTTS can be embedded in Java servlets, and as JSML is XML-based the XSLT pipelines can easily produce JSML output. However, the current version of FreeTTS has some restrictions: JSML markup is accepted but not actually applied to the speech output, and there are only English and MBROLA voices.

4 The Demonstration System

The demonstration system performs bilingual generation of responses, in Finnish and English, as part of a Helsinki bus timetable enquiry system. The responses depend on the dialogue context and can vary from full sentences to short elliptical phrases. The system demonstrates only generation - it does not include speech recognition, language understanding or dialogue management components.

4.1 Input: an Agenda

The starting point is an *agenda*, a set of concepts marked with Topic and NewInfo tags as explained by Jokinen and Wilcock (2001). In the dialogue system this is given by the dialogue manager. In the demo, a number of different starting agendas are provided, and their contents can be changed as desired.

```
<AG id="AG1" type="transcription"
  timeline="Timeline1">
<Anchor id="AG1_anchor1" offset="1"/>
<Anchor id="AG1_anchor2" offset="2"/>
<Annotation id="AG1_ann1" type="Concept"
  start="AG1_anchor1" end="AG1_anchor2">
  <Feat name="score">0.0123</Feat>
  <Feat name="name">route</Feat>
  <Feat name="status">new-info</Feat>
  <Feat name="value">81</Feat>
</Annotation>
<Annotation id="AG1_ann2" type="Concept"
  start="AG1_anchor1" end="AG1_anchor2">
  <Feat name="score">0.0123</Feat>
  <Feat name="name">time</Feat>
  <Feat name="status">new-info</Feat>
  <Feat name="value">11:37</Feat>
</Annotation>
<Annotation id="AG1_ann3" type="Concept"
  start="AG1_anchor1" end="AG1_anchor2">
  <Feat name="score">0.1234</Feat>
  <Feat name="name">place</Feat>
  <Feat name="constr">depart</Feat>
  <Feat name="status">topic</Feat>
  <Feat name="value">herttoniemenranta
  </Feat>
</Annotation>
</AG>
```

Figure 1: An Agenda

The agenda is represented in the full dialogue system as an annotation graph (Bird et al., 2001), as in Figure 1. This example shows an agenda for a response following the enquiry *When does the* *next bus leave from Herttoniemenranta?* The concept for departure place is marked as topic, and the concepts for route number and departure time are marked as new information. The response, which will be generated step-by-step in the next few sections, will be *Number 81 leaves from there at 11:37.*

4.2 Text Planning

In text planning, the content determination stage simply extracts the concepts from the annotation graph. Because the dialogue manager has already decided the relevant concepts and put them in the agenda, no other content determination is needed.

The discourse structuring stage creates a text plan tree using the form of template-based generation described by Wilcock (2001). In the dialogue-oriented system, the text plan is called a response plan.

```
<ResponsePlan>
  <Message>
    <type>NumFromDepMsg</type>
    <concept info="NewInfo">
      <type>bus-number</type>
      <value>81</value>
    </concept>
    <concept info="NewInfo">
      <type>departure-time</type>
      <value>11:37</value>
    </concept>
    <concept info="Topic">
      <type>departure-place</type>
      <value>herttoniemenranta</value>
    </concept>
  </Message>
</ResponsePlan>
```

Figure 2: A Text Plan (Response Plan)

The text plans are XML tree structures containing variable slots, which will be filled in later by the microplanning stages. In the text planning stage, the concepts from the agenda are copied directly into the appropriate slots. So in Figure 2, the concepts are the same as in Figure 1, only the format has changed.

Note that in this example there is only one message, which is typical in spoken dialogue responses. In multi-paragraph text generation there would be large numbers of messages. Note also that the departure place is Topic, and the bus number and time are NewInfo.

In the demonstration system, tracing can be switched on so that the text plan is displayed.

4.3 Microplanning

The processing during microplanning is done by a sequence of XSLT transformations, as described by Wilcock (2001). The text plan tree is replaced by a text specification tree, here called a response specification.

At later stages of the pipeline, further information is added to the tree or nodes in the tree are replaced by new nodes. In the referring expression stage of microplanning, domain concepts are replaced with linguistic referring expressions.

```
<ResponseSpec>
  <PhraseSpec>
    <head>leave</head>
    <subject>
      <head>number</head>
      <attribute>81</attribute>
    </subject>
    <adverbial>
      <advtype>from-place</advtype>
      <head>from</head>
      <object>
        <head>there</head>
      </object>
    </adverbial>
    <adverbial>
      <advtype>at-time</advtype>
      <head>at</head>
      <object>
        <attribute>11:37</attribute>
      </object>
    </adverbial>
  </PhraseSpec>
</ResponseSpec>
```

Figure 3: A Text Specification

In the text specification in Figure 3 the concepts of Figure 2 have been replaced by linguistic specifications. In the lexicalization stage, the `<head>` words are inserted with their dependents, using a form of head-dependency structure.

In the referring expressions stage, the departure-place concept which was marked as Topic in Figure 2 has been pronominalized as *there*. If the same departure-place concept were marked as NewInfo, it would be realized by the actual text value of the departure placename.

In the demonstration system, tracing can also be switched on so that the generated text specification is displayed.

4.4 Realization

The realization stage produces output which is marked up in Java Speech Markup Language (Sun Microsystems, 1999).

```
<jsml lang="en">
  <div type="sent">
    number
    <sayas class="number">81</sayas>
    leaves from there at
    <sayas class="time">11:37</sayas>
  </div>
</jsml>
```

Figure 4: Speech Markup

In Figure 4, the `<head>` words of Figure 3 provide the main content. The speech markup is rather simplistic: `<div type="sent">` marks sentence boundaries, `<sayas="number">` tells the speech synthesizer that "81" should be pronounced "eighty-one" not "eight one".

The JSML output is passed to the FreeTTS speech synthesizer (Sun Microsystems, 2002) which produces the spoken response, in this case *Number 81 leaves from there at 11:37.*

5 Conclusion

The demonstration framework shows natural language generation techniques closely integrated with recent developments in XML web technology (Xalan XSLT processors, Apache Xindice native XML database, Tomcat servlet engine, Cocoon web publishing framework, and FreeTTS speech synthesizer). The software is all free, open-source, and implemented in Java.

Using Cocoon, it is possible to modify an XSLT transformation stylesheet and see the effect immediately. The framework described here can therefore be used as a development environment for XML-based NLG applications.

By switching on tracing to display the intermediate text plan and text specification, the approach can also be used to teach natural language generation concepts.

The approach is being further developed.

References

Apache Jakarta Project. 2003. Tomcat (latest version). http://jakarta.apache.org/tomcat/.

Apache XML Project. 2003a. Apache Xindice. http://xml.apache.org/xindice/.

Apache XML Project. 2003b. Cocoon (latest version). http://xml.apache.org/cocoon/.

Apache XML Project. 2003c. Xalan-Java (latest version). http://xml.apache.org/xalan-j/.

Tilman Becker and Stephan Busemann, editors. 1999. *May I Speak Freely? Between Templates and Free Choice in Natural Language Generation.* Proceedings of the KI-99 Workshop. DFKI, Saarbrücken.

Steven Bird, Kazuaki Maeda, and Xiaoyi Ma. 2001. AGTK: The annotation graph toolkit. In *IRCS Workshop on Linguistic Databases*, Philadelphia.

Kristiina Jokinen and Graham Wilcock. 2001. 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.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems.* Cambridge University Press.

Sun Microsystems. 1999. Java Speech Markup Language Specification, version 0.6. http: //java.sun.com/products/java-media/ speech/forDevelopers/JSML.

Sun Microsystems. 2002. FreeTTS: A speech synthesizer written entirly in the Java programming language. http://freetts.sourceforge.net/.

Kees van Deemter, Emiel Krahmer, and Mariët Theune. 1999. Plan-based vs. template-based NLG: A false opposition? In Becker and Busemann (1999), pages 1–5.

Graham Wilcock. 2001. Pipelines, templates and transformations: XML for natural language generation. In *Proceedings of the 1st NLP and XML Workshop*, pages 1–8, Tokyo.

Graham Wilcock. 2002. XML-based Natural Language Generation. In *Towards the Semantic Web and Web Services: XML Finland 2002 Slide Presentations*, pages 40–63, Helsinki.