Implementing a Variety of Linguistic Annotations through a Common Web-Service Interface

Adam Funk and Ian Roberts and Wim Peters
University of Sheffield, Department of Computer Science
Regent Court, 211 Portobello, Sheffield, S1 4DP, UK
a.funk,i.roberts,w.peters@dcs.shef.ac.uk

19th February 2010

We present a web service toolkit and common client for a series of natural language processing (NLP) services as a contribution to CLARIN'S European Demonstrator [4].

The CLARIN services are standard SOAP web services. They take their input as binary data (as a MIME attachment to the SOAP message, according to the MTOM specification)—though the services are intended to process text they can handle input in many formats including XML, HTML and PDF, extracting the text from the source data using the format handling mechanism provided by GATE. A service loads the input data into a GATE Document object, then processes that Document using a GATE *DocumentProcessor* (typically a wrapper around a saved GATE application), and returns its output as XML data.

The various components making up the service implementation are configured using the Spring framework, making it simple to slot in alternative *DocumentProcessor* implementations for different services without changes to the code. The aspect-oriented programming tools provided by Spring are used to allow pooling of several identical *DocumentProcessors*, to support multiple concurrent web service clients. The web service layer is provided by the Apache CXF toolkit, which itself uses Spring extensively and thus was a good fit with the Spring-driven architecture adopted for the business logic.

This toolkit will work easily for any GATE application, and with suitable modification (in the Spring beans configuration) can use any class that *implements DocumentProcessor*—in effect, any class that can analyse a single GATE Document and produce XML output (in any valid XML format). Each web service provides a WSDL file available from the server and uses three methods:

- *process* send only the document content as a `byte[]`;
- *processWithURL* also sends the document URL—the GATE Factory will take the filename into consideration when instantiating the Document (to distinguish PDFs properly, for example);
- *processWithParams* sends a parameter list, which allows the client to specify the original URL, encoding, and mime type (this method allows the greatest flexibility).

We also provide a GUI Java client[1], supplied as a ZIP file with the necessary libraries, so the user needs only a Java 5 runtime environment (JRE). This client uses the *processWithParams* method and sends the `file://` URL, and user-selected encoding

[1]The full paper will include a screenshot.

along with the content of the selected local file. The user selects the service from the list of endpoint URLs included with the client, but can also type in a URL if he is aware of a service that has been added since the client software was issued. Of course, developers can also use the services' WSDL files to produce their own clients for users' direct use or embedment in other software.

We have implemented the following services so far, making use of standards which we have worked with in previous projects (particularly LIRICS[2], SEKT[3], and MUSING[4]):

- *annie-alpha* runs the ANNIE[3] named-entity recognition pipeline and returns the fully annotated document in GATE XML format;
- *maf-en* runs GATE's sentence-splitter, tokenizer, POS-tagger and lemmatizer for English, and returns a MAF [1] XML document;
- *chunking-synaf-en* runs GATE's sentence-splitter, tokenizer, POS-tagger, and NP, VP, and PP chunkers for English, constructs a simple tree for each sentence based on containment, and returns a SYNAF [2] XML document;
- *annie-rdf* runs ANNIE, analyses ANNIE's annotations by type and features, generates RDF according to the PROTON[5] [5] ontology, and returns an RDF-XML document.

We plan to deploy services with MAF output for some other European languages (probably a selection from Bulgarian, Dutch, German, and Spanish) in the near future, based on the resources we have available, and are open to suggestions for others, especially if suitable language resources and processing tools are available to be shared with us and suitable for integration with GATE.

Acknowledgements

This research is partially supported by the European Union's Seventh Framework Program project CLARIN (FP7-212230).

References

[1] Language resource management—morpho-syntactic annotation framework (MAF). Standard ISO/DIS 24611, ISO TC37/SC4, December 2008.

[2] Language resource management—syntactic annotation framework (SynAF). Standard ISO/DIS 24615, ISO TC37/SC4, 2010.

[3] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[4] Marc Kemps-Snijders, Nùria Bel, and Peter Wittenburg. Proposal for a CLARIN European demonstrator. Technical report, CLARIN Consortium, September 2009.

[5] I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (BULO) guidance. Deliverable D1.8.1, SEKT Consortium, July 2005. URL `http://proton.semanticweb.org/D1_8_1.pdf`.

[2]`http://lirics.loria.fr/`
[3]`http://www.sekt-project.com/`
[4]`http://www.musing.eu/`
[5]`http://proton.semanticweb.org/`