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

Adam Funk, Ian Roberts, Wim Peters

Department of Computer Science
University of Sheffield
Regent Court, Sheffield, S1 4DP, UK
{a.funk,i.roberts,w.peters}@dcs.shef.ac.uk

Abstract

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. We have also deployed and tested several natural language processing and information extraction services for English and propose to develop further compatible services using resources for other languages.

1. Introduction

An important goal of CLARIN is to make language resources and technology available to humanities researchers and other end users, especially through web services. The European Demonstrator (Kemps-Snijders et al., 2009) prototype system will integrate a number of resources and services available from the participating institutions. We present here a web service toolkit and client, along with the implementation of a series of NLP and IE services, as a contribution to that system.

2. Web service implementation

Our 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. The 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 (any valid XML element is allowed for versatility). (Please refer to the GATE manual (Cunningham et al., 2010) and API documentation[1] for details of the GATE library.)

All the services share a common WSDL interface as their inputs and outputs are the same; only the underlying GATE application needs to vary between services. The standard interface simply specifies the output as any XML in any namespace, and the implementation does not restrict the XML that the underlying application can produce. However if the output types for a specific service are known and there is a suitable W3C XML Schema available then there is the option to use a custom WSDL for that service with the more constraining schema included, which may be beneficial for certain types of client.

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, typically a *SerialAnalyserController* or *ConditionalSerialAnalyserController* (corpus pipeline); furthermore, with suitable modification in the Spring beans configuration, it can use any class that *implements DocumentProcessor*—in effect, any class that can analyse a single GATE Document (or a GATE *Corpus* containing one Document) and produce any valid XML document (the root element of which which we treat as the result and embed in the SOAP response). Each web service provides a WSDL file available from the server and offers 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).

3. Services currently available

We have implemented the following services so far, making use of standards which we have worked with in

[1] `http://gate.ac.uk/documentation.html`

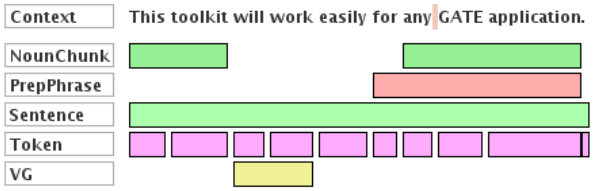

Figure 1: Spans of syntactic elements produced within the chunking service

previous projects (particularly LIRICS[2], SEKT[3], and MUSING[4]).

- The *annie-alpha* service runs the ANNIE (Cunningham et al., 2002) named-entity recognition and orthographic co-reference pipeline and returns the fully annotated document in GATE XML format. The file saved by the client contains ANNIE's output in the default AnnotationSet and the input document's HTML or XML mark-up in the "Original markups" AnnotationSet.

- The *maf-en* service runs GATE's sentence-splitter, tokenizer, POS-tagger and morphological analyser (lemmatizer) for English, and returns an XML document containining the morphosyntactic information according to the MAF (ISO, 2008) standard.

- The *chunking-synaf-en* service runs GATE's sentence-splitter, tokenizer, POS-tagger, and NP and VP chunkers (Cunningham et al., 2010, §17) for English, as well as a simple PP chunker, to produce annotations as shown in Figure 1 (where *VG* means verb group). The application then constructs a simple syntactic tree for each sentence based on simple containment (each phrase or token annotation is a constituent of the smallest sentence or phrase annotation containing it), as shown in Figure 2, and returns an XML document according to the SYNAF (ISO, 2010) standard.

  The syntactic detail is not complete but chunking and constructing a tree this way is reasonably accurate and reliable for many purposes and much faster (especially for large documents over a web service) than full parsing. (The verb chunker's annotations also contain features indicating tense, voice, etc., which will be incorporated into the SynAF output in the improved version of this service.)

- The *annie-rdf* service runs ANNIE, then analyses ANNIE's annotations by type and features and generates RDF representing the recognized entities as instances according to the PROTON[5] (Terziev et al., 2005) ontology, and returns an RDF-XML document.

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

4. Reference client

We also provide a GUI Java client, 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 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. Figure 3 shows this client's main panel used for sending files to the services, and Figure 4 shows the output panel, which allows the user to inspect the output and save it to a local file.
Of course, developers can also use the services' WSDL files to produce their own clients for users' direct use or embedment in other software.

5. Conclusion and future work

The services described here have been proposed as contributions to CLARIN's European Demonstrator. We also 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).

6. References

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. 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)*.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, A. Funk, G. Gorrell, J. Petrak, H. Saggion, D. Damljanovic, and A. Roberts. 2010. *Developing Language Processing Components with GATE Version 5.0 (a User Guide)*. The University of Sheffield.

ISO. 2008. Language resource management—morphosyntactic annotation framework (MAF). Standard ISO/DIS 24611, ISO TC37/SC4, December.

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

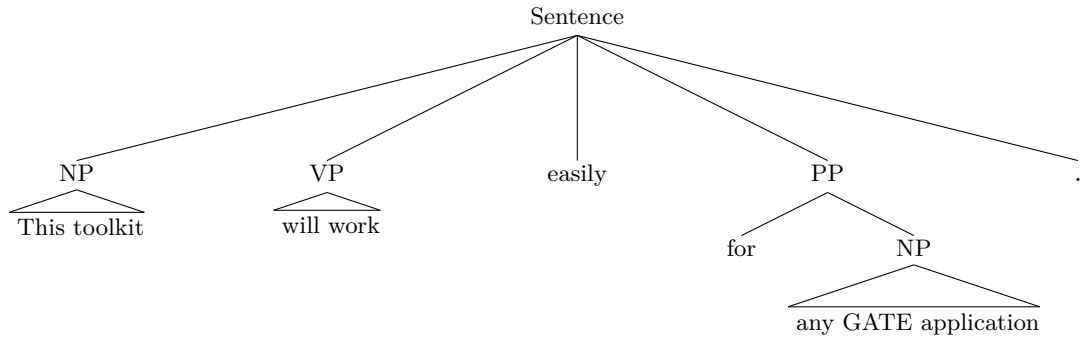

Figure 2: Approximate syntactic tree produced from the annotations in Figure 1

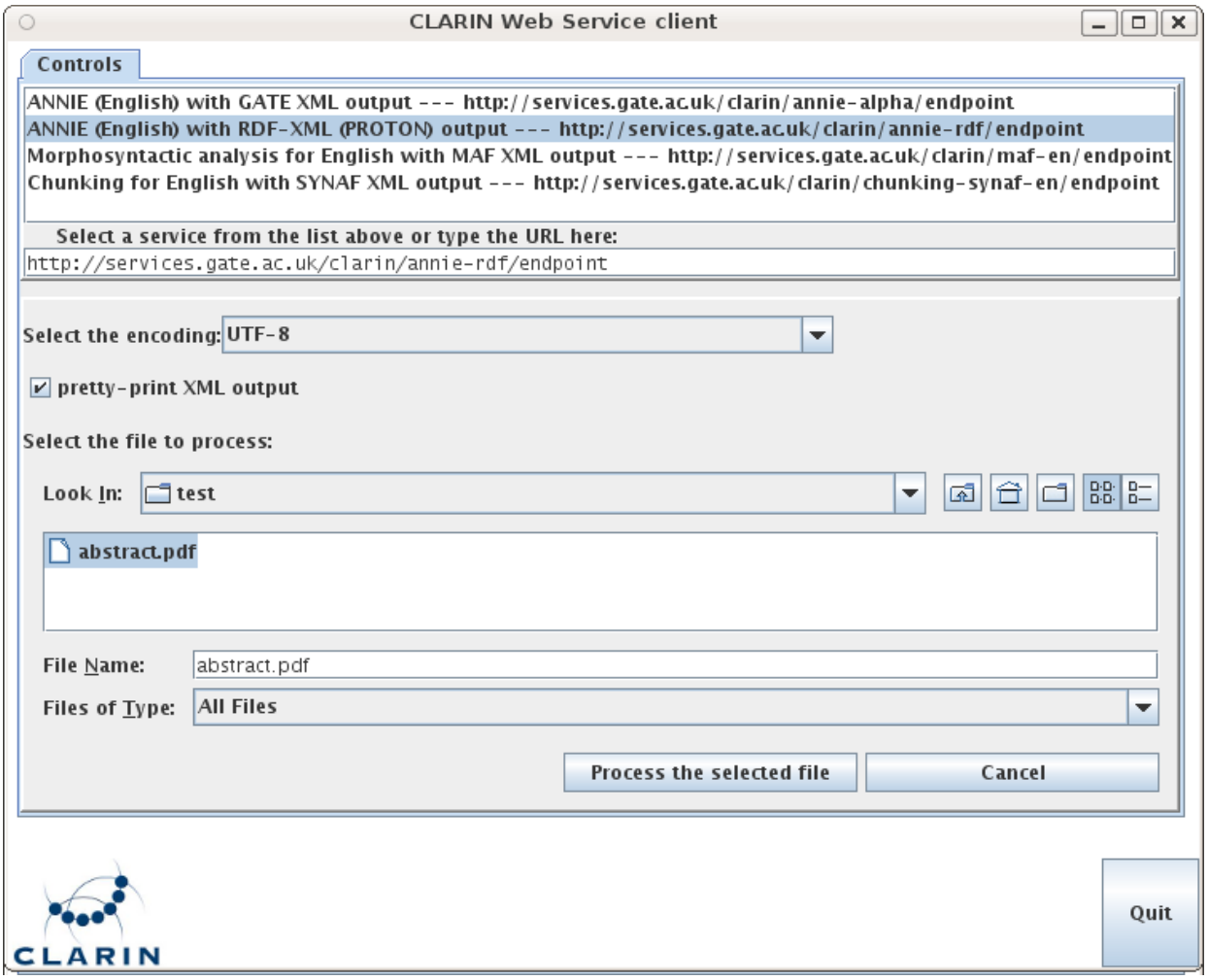

Figure 3: Main panel of the GUI client

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

I. Terziev, A. Kiryakov, and D. Manov. 2005. Base upper-level ontology (BULO) guidance. Deliverable D1.8.1, SEKT Consortium, July.

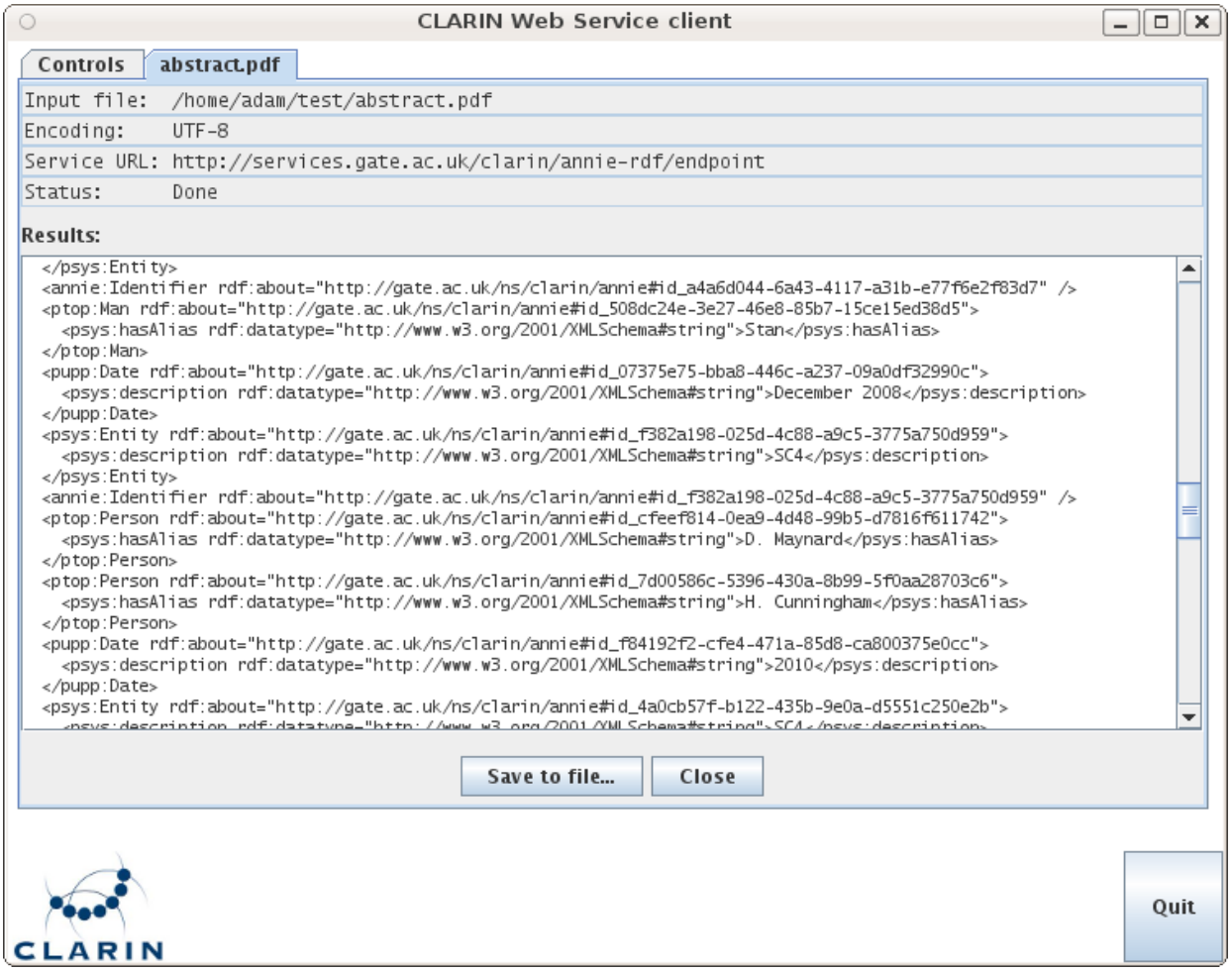

Figure 4: Output panel of the GUI client