Flexible Interfaces in the Application of Language Technology to an eScience Corpus

C.J. Rupp, Ann Copestake, Simone Teufel, Benjamin Waldron

Computer Laboratory, University of Cambridge

Abstract

We describe two key interfaces used in an architecture for applying a range of Language Technology tools to a corpus of Chemistry research papers, in order to provide a basis of robust linguistic analyses for Information Extraction tasks. This architecture is employed in the context of the eScience project ‘Extracting the Science from Scientific Publications’ (a.k.a. SciBorg), as described in Copestake et al. (2006). The interfaces in question are the common representation for the papers, delivered in a range of formats, and the coding of various types of lingustic information as standoff annotation. While both of these interfaces are coded in XML their structure and usage are quite distinct. However, they are employed at the main convergence points in the system architecture. What they share is the ability to represent information from diverse origins in a uniform manner. We emphasise this degree of flexibility in our description of the interface structures and the design decisions that led to these definitions.

1 Introduction

The purpose of this paper is to document two interface structures that play a crucial role in the architecture of the eScience project ‘Extracting the Science from Scientific Publications’ (also known as SciBorg). The project’s aims and the architecture itself are described in more detail in Copestake et al. (2006).

As the official project title suggests, SciBorg is concerned with Information Extraction (IE) from published scientific research, in this case Chemistry. However, the real challenge of the project lies in the comprehensive application of current Language Technology to an extensive corpus of research papers to provide robust linguistic analsyses, on which various Information Extraction tasks can be based. As we are, effectively, mining the text of Chemistry research, we feel it is appropriate to refer to the corpus of research papers as an *eScience corpus.*

In this context, the most significant interfaces are: the common representation for the papers as an input to the analysis tools and the pool of resulting analyses. We emphasise the flexibility required in both of these interfaces, but for different reasons. While the research papers may be delivered in a range of formats, an adequate common representation for both content and formatting information economises on the number of interfaces to be supported, as long as the cost of maintaining the converion processes can be kept to a minmum. In applying existing analysis tools, at different levels of analysis, we attempt to maximise the synergies between the available components, although the components themselves were not, necessarily, developed with this in mind. This means that interface structure that brings the analyses together must accommodate those interactions.

We adopt a common XML markup for the research papers and a standoff annotation formalism for the coding of various types of linguistic information. While both of these interfaces are coded in XML their structure and usage are quite distinct. What they share is the ability to represent information from diverse origins in a uniform manner.

2 The SciBorg Corpus

The SciBorg project is concerned with Information Extraction from published Chemistry research papers. The three publishers affiliated to the project: The Royal Society of Chemistry (RSC), Nature Publishing Group (NPG) and International Union of Crystallography (IUCr), have provided a corpus of recently published papers. Delivery is either in the form of XML, or (in the case of IUCr) in SGML that is easily converted to XML. However, the original XML encoding is specific to the publishers, following a DTD defined for their own needs. Obviously, there are far fewer interfaces to maintain, if we convert all the papers in the corpus to a common XML encoding, so that subsequent processing modules only have to interface with one XML encoding. For this purpose we have adopted an XML schema that has been developed over the course of several projects for the precise purpose of representing the logical structure of scientific research papers. Nevertheless, some adaptations to the schema were require for the specific needs of a corpus collected in well-defined XML encodings of publishers' markup. We have named the result SciXML, intuitively XML for Science.

3 The Development of SciXML

SciXML originates in XML markup for the logical structure of scientific papers in Computational Linguistics (Teufel and Moens, 1997; Teufel, 1999). It has subsequently been employed to corpora from a variety of disciplines, including Cardiology (Teufel and Elhadad, 2002) and Genetics (Hollingsworth et al., 2005).

What is equally significant is that these corpora, while consistent in the function of their texts, were collected from a variety of different sources and in varying formats, so that conversions to SciXML have been defined from: LaTeX, HTML and PDF (via OCR). The conversion from low level formatting produced a cumulative effect on the immediate precursor for our SciXML schema. The more functional levels of the markup were impoverished, as only distinctions that affected the formatting could be retrieved. The handling of lists was rather simplified and tables excluded, because of the difficulty of processing local formatting conventions. Equally, the applications had no necessity to represent papers in full formatting, so information like the modification of font faces at the text level was excluded. While footnotes were preserved these were collected at the end of the paper, effectively as end notes, alongside the vestigial representations of tables and figures, chiefly their captions.

SciBorg has the advantage of access to publishers' markup which supports functional or semantic distinctions in structure and provides a detailed coding of the content and structure of lists and tables, rather than just their formatting on the page. However, we envisage IE applications which involve some rendering of the paper contents in a readable form, e.g. in authoring and proof reading aids for Chemists. While not as detailed as the publishers page formatting we would require the paper content to be recognisable, exploiting HTML as a cheap solution to any more complex display problems. This implies retaining more of the explicit formatting information, particularly at the text level. In practice, this has meant the addition of inline markup for font face selections and some inclusion of LaTeX objects for formulae, as well as the preservation of the origin points for floats, as they are known to LaTeX (table and figures). As a result SciXML retains a focus on the logical structure of the paper, but preserves as much formatting information as is required for effectively rendering the papers in an application.

4 The SciXML Schema

The resulting form of SciXML is defined as a Relax NG schema. Since this is an XML-based formalism, it is difficult to exhibit any substantive fragment in the space available here. Figure 1 shows just the overall strucuture of a paper and the first level of elements in the <REFERENCELIST> element. The most comprehensive description that is appropriate here is a catalogue of the types of construct in SciXML.

Paper Identifiers: We require enough information to identify papers both in processing and when tracing back references. While publisher's markup may contain an extensive log of publication and reviewing, a handful of elements suffice for our needs: <TITLE>, <AUTHOR>, <AUTHORS>, <FILENO>, <APPEARED>

Sections: The hierarchical embedding of text segments is encoded by the <DIV> element, which is recursive. A DEPTH attribute indicates the depth of embedding of a division. Each division starts with a <HEADER> element.

Paragraphs are marked as element <P>. The paragraph structure has no embedding in SciXML, but paragraph breaks within <ABSTRACT>, <CAPTION> and <FOOTNOTE> elements and within list items (<LI>) are preserved noted with a <SUBPAR> element.

Abstract, Examples and Equations are text sections with specific functions and formatting and can be distinguished in both publishers' markup and in the process of recovering information from PDF. We have added a functionally determined section <THEOREM>, as we have encountered this type of construct in some of the more formal papers. Similarly, linguistic examples were distinguished early on, as Computational Linguistics was the first corpus to be treated.

Tables, figures and footnotes are collected in a list element at the end of the document, <TABLELIST>, <FIGURELIST>, <FOOTNOTELIST>, respectively. The textual position of floats is marked by an <XREF/> element. The reference point of a footnote is marked by a separate series of markers: <SUP>, also used for similar functions in associating authors and affiliations.

```
<define name="PAPER.ELEMENT">
  <element name="PAPER">
    <ref name="METADATA.ELEMENT" />
    <optional>
      <ref name="PAGE.ELEMENT" />
    </optional>
    <ref name="TITLE.ELEMENT" />
    <optional>
      <ref name="AUTHORLIST.ELEMENT" />
    </optional>
    <optional>
      <ref name="ABSTRACT.ELEMENT" />
    </optional>
    <element name="BODY">
      <zeroOrMore>
        <ref name="DIV.ELEMENT" />
      </zeroOrMore>
    </element>
    <optional>
      <element name="ACKNOWLEDGMENTS">
        <zeroOrMore>
          <choice>
            <ref name="REF.ELEMENT" />
            <ref name="INLINE.ELEMENT" />
          </choice>
        </zeroOrMore>
      </element>
    </optional>
    <optional>
      <ref name="REFERENCELIST.ELEMENT">
    </optional>
    <optional>
      <ref name="AUTHORNOTELIST.ELEMENT">
    </optional>
    <optional>
      <ref name="FOOTNOTELIST.ELEMENT">
    </optional>
    <optional>
      <ref name="FIGURELIST.ELEMENT">
    </optional>
    <optional>
      <ref name="TABLELIST.ELEMENT">
    </optional>
  </element>
</define>

<define name="REFERENCELIST.ELEMENT">
  <element name="REFERENCELIST">
    <zeroOrMore>
      <ref name="REFERENCE.ELEMENT" />
    </zeroOrMore>
  </element>
</define>
```

Figure 1: A fragment of the Relax NG schema for SciXML

Lists: various types of lists are supported with bullet points or enumeration, according to the TYPE attribute of the <LIST> element. Its contents will be uniformly marked up as <LI> for list items.

Cross referencing takes a number of forms including the <SUP> and <XREF> mentioned above. All research papers make use textual cross references to identify sections and figures. For Chemistry, reference to specific compounds was adopted, from the publishers' markup conventions. The other crucial form of cross reference in research text is citations, linking to bibliographic information in the <REFERENCELIST>.

Bibliography list: The bibliography list at the end is marked as <REFERENCELIST>. It consists of <REFERENCE> items, each referring to a formal citation. Within these reference items, names of authors are marked as <SURNAME> elements, and years as <YEAR>.

5 The Conversion Process

The conversion from the publisher's XML markup to SciXML can be carried out using an XSLT stylesheet. In fact, most of the templates are quite simple, mainly reorganising and/or renaming content. The few exceptions are collections of elements such as footnotes and floats to list elements and replacing the occurrence *in situ* with a reference marker. Elements that explicitly encode the embedding of text divisions are systematically mapped to a non-hierarchical division with a DEPTH attribute recording the level of embedding. Figure 2 shows a template for converting a section (<sec>) element from a publisher's markup to a <DIV> in SciXML. The SciXML conventions show an affinity for the structure of the formatted text which follows directly from usage with text recovered from PDF. For our current purposes this is considered harmless, as no information is lost. We assume that an application will render the content of a paper, as required, e.g. in HTML. In fact, we already have demonstration applications that systematically render SciXML in HTML via an additional stylesheet. The one SciXML convention that is somewhat problematic is the flattening of paragraph structure. While divisions can embed divisions, paragraphs may not embed paragraphs, not even indirectly. To avoid information loss, here an empty paragraph break marker is added to list elements, abstracts and footnotes which may, in the general case, include paragraph divisions. Although earlier versions of SciXML also encoded sentence boundaries, the sentence level of annotation has been transferred to the domain of linguistic standoff annotation, where it can be generated by automatic sentence splitters.

```
<xsl:template match="sec">
<DIV DEPTH="{@level}">
<xsl:apply-templates/>
</DIV>
</xsl:template>
```

Figure 2: An XSLT conversion template

While the XSLT stylesheets that convert the publisher XML to SciXML are relatively straightforward, each publisher DTD or schema requires a separate script. This places a manual overhead on the inclusion of papers from a new publisher, but fortunately at a per publisher rather than per journal level. In practice, we have found that the element definitions are still sufficiently similar to allow a fair amount of cut-and-paste programming, so that the overhead decreases as more existing conversion templates exist to draw on. A recent extension of SciXML conversion to the PLoS DTD presented remarkably few unknown templates.

6 Language Technology in SciBorg

The goals of the SciBorg project, as its full title suggests, concern Information Extraction, in fact a range of IE applications are planned all starting from a corpus of published chemistry research. However, the common path to those goals and the real challenge of the project is the application of Language Technology, and, in particular, linguistically motivated analysis techniques. In practice, this involves the use of so-called 'deep' parser, based on detailed formal descriptions of the language with extensive grammars and lexicons, and shallower alternatives, typically employing stochastic models and the results of machine learning. This multi-engine approach has been employed in Verbmobil (Ruland et al., 1998; Rupp et al., 2000) and Deep Thought (Callmeier et al., 2004).

While there have been considerable advances in the efficiency of deep parsers in recent years, there are a variety of ways that performance can be enhanced by making use of shallower results, e.g. as preprocessors or as a means of selecting the most interesting sections of a text for deeper analysis. In fact, the variety of different paths through the analysis architecture is a major constraint on the design of our formalism for linguistic annotations, and the one which precludes the use of the major existing frameworks for employing Language Technology in IE, such as GATE (`http://gate.ac.uk`).

7 Multiple Analysis Components

The main deep and shallow parsing components that we use have been developed over a period of time and represent both the state of the art and the result of considerable collaboration.

PET/ERG: PET (Callmeier, 2002) is a highly optimise HPSG parser that makes use of the English Resource Grammar (ERG) (Copestake and Flickinger, 2000). The ERG provides a detailed grammar and lexicon for a range of text types. The coverage of the PET/ERG analysis engine can be extended by an unknown word mechanism, provided that a partial identification of the word class is possible, e.g. by POS (part of speech) tagging.

RASP provides a statistically trained parser that does not require a full lexicon (Briscoe and Carroll, 2002). The parser forms part of a sequence of analysers including a sentence splitter, tokeniser and a POS tagger.

A key factor in combining results from multiple parsers is that they present compatible results. Both of the parsers we are using are capable of producing analyses and partial analyses in the same form of representation: Robust Minimal Recursion Semantics (Copestake, 2003), henceforth RMRS. This is a form of underspecified semantics resulting from the tradition of underspecification in symbolic Machine Translation. In RMRS, the degree of underspecification is extended so that all stages of deep and shallow parsing can be represented in a uniform manner. It is therefore feasible to combine the results of the deep and shallow parsing processes.

Our presentation of the parsers above should have suggested one other path through the system architecture, in that the RASP tagger can provide the information necessary to run the unknown word mechanism in the PET parser, what this requires is a mapping from the part of speech tag to an abstract lexical type in the ERG grammar definition.

The combination of results from deep and shallow parsers is only part of the story, as these are general purpose parsers. We will also need components specialised for research text and, in particular, for Chemistry. The specialised nature of the text of Chemistry research is immediately obvious, both in specialised terms and in sections which are no longer linguistic text, as we know it. A sophisticated set of tools for the analysis of Chemistry research is being developed within the SciBorg project, on the basis of those described in Townsend et al. (2005). These range from NER (Named Entity Recognition) for Chemical terms, through the recognition of data sections, to specialised Chemistry markup with links to external and automatically generated information sources. For the general parsing task the recognition of specialised terms and markup of data sections are the most immediate contribution. Though this does, to some extent, exacerbate the problems of ambiguity, as now deep, shallow and Chemistry NER processes have different results for the same text segment.

The overlap between ordinary English and Chemical terms can be trivially demonstrated, in as much as the prepositions *in* and *as* in sentence initial position can be easily confused with *In* and *As*, the chemical symbols for indium and arsenic, respectively. Fortunately some English words that moonlight as chemical symbols are less common in research text, such as *I*, but this is only an example of the kinds of additional ambiguity. Formally, the word *lead* would have at least 3 analyses, as verb, noun and element, but the latter two are not orthogonal.

8 Standoff Annotation

Standoff annotation is an increasingly common solution for pooling different types of information about a text. Essentially, this entails a separation of the original text and the annotations into two separate files. There are some practical advantages in this: you are less likely to obscure the original text under a mesh of annotations and less likely to proliferate numerous partially annotated versions of the object text. However, the true motivation for standoff annotation is whether different annotation schemes will impose different structures on the text. To some extent XML forces a standoff annotation scheme, by enforcing a strict tree structure, so that even one linguistic annotation of a formatted text above the word level risks becoming incompatible with the XML DOM tree. As a simple example, we exhibit here a fragment of formatted text from a data section of a Chemistry paper, alongside its SciXML markup and a simple linguistic markup for phrase boundaries.

Formatted text *calculated for* C_{11} H_{18} O_3

SciXML markup <**it**> calculated for </**it**>
C<sb>11</sb>H<sb>18</sb>O<sb>3</sb>

Phrasal markup <v>calculated</v>
<**pp**>for <ne>C11H1803</ne></**pp**>

Here, the assignment of a simple phrase structure conflicts with the formatting directives for font face selection. The XML elements marked in bold face cannot be combined in the same XML dominance tree.

With each additional type or level of annotation the risk of such clashes increases, so a clean separation between the text and its annotations is helpful, but where do you separate the markup and how do you maintain the link between the annotations and the text they address. As we have a common XML markup schema, including logical structure and some formatting information we have a clear choice as to our text basis. The link between text and annotations is usually maintained by indexing. Here, there are some options available.

8.1 Indexing

The indexing of annotations means that each element of the standoff file encodes references to the position in the text file where the text it relates to occurs, typically this encodes a span of the text. Some form of indexing is a prerequisite for standoff annotation. The simplest indexing is by byte offset, encoding the position of a text segment in terms of the number of bytes in the file preceding its start and finish positions. This is universal, in that it can cope with all types of file content, but is not stable under variations in character encoding. For an XML source text, indexing by character offset is more useful, particularly if character entities have been resolved. Extraneous binary formats such as graphics will be encoded externally as <NDATA> elements. In fact, the variant of character offset we adopt involves numbering the Unicode character points, as shown in Figure 3.

Raw Text: ”<p>Come <i>here</i>!</p>”

Unicode character points:

```
.<.p.>.C.o.m.e . Δ . < . i . > . h .e . r . e .
0  1 2  3 4 5 6 7    8    9   10  11  12 13 14  15
                < . / . i . > . ! . < . / . p . > .
                  16  17  18  19  20  21 22 23  24
```

Figure 3: character pointers (points shown as ‘.’)

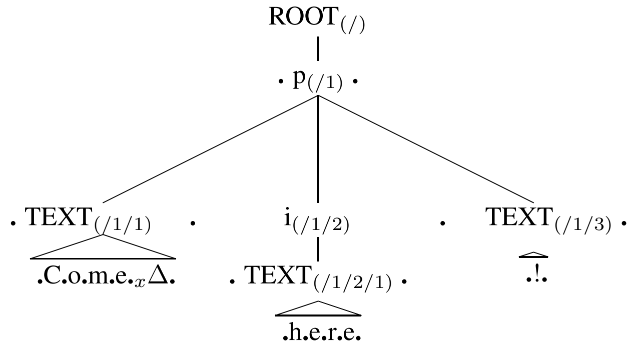
A simple annotation of linguistic tokens would then produce three elements:

```
<w from=’3’ to=’7’>Come</w>
<w from=’11’ to=’15’>here</w>
<w from=’19’ to=’20’>!</w>
```

XML also offers an additional structure for navigating around the file contents.We can take the XML DOM trees as a primary way of locating points in the text and character positions as secondary, so that a character position is relative to a branch in the XML tree. This XPoint notation is demonstrated in Figure 4. The XML tree positions will not be affected by changes within an XML element, e.g. additional attributes, but will not be stable against changes in the DOM structure itself.

We currently make use both character offset indexing and XPoint indexing in different modules. This requires conversion via a table which relates XPoints to character positions for each file. The initial choice of indexing mode for each subprocess was influenced by the availability of different styles of XML parser[1], but also informed by the ease of converting character information from non-XML linguistic tools. In the long term our aim would be to eliminate the use character offset indexing.

[1] Computing character offsets is easier with an event-based parser that has access to the input parsed for each event. XPoints can be tracked more easily in a DOM parser.



xpoint at x is: "`/1/1.4`"

Figure 4: xpoint-based pointers (points shown as '.')

9 Types of Annotation

We have motivated standoff annotation on the basis of the incompatibility between the tree structure of an XML document and the annotation of arbitrary segments of that document. The collection of annotations can be encoded in an XML format with its own DTD, because they form, together, the representation of a grap structure, as a set of edges. The nodes of this graph are, ultimately, the character positions in the original XML document, whichever way you index them. This is similar to a chart or well-formed substring table in a parser, except that muliple tokenisations can lead to varying word boundaries. We therefore have more potential nodes in the annotation graph than a (single) parser's chart. In contrast, a word lattice, as used in speech recognition, may have still more potential nodes. In a word lattice, the succession of nodes is not only determined by a physical convergence, but also by factors like statistical language models. These additional constraints mean that a word lattice may have distinct nodes that represent the same time frame. While this condition will not occur in our annotation graphs, we have adopted a standard for the DTD of our graphs which is general enough to support the treatment of word lattice inputs in speech processing. The reason for this is to link up with existing standards in the Language Technology community, and, in particular, in the HPSG processing comunity.

The origin of our annotation standard lies with the (ISO working draft) MAF standard (Clement and de la Clergerie, 2005) for morphological annotations. A variant of this had been developed within the DELPH-IN community as an emergent standard for input to parsers. This is known as SMAF (Waldron et al., 2006). Our annotation graphs collect all the annotations for a whole document in one file. This means that we require require a further generalisation on the SMAF standard. We have termed this SAF (Waldron and Copestake, 2006). We also encode analysis results, as well as input sentences, tokens and tagging, in the same annotation file. Although these various annotations have the same general form and the content of an annotation is essentially based on RMRS, the details for each type of annotation vary slightly.

9.1 Sentences

The results of a sentence splitter are represented by an annotation edge with the `type` sentence, unique identifier, initial and final index positions, initial and final lattice nodes and the text content of the sentence, as a `value` attribute string.

```
<annot type='sentence' id='s133'
from='42988' to='43065'
source='v4987' target='v5154'
value='calculated for C11H18O3'/>
```

9.2 Tokens

A tokeniser determines tokens at the word level, including punctuation and common abbreviations. These are represented by annotations of the `type` token, with a span in terms of offset positions and lattice nodes, as well as a dependency and the token string as a `value`.

```
<annot type='token' id='t5153'
from='43035' to='43065'
source='v5152' target='v5153'
deps='s133' value='C11H18O3'/>
```

We record a dependency between the tokenisation and sentence splitting because of the sequence of processing in the SciBorg architecture.

9.3 POS Tags

Part of speech tags are annotated as being dependent on a tokenisation, this is a fixed relation between the two processing steps and it allows us some economy in representing the annotation.

```
<annot type='pos' id='p5153' deps='t5153'
source='v5152' target='v5153' value='NP1'/>
```

```
<annot type=’rmrs’ id=’r2’ from=’42988’ to=’43065’ source=’v5150’ target=’v5153’>
<rmrs cfrom=’42988’ cto=’43043’>
<label vid=’1’/>
<ep cfrom=’42988’ cto=’43030’>
  <realpred lemma=’calculate’ pos=’v’ sense=’1’/><label vid=’10’/><var sort=’e’ vid=’2’/></ep>
<ep cfrom=’43031’ cto=’43034’>
  <realpred lemma=’for’ pos=’p’/><label vid=’10001’/><var sort=’e’ vid=’13’/></ep>
<ep cfrom=’43035’ cto=’43065’>
  <gpred>proper_q_rel</gpred><label vid=’14’/><var sort=’x’ vid=’12’/></ep>
<ep cfrom=’-1’ cto=’-1’>
  <gpred>named_rel</gpred><label vid=’17’/><var sort=’x’ vid=’12’/></ep>
<rarg><rargname>ARG2</rargname><label vid=’10’/><var sort=’x’ vid=’3’/></rarg>
<rarg><rargname>ARG1</rargname><label vid=’10001’/><var sort=’e’ vid=’2’/></rarg>
<rarg><rargname>ARG2</rargname><label vid=’10001’/><var sort=’x’ vid=’12’/></rarg>
<rarg><rargname>RSTR</rargname><label vid=’14’/><var sort=’h’ vid=’15’/></rarg>
<rarg><rargname>BODY</rargname><label vid=’14’/><var sort=’h’ vid=’16’/></rarg>
<rarg><rargname>CARG</rargname><label vid=’17’/><constant>*TOP*</constant></rarg>
<ing><ing-a><var sort=’h’ vid=’10’/></ing-a><ing-b><var sort=’h’ vid=’10001’/></ing-b></ing>
<hcons hreln=’qeq’><hi><var sort=’h’ vid=’15’/></hi><lo><label vid=’17’/></lo></hcons>
</rmrs>
</annot>
```

Figure 5: An RMRS annotation

Using the RASP tagger, the tagset is based on CLAWS. The POS tagger also provides the option of an RMRS output.

9.4 Chemical Terms

We use the NER functionality in OSCAR-3 (Corbett and Murray-Rust, 2006) to provide an annotation for Chemical terms.

```
<annot type="oscar" id="o554"
from="/1/5/6/27/51/2/83.1"
to="/1/5/6/27/51/2/88/1.1" >
  <slot name="type">compound</slot>
  <slot name="surface">C11H18O3</slot>
  <slot name="provenance">formulaRegex</slot>
</annot>
```

This example differs from the annotations shown above in two obvious respects: the indexing is given in XPoint rather than character offsets and the XML element has content rather than a `value` attribute. This representation provides information about the way that the named entity recognition was arrived at. For use in further stages of analysis, this content should be made compatible with an RMRS representation.

9.5 RMRS Annotations

For the sake of completeness we include an example of an RMRS annotation in Figure 5. This represents the result for a partial analysis. The content of this element is represented in the XML form of RMRS annotation. However, the mechanisms which make RMRS highly suitable for representing arbitrary levels of semantic underspecification in a systematic and extensible way, e.g. allowing monotonic extension to a more fully specified RMRS, up to the representation of a full logical formula, make this representation relatively complex. Copestake (2003) provides a detailed account of the RMRS formalism.

10 SAF as a Common Interface

A SAF file pools linguistic annotations from all levels of language processing and from distinct parsing strategies. In this role it provides a flexible interface in a heterarchical processing architecture that is not committed to a single pipelined path through a fixed succession of processes. While this is reminiscent of a blackboard architecture or more localised pool architectures, it is only a static representation of the linguistic annotations. It does not provide any specific mechanisms for accessing the information in the annotations, nor does it require any particular communications architecture in the processing modules. In fact, there may have been more efficient ways of representing a graph or lattice of linguistic annotations, if it were not for the fact that the linguistic components we employ were defined in a range of different programming languages, so that an external XML interface structure has priority over any shared data structure. Given this fact, SAF is a highly appropriate choice for the common interface that is a key feature of any multi-engine analysis architecture.

11 Conclusions

We have presented two interface structures used in the architecture of the SciBorg project. Each of these occupies a crucial position in the architecture. SciXML provides a uniform XML encoding for research papers from various publishers, so that all subsequent language processing only has to interface with SciXML constructs. SAF provides a common representation format for the results of various Language Technology components. We have emphasised the flexibility of these interfaces. For SciXML, this consists in conversion from publishers’ markup following a range of DTDs, as well as other formats, including HTML, LaTeX and even PDF. For SAF, the primary mark of flexibility is in allowing partial results from multiple parsers to be combined at a number of different levels.

12 Acknowledgements

We are very grateful to the Royal Society of Chemistry, Nature Publishing Group and the International Union of Crystallography for supplying papers. This work was funded by EPSRC (EP/C010035/1) with additional support from Boeing.

References

Briscoe, Ted, and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*.

Callmeier, U., A. Eisele, U. Schäfer, and M. Siegel. 2004. The DeepThought Core Architecture Framework. In *Proceedings of LREC-2004*, 1205–1208. Lisbon, Portugal.

Callmeier, Ulrich. 2002. Pre-processing and encoding techniques in PET. In Stephan Oepen, Daniel Flickinger, Jun'ichi Tsujii, and Hans Usz koreit, eds., *Collaborative Language Engineering: a case study in efficient gra mmar-based processing*. Stanford: CSLI Publications.

Clement, L., and E.V. de la Clergerie. 2005. MAF: a morphosyntactic annotation framework. In *Proceedings of the 2nd Language and Technology Conference*. Poznan, Poland.

Copestake, Ann. 2003. Report on the design of RMRS. DeepThought project deliverable.

Copestake, Ann, Peter Corbett, Peter Murray-Rust, C. J. Rupp, Advaith Siddharthan, Simone Teufel, and Ben Waldron. 2006. An Architecture for Language Technology for Processing Scientific Texts. In *Proceedings of the 4th UK E-Science All Hands Meeting*. Nottingham, UK.

Copestake, Ann, and Dan Flickinger. 2000. An open-source grammar development environment and broad-cover age English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, 591–600.

Corbett, Peter, and Peter Murray-Rust. 2006. High-throughput identification of chemistry in life science texts. In *Proceedings of the 2nd International Symposium on Computational Life Science (CompLife '06)*. Cambridge, UK.

Hollingsworth, Bill, Ian Lewin, and Dan Tidhar. 2005. Retrieving Hierarchical Text. 2005. Structure from Typeset Scientific Articles - a Prerequisite for E-Science Text Mining. In *In Proceedings of the 4th UK E-Science All Hands Meeting*, 267–273. Nottingham, UK.

Ruland, Tobias, C. J. Rupp, Jörg Spilker, Hans Weber, and Karsten L. Worm. 1998. Making the Most of Multiplicity: A Multi-Parser Multi-Strategy Architecture for the Robust Processing of Spoken Language. In *Proc. of the 1998 International Conference on Spoken Language Processing (ICSLP 98)*, 1163–1166. Sydney, Australia.

Rupp, C. J., Jörg Spilker, Martin Klarner, and Karsten Worm. 2000. Combining Analyses from Various Parsers. In Wolfgang Wahlster, ed., *Verbmobil: Foundations of Speech-to-Speech Translation*, 311–320. Berlin: Springer-Verlag.

Teufel, S., and N. Elhadad. 2002. Collection and linguistic processing of a large-scale corpus of medical articles. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002)*.

Teufel, Simone. 1999. Argumentative Zoning: Information Extraction from Scientific Text. Ph.D. thesis, School of Cognitive Science, University of Edinburgh, Edinburgh, UK.

Teufel, Simone, and Marc Moens. 1997. Sentence extraction as a classification task. In Inderjeet Mani and Mark T. Maybury, eds., *Proceedings of the ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, 58–65.

Townsend, Joe, Ann Copestake, Peter Murray-Rust, Simone Teufel, and Chris Waudby. 2005. Language Technology for Processing Chemistry Publications. In *Proceedings of the fourth UK e-Science All Hands Meeting (AHM-2005)*. Nottingham, UK.

Waldron, Benjamin, and Ann Copestake. 2006. A Standoff Annotation Interface between DELPH-IN Components. In *The fifth workshop on NLP and XML: Multi-dimensional Markup in Natural Language Processing (NLPXML-2006)*.

Waldron, Benjamin, Ann Copestake, Ulrich Schäfer, and Bernd Kiefer. 2006. Preprocessing and Tokenisation Standards in DELPH-IN Tools. In *Proceedings of the 5th International Conference on Language Re sources and Evaluation (LREC-2006)*. Genoa, Italy.