

Linguistic Processing Pipelines: Problems and Solutions

Graham Wilcock

University of Helsinki

graham.wilcock@helsinki.fi

1 Introduction

Many of the typical tasks in linguistic processing pipelines can be done with tools from OpenNLP (<http://opennlp.sourceforge.net>).

There are OpenNLP components for sentence detection, tokenization, part-of-speech tagging, phrase chunking, syntactic parsing, named entity recognition, and coreference resolution. These tools have been widely used for several years, and form a good basis for illustrating issues in configuring linguistic processing pipelines. The paper compares several different approaches to creating pipelines based on the tools.

OpenNLP tools use maximum entropy-based statistical language models. Recent versions of OpenNLP provide ready-made models for several languages, including German.

2 Pipelines in Scripts

Applications can use OpenNLP components via the Java API, or pipelines of OpenNLP tools can be created by writing scripts. Sample Linux shell scripts are provided by OpenNLP. The basic mechanism is the Unix pipe: the output stream from one component is piped directly into the input stream of the next component. This data is transient.

The OpenNLP tools are open source Java and platform-independent, but the scripts that invoke them are platform-dependent. Although Windows has a pipe mechanism similar to the Unix pipe, Windows .bat files do not include all the facilities of Unix/Linux shell scripts. There are also many small differences in the script syntax. Converting pipelines from Linux shell scripts to Windows .bat files is therefore error-prone.

3 OpenNLP in the WordFreak GUI

Some people are happy to write scripts, but some prefer graphical user interfaces. WordFreak (<http://wordfreak.sourceforge.net>)

provides an attractive, easy-to-use GUI for both manual and automatic linguistic annotations. It is open source Java and platform-independent.

OpenNLP tools can be used in WordFreak as plugins, and WordFreak is very convenient for manually correcting annotations made by the OpenNLP tools. However, each tool is launched from the GUI menus separately by the user. There is no way to define a pipeline in WordFreak.

4 Pipelines in the GATE GUI

GATE (<http://gate.ac.uk>) has a GUI for linguistic annotations, and also supports pipelines. ANNIE is a ready-to-run GATE pipeline, with a sentence splitter, tokenizer, tagger, and gazetteer lookup for named entity recognition.

The GUI provides facilities for configuring the GATE pipeline by adding and removing components. A wide range of components are provided by GATE, but they do not include the OpenNLP tools. Although it is easy to configure the pipeline with existing GATE components, it is not so easy to add a new component to GATE. Nobody has made the OpenNLP tools available in GATE

5 XSLT Transformations

Both WordFreak and GATE can output linguistic annotations in stand-off XML formats, but they each use their own specific format. This raises the issue of interoperability. How can annotations be interchanged when tools use different formats?

This can be done by XSLT transformations. For example, GATE XML can be transformed to WordFreak XML format. Wilcock (2009) shows sample XSLT stylesheets for transformations between WordFreak, GATE and UIMA.

How can XSLT transformations be used with pipelines? Frequent format changes between processing steps are not good. A better approach is to run a complete pipeline up to a certain step in one

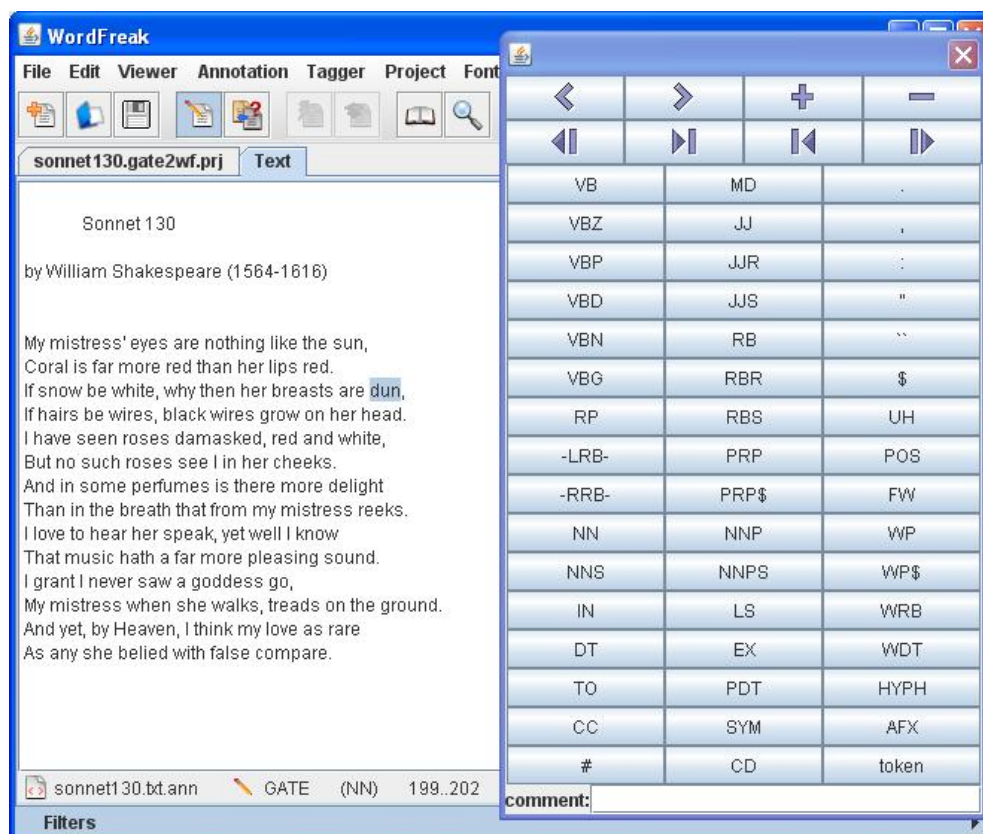


Figure 1: Editing GATE annotations in WordFreak

framework, then transform the output into a different format and input it to another framework.

Figure 1 illustrates this approach. The complete ANNIE pipeline was run in GATE. The annotations in GATE XML format were transformed to WordFreak XML format by an XSLT stylesheet from (Wilcock, 2009). The annotations made by GATE are now manually edited in WordFreak. GATE tagged *dun* incorrectly as NN. The WordFreak POS menu is used to change the tag to JJ.

6 OpenNLP in Ant

Apache Ant (<http://ant.apache.org>) is widely used for organizing workflows of many kinds, so it is natural to use Ant for linguistic processing pipelines. Ant is Java- and XML-based, and employs techniques that avoid many of the platform-dependency problems of scripts.

Pipelines of OpenNLP components are easy to define in Ant. As the OpenNLP tools are Java, each component is an Ant `<java>` task. Useful sequences of tasks can be defined as named Ant targets. For example, the sequence of OpenNLP sentence detector, tokenizer and tagger can be de-

defined as a single "tagging" target. Dependencies between targets are fully supported, for example the "chunking" target can be dependent on the "tagging" target having been done.

Ant provides convenient ways to handle things that often cause trouble in scripts. For example, most of the OpenNLP tools require the same set of jar files to be on the CLASSPATH. In Ant, a `<path>` can be carefully defined once and then referenced repeatedly whenever it is required.

A significant advantage of Ant is its support for XSLT transformations by the Ant `<xslt>` task. This makes it easy to include XML format conversions at any point in an Ant processing pipeline.

7 OpenNLP in UIMA

A more recent approach is Apache UIMA (<http://incubator.apache.org/uima>), which manages to exclude many of the problems, and include many of the advantages, of the different approaches that preceded it. It is open-source Java and platform-independent.

Like WordFreak and GATE, UIMA can be used with a GUI. Instead of having its own GUI it uses

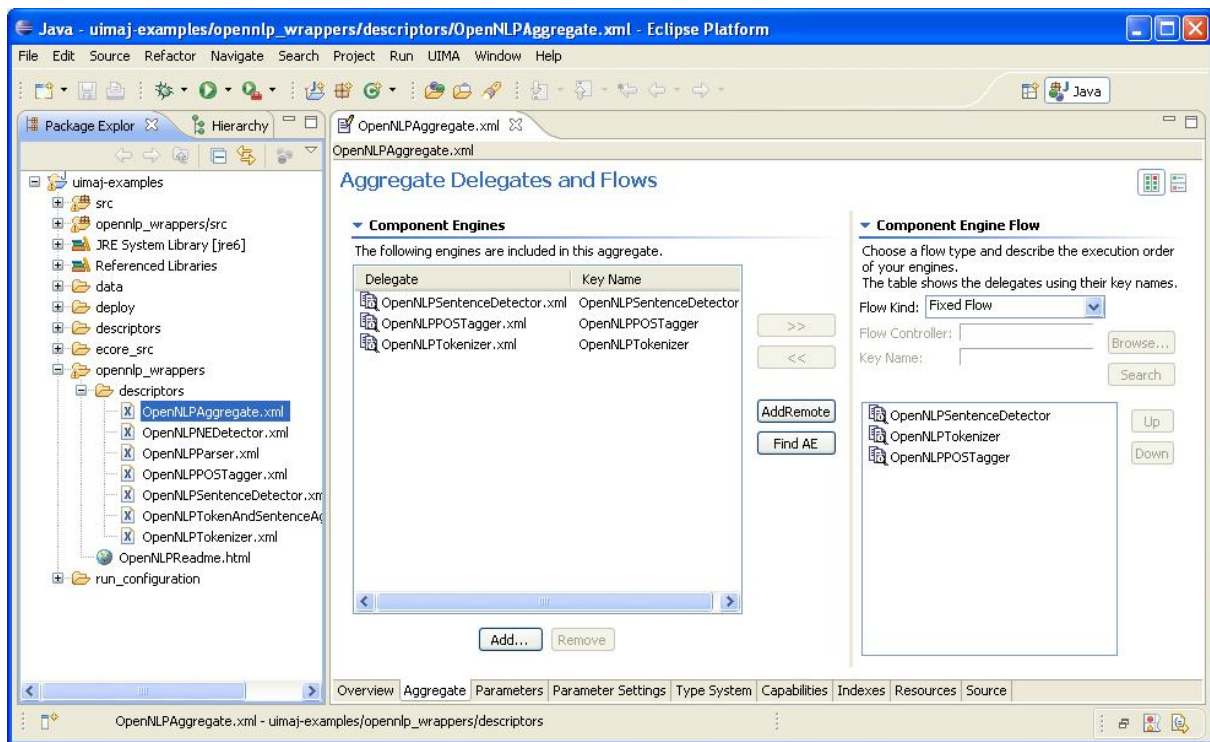


Figure 2: Configuring an OpenNLP annotation pipeline in UIMA

Eclipse, an existing widely-used GUI which many developers already know. UIMA also provides Linux and Windows scripts to run its components, for people who don't want to learn Eclipse.

Like WordFreak and GATE, UIMA can output linguistic annotations in stand-off XML format. Instead of having its own specific format, UIMA supports interoperability by using XML Metadata Interchange (XMI), an OMG standard.

In UIMA, annotators run in analysis engines. New annotators are written in Java, and existing annotation tools such as the OpenNLP tools are converted to UIMA annotators by Java wrappers. Pipelines of annotators run in aggregate analysis engines. Pipelines can be configured by writing XML descriptors (similar in some ways to Ant), or by means of a graphical tool in the GUI.

Figure 2 illustrates this. A pipeline of OpenNLP components is being configured in UIMA using the Component Descriptor Editor. The underlying XML descriptor file that defines the pipeline in detail is automatically generated by this tool.

An advantage of UIMA, that goes beyond the facilities of previous approaches, is its use of a type system that defines annotation types and their features. This supports interoperability of components in a pipeline. Types are used to ensure that

output from one component will be the right type for input to the next component in the pipeline.

8 Conclusion

The paper compared several approaches to managing linguistic processing pipelines, and described problems and possible solutions. Each approach has advantages and disadvantages. Fortunately the later approaches have benefitted greatly from the experience gained with the earlier approaches.

References

- Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *40th Anniversary Meeting of the Association for Computational Linguistics*, Philadelphia.
- Thilo Götz and Oliver Suhre. 2004. Design and implementation of the UIMA Common Analysis System. *IBM Systems Journal*, 43(3):476–489.
- Thomas Morton and Jeremy LaCivita. 2003. Wordfreak: An open tool for linguistic annotation. In *Proceedings of HLT-NAACL 2003, Demonstrations*, pages 17–18, Edmonton.
- Graham Wilcock. 2009. *Introduction to Linguistic Annotation and Text Analytics*. Morgan and Claypool.