Middleware for Creating and Combining Multi-dimensional NLP Markup

Ulrich Schäfer
German Research Center for Artificial Intelligence (DFKI)
Language Technology Lab
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
email: ulrich.schaefer@dfki.de

Abstract

We present the Heart of Gold middleware by demonstrating three XML-based integration scenarios where multidimensional markup produced online by multilingual natural language processing (NLP) components is combined to deliver rich, robust linguistic markup for use in NLP-based applications like information extraction, question answering and semantic web. The scenarios include (1) robust deep-shallow integration, (2) shallow processing cascades, and (3) treebank storage of multi-dimensionally annotated texts.

1 Introduction and Motivation

Heart of Gold is a middleware architecture for creating and combining markup produced by multiple natural language processing components in multilingual environments. It was initially developed for a special sort of multi-dimensional annotation, namely application-oriented, XML- and XSLT-based online integration of various shallow NLP components with a deep HPSG parser for increased robustness in the hybrid natural language processing paradigm (Callmeier et al., 2004).

The middleware, however, can also be used for various other online and offline tasks related to multi-dimensional markup creation and integration. These comprise automatic corpus annotation, incorporation of multi-dimensional markup into a single XML representation, and NLP component cascades interleaved with XSL annotation transformation. The middleware provides XML-RPC interfaces for simple, networking-enabled and programming language-independent application and component integration. Heart of Gold is available as one of the DELPH-IN open source tools available from http://www.delph-in.net\(^1\).

2 Middleware Architecture

Fig. 1 gives a schematic overview of the middleware server in between applications (above) and external NLP components (below). When a new application session in Heart of Gold is started, it takes a configuration specifying NLP components to start for the session. Each component is started according to its own parameterized configuration. The client can send texts to the middleware and the

---

\(^1\)I would like to thank Robert Barbey, Özgür Demir and Thomas Klöcker for their implementation work, Bernd Kiefer and the co-authors in the cited papers for fruitful cooperation, and the reviewers for valuable comments. This work has been supported by a grant from the German Federal Ministry of Education and Research (FKZ 01IW02).
component dependency structure in general forms a graph.

2.1 Session and multi-dimensional annotation management

The resulting multi-dimensional annotations are stored in a per-session markup storage (Fig. 2) that groups all annotations for an input query (a sentence or text) in annotation collections. The markup storage can also be made persistent by saving it to XML files or to an XML database. Annotations can be accessed uniquely via a URI of the form hog://sid/acid/aid in XPath expressions where sid is a session ID, acid is an annotation collection ID and aid is an annotation identifier typically signifying the name of the producing component. Structured metadata like configuration and processing parameters (e.g. processing time and date, language ID etc.) are always stored within the annotation markup as first root daughter element.

2.2 XML standoff markup as first-class citizen

Unlike other NLP architectures (e.g. GATE (Cunningham et al., 2002) etc.), Heart of Gold treats XML standoff annotations (Thompson and McKelvie, 1997) as first class citizens and natively supports XML (and only XML) markup of any kind. Moreover, Heart of Gold does not prescribe specific DTDs or Schemata for annotations, provided that the markup is well-formed. In this sense, it is a completely open framework that may however be constrained by requirements of the actually configured components. The advantage of this openness is easy integration of new components. Mappings need only be defined for the immediately depending annotations (see next section) which is by far not an n-to-n mapping in practical applications.

However, the fact that a specific DTD or Schema is not imposed by the middleware does not mean that there are no minimal requirements. Linking between different standoff annotations is only possible on the basis of a least common entity, which we propose to be the character spans in the original text\(^2\). Moreover, we additionally propose the use of the XML ID/IDREF mechanism to facilitate efficient integration and combination of multi-dimensional markup.

Finally, depending on the scenario, specific common, standardized markup formats are appropriate, an example is RMRS (Copestake, 2003) for deep-shallow integration in Section 3 or the XML-encoded typed feature structure markup generated by SProUT (Drożdżyński et al., 2004).

2.3 XSLT as 'glue' and query language

We propose and Heart of Gold heavily relies on the use of XSLT for combining and integrating multi-dimensional XML markup. The general idea has already been presented in (Schäfer, 2003), but the developments and experiences since then have encouraged us to proceed in that direction and Heart of Gold can be considered as a successful, more elaborated proof of concept. The idea is related to the open markup format framework presented above: XSLT can be used to transform XML to other XML formats, or to combine and query annotations. In particular, XSLT stylesheets may resolve conflicts resulting from multi-dimensional markup, choose among alternative readings, follow standoff links, or decide which markup source to give higher preference.

(Carletta et al., 2003), e.g. propose the NXT Search query language that extends XPath by adding query variables, regular expressions, quantification and special support for querying temporal and structural relations. Their main argument against standard XPath is that it is impossible to constrain both structural and temporal relations within a single XPath query. Our argument is that XSLT can complement XPath where XPath alone is not powerful enough, yet providing a standardized language. Further advantages we see in the XSLT approach are portability and efficiency (in contrast to ‘proprietary’ and slow XPath extensions like NXT), while it has a quite simple syntax in its (currently employed) 1.0 version. XSLT can be conceived as a declarative specification language as long as an XML tree structure

\(^2\)Our experience is that a common tokenization is not realistic—too many existing NLP components have differing concepts of what constitutes a token.
Figure 3: Heart of Gold analysis results in GUI with specialized XML visualizations

is preserved (not necessarily fully isomorphic to the input structure). However, XSLT is Turing-capable and therefore suitable to solve in principle any markup integration or query problem. Finally, extensions like the upcoming XSLT/XPath 2.0 version or efficiency gains through XSLTC (translet compilation) can be taken on-the-fly and for free without giving up compatibility. Technically, the built-in Heart of Gold XSLT processor could easily replace or complement by an XQuery processor. However, for the combination and transformation of NLP markup, we see no advantage of XQuery over XSLT.

Heart of Gold comes with a built-in XSL transformation service, and module adapters (Section 2.4) can easily implement transformation support by including a few lines of code. Stylesheets can also be generated automatically in Heart of Gold, provided a systematic description of the transformation input format is available. An example is mapping from named entity grammar output type definitions in scenario 1 below. Stylesheets are also employed to visualize the linguistic markup, e.g. by transforming RMRS to HTML (Fig. 3) or \LaTeX.

2.4 Integrated NLP components

NLP components are integrated through adapters called modules (either Java-based, subprocesses or via XML-RPC) that are also responsible for generating XML standoff output if this is not supported natively by the components (e.g., TnT, Chunkie). Various shallow and deep NLP components have already been integrated, cf. Fig. 4.

Figure 4: Integrated components. References for components and resources not cited are available on http://heartofgold.dfki.de/Publications_Components.html

3 Scenario 1: Deep-Shallow Integration

The idea of hybrid deep-shallow integration is to provide robust linguistic analyses through multi-dimensional NLP markup created by shallow and deep components, e.g. those listed in Fig. 4. Robustness is achieved in two ways: (1) various shallow components perform preprocessing and partial statistical disambiguation (e.g. PoS tagging of unknown words, named entity recognition) that can be used by a deep parser by means of a so-called XML input chart (multi-dimensional markup combined through XSLT in a single XML
document in a format convenient for the parser). (2) shallow component’s output is transformed through XSLT to partial semantic representations in RMRS syntax (Copestake, 2003) that is potentially more fine-grained and structured than what is digestible by the deep parser as preprocessing input (mainly PoS/NE type and span information via the XML input chart). This allows for (a) a fallback to the shallow representation in case deep parsing fails (e.g., due to ungrammatical input), (b) combination with the RMRS generated by deep parsing or fragments of it in case deep parsing fails.

First application scenarios have been investigated successfully in the DEEPTHought project (Uszkoreit et al., 2004). A further application (hybrid question analysis) is presented in (Frank et al., 2006). Recently, linking to ontology instances and concepts has been added (Schäfer, 2006).

4 Scenario 2: Shallow Cascades

The second scenario is described in (Frank et al., 2004) in detail. A robust, partial semantics representation is generated from a shallow chunker’s output and morphological analysis (English and German) by means of a processing cascade consisting of four SProUT grammar instances with four interleaved XSLT transformations. The cascade is defined using the declarative system description language SDL (Krieger, 2003). An SDL architecture description is compiled into a Java class which is integrated in Heart of Gold as a sub-architecture module (Fig. 5). The scenario is equally a good example for XSLT-based annotation integration. Chunker analysis results are included in the RMRS to be built through an XSLT stylesheet using the XPath expression

\[
\text{document($\text{uri}/\text{chunkie/chunks/chunk[}
\text{@cstart=$\text{begin}span and @cend=$\text{end}span]\]
\]

where $\text{uri}$ is a variable containing an annotation identifier of the form hog://sid/acid/aid as explained in Section 2.1.

5 Scenario 3: Corpus Annotation

Given the powerful online middleware architecture described above, automatic, multi-dimensional corpus annotation can then be regarded as a simple by-product. Heart of Gold supports persistent storage of XML markup either on the file system or to XML databases through the built-in XML:DB interface. Through XSLT, it is possible to combine multi-dimensional markup (that would straightforwardly be stored in multiple XML documents) into a single XML document.

Figure 5: SProUT XSLT cascade in a Heart of Gold architecture instance.

References


