Representing and Querying Standoff XML

Stefanie Dipper[1], Michael Götze[1], Uwe Küssner[2], and Manfred Stede[1]

[1] Department of Linguistics
University of Potsdam
14415 Potsdam, Germany
{dipper,goetze,stede}@ling.uni-potsdam.de

[2] Uwe Küssner IT-Consulting
Markgrafendamm 24 Haus 16
10245 Berlin, Germany
uwekuessner@web.de

Abstract. The paper discusses the representation and exploitation of multi-level annotated linguistic data. We first present a standoff XML representation, which distributes information over separate, standoff layers and allows us to represent annotations of various kinds in a uniform, generic way. This format serves as our interchange format. We further introduce an XML-inline representation that is designed to provide for a more efficient processing of the data. This format is computed on the basis of the standoff representation and uses fragments to represent overlapping elements. We then compare both representations by testing their performance with regard to a testsuite. Not surprisingly, the inline variant performs much better than the standoff variant, in particular with more complex queries.

Key words: XML, standoff annotation, evaluation, querying

1 Introduction

In recent years, much corpus-based work has shifted attention from corpora annotated by part-of-speech and syntactic annotations only (treebanks) to corpora that are annotated by properties beyond the morphosyntactic level. Often this information is added to already-annotated corpora, which allows the user to combine the different types of annotation by posing cross-level queries and to search for putative interactions between different linguistic domains. Recent such corpora include semantic labelling (PropBank, Kingsbury and Palmer (2002); FrameNet, Johnson et al. (2003); SALSA, Erk et al. (2003)), pragmatic information (Penn Discourse TreeBank, Miltsakaki et al. (2004); RST Discourse Treebank, Carlson et al. (2003); Potsdam Commentary Corpus, Stede (2004)), and dialogue structure (Switchboard SWBD-DAMSL, Jurafsky et al. (1997)).

This scenario presupposes that new information can be integrated into the corpus. That is, the representations must be flexible and general enough to accommodate various kinds of information. At the same time, the format should support complex and efficient querying, also across levels.

In this paper, we present *PAULA*, our standoff XML format for data representation, which comes with various import filters for tool-specific formats (TIGER XML, RST Tool, MMAX2, Exmaralda[3]) and a generic importer for XML data, and export filters to statistical analysis (by WEKA[4]) as well as to our linguistic database ANNIS[5].

The paper is organized as follows: Section 2 introduces first our standoff interchange format and then an XML-inline variant that we derive from it. Section 3 compares the two formats with respect to efficiency of querying, on the basis of a testsuite of representative queries. Section 4 gives a short conclusion and outlook.

2 XML Representations

In our application scenario, we need to integrate data that has been annotated at various levels, and with different tools. That is, we have to deal with data heterogeneity due to (i) annotation tools with different output formats, and (ii) annotation layers with different formal properties (syntactic annotations are encoded by hierarchical structures like trees or graphs; speech and dialogue annotations often use time-aligned tiers; etc.).

Another difficulty stems from the fact that the segments that annotations are attached to may overlap. For instance, syntactic and prosodic segments are often encoded by trees, which in certain cases give rise to conflicting hierarchies. A simple example featuring overlapping segments is given in Figure 1: At the phonemic level (= third tier), tokens 3 and 4, *de la* ('of the'), are treated as one unit, with the annotation *dla*, whereas at the level encoding syntactic constituents, tokens 4–6, *la crème glacée* ('the ice-cream'), form an *NP* constituent, cf. tier 8.

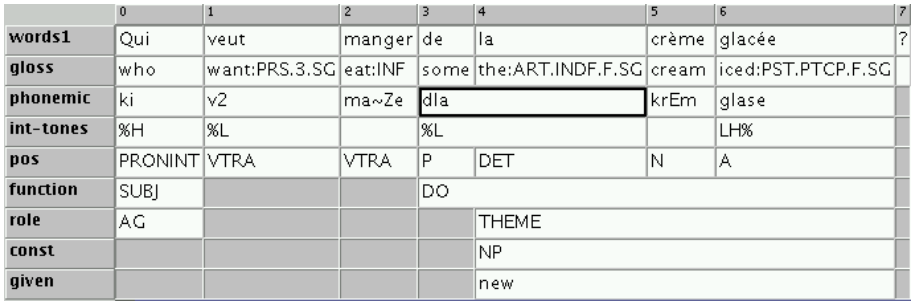
For representing multi-level annotations, models like the NITE Object Model (NOM, Carletta et al. (2003)), ATLAS (Laprun et al., 2003), or the *Linguistic Annotation Framework* (LAF, Ide et al. (2003)) have been developed. They define general, multi-rooted graphs whose nodes can be augmented by features, including timing information. To serialize these structures, XML-based formats have been designed: NITE-XML, the ATLAS Interchange Format AIF, and the "LAF dump format" (Ide and Romary, 2001), respectively. PAULA, the format we developed, is inspired by the LAF format. PAULA serves as an *interlingua* for the representation of all kinds of annotations.[6]

[3] TIGER-XML: `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch`
RST Tool: `http://www.wagsoft.com/RSTTool/`
MMAX2: `http://mmax.eml-research.de/`
Exmaralda: `http://www1.uni-hamburg.de/exmaralda/`

[4] WEKA: `http://www.cs.waikato.ac.nz/ml/weka/`

[5] ANNIS: `http://www.sfb632.uni-potsdam.de/annis/`

[6] The exact form of the LAF format is still under discussion; AIF is available in a beta version. In contrast to the other formats, NITE-XML does not define a *generic* XML representation but sticks more closely to the source data and, e.g., takes over tag names from the source.



	0	1	2	3	4	5	6	7
words1	Qui	veut	manger	de	la	crème	glacée	?
gloss	who	want:PRS.3.SG	eat:INF	some	the:ART.INDF.F.SG	cream	iced:PST.PTCP.F.SG	
phonemic	ki	v2	ma~Ze	dla		krEm	glase	
int-tones	%H	%L		%L			LH%	
pos	PRONINT	VTRA	VTRA	P	DET	N	A	
function	SUBJ			DO				
role	AG				THEME			
const					NP			
given					new			

Fig. 1. Overlapping segments: phonemic vs. syntactic units (screenshot of the annotation tool EXMARaLDA, http://www.rrz.uni-hamburg.de/exmaralda/)

All XML-based formats have to face the fact that overlapping segments cannot be represented by XML in a straightforward way (Barnard et al., 1995). In the next sections, we present two ways of dealing with this problem: by standoff and inline representation.

2.1 Standoff Representation

To integrate annotations from different sources, we developed *PAULA* ("Potsdamer Austauschformat für linguistische Annotation", 'Potsdam Interchange Format for Linguistic Annotation'), a standoff XML format that uses generic elements and attributes. In this format, `<mark>` elements ('markables') denote units of annotations, such as a span of words that constitute a phonemic phrase. `<feat>` elements specify the features that are annotated to the markables (e.g., the phonemic transcription of the markable); the `<feat>` elements are anchored to `<mark>` elements by means of XPointer expressions. `<struct>` elements specify structured markables, for representing trees or graphs. In the following, we briefly present our representation of the example in Figure 1.[7]

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE paula SYSTEM "paula_text.dtd">
<paula version="1.0">
  <header paula_id="exm-frn_text" id="TIE0" type="text"/>
  <body>Qui veut manger de la crème glacée ? ...</body>
</paula>
```

Fig. 2. Text file *exm-frn.text.xml*

[7] For a detailed presentation, see Dipper (2005) and `http://www.sfb632.uni-potsdam.de/projects/d1/paula/doc/`.

```
<markList type="tok" xml:base="exm-frn.text.xml">
   <mark id="tok_1"
    xlink:href="#xpointer(string-range(//body,'',1,3')))"/><!--Qui-->
   <mark id="tok_2"
    xlink:href="#xpointer(string-range(//body,'',5,4')))"/><!--veut-->
   ...
```

```
<markList type="phonemicSeg" xml:base="exm-fr.tok.xml">
   <mark id="phonemicSeg_1" xlink:href="#tok_1"/> <!-- Qui -->
   <mark id="phonemicSeg_2" xlink:href="#tok_2"/> <!-- veut -->
   <mark id="phonemicSeg_3" xlink:href="#tok_3"/> <!-- manger -->
   <mark id="phonemicSeg_4"
         xlink:href="#xpointer(id('tok_4')/range-to(id('tok_5')))"/>
   ...                                              <!-- de la -->
```

Fig. 3. Upper part: token file *exm-frn.tok.xml*, referring to the text file. Lower part: phonemic-segment file *exm-frn.phonemicSeg.xml*, referring to the token file

The source text is stored in one file, as the content of the element `<body>`, see Fig. 2. Token boundaries are defined by another file, referring to the text by XPointer expressions, see Fig. 3 (upper part). Each token is marked by an element `<mark>`, whose `id` attribute serves as the anchor for annotations of that token. Segments also make use of `<mark>` elements, see Fig. 3 (lower part). Segments may refer to *spans* of tokens; e.g. the fourth `<mark>` element refers to the sequence of tokens with IDs `tok_4` and `tok_5`.

Annotations, such as phonemic information, consist of `<feat>` elements that refer to `<mark>` elements by an `xlink:href` attribute and encode the value (e.g. *ki*) annotated to that token in the attribute `value`, see Fig. 4. The name of the annotated feature (such as *phonemic*) is encoded by the `type` attribute of the `<featList>` element. Each type of annotation is stored in a separate file.

```
<featList type="phonemic" xml:base="exm-frn.phonemicSeg.xml">
   <feat xlink:href="#phonemicSeg_1" value="ki"/> <!-- Qui -->
   <feat xlink:href="#phonemicSeg_2" value="v2"/> <!-- veut -->
   <feat xlink:href="#phonemicSeg_3" value="ma~Ze"/> <!-- manger -->
   <feat xlink:href="#phonemicSeg_4" value="dla"/> <!-- de la -->
   ...
```

Fig. 4. Phonemic annotation, referring to the phonemic-segment file *exm-frn.phonemicSeg.xml*

For hierarchical annotations like trees or graphs, we use the elements `<struct>` and `<rel>` to encode basic subtrees (our example does not contain such data structures). `<rel>` elements point to children dominated by the mother node

`<struct>`. Annotations attached to the subtrees, such as categorial or functional information, is again encoded standoff by `<feat>` elements.

The highly modularized nature and generality of our standoff representation allows us to incorporate all kinds of annotations and to incrementally add new layers to already-existing corpora. Moreover, it easily accommodates layers that define 'contradictory' information, such as overlapping segments, conflicting hierarchies, incompatible feature annotations of the same kind from different sources, etc. However, the generality has its price: Further processing of the data becomes expensive. Besides the standoff format, which serves as our interchange format, we therefore compute a supplementary internal *inline* representation, to provide for a more efficient querying of the data.

2.2 Inline Representation

A more "natural" way of representing the data is by XML *inline* representations. In such an approach, all annotations referring to the same token or span are collected and annotated as attributes of one element, as in the following (simplified) example, which collects all annotations referring to the token *crème*, cf. the previous figures.

```
<token id="tok_6" pos="N" gloss="cream" phonemic="krEm">crème</token>
```

The principles that guide the design of such an inline version are: (i) information (annotation) is encoded at the units that they refer to (i.e., without the use of pointers), as in the example above; (ii) no redundancy; (iii) use of the genuine XML data model (e.g., use of the XML-child relation to encode dominance relation in trees).

PAULA-inline, the inline format we have developed, adheres to these principles *cum grano salis*: It uses pointers, e.g., for the encoding of non-local dependencies. Moreover, our data contains two types of "structural" relation, the classical dominance relation, as realized by syntactic trees, plus an "inclusion" relation. The inclusion relation relates a "larger" segment with a "smaller" segment, if the larger one *extensionally* includes the smaller one. We prefer the term *alignment* for this relation, since it does not make any statements about dominance: Co-extensional segments include each other symmetrically.

PAULA-inline uses the XML-child relation to encode the alignment relation. To mark "real" embedding in structures such as trees (PAULA `<struct>` elements), we use the special element `<_rel>`, which explicitly encodes the dominance relation in PAULA-inline. This allows us to annotate *edges* between nodes.

Our example from the previous section, encoded in PAULA-inline, actually looks more complex than the simplified example above, due to the extra segment layers that we use, such as the phonemic-segment markable layer. The corresponding fragment from the inline file is shown in Figure 5.

We stated above that XML embedding cannot be used for the representation of overlapping segments. For such data, we use the strategy of *fragmentation* (cf. Sperberg and Burnard (1994), Barnard et al. (1995)): One of the overlapping

```
<posSeg id="posSeg_6" pos="N">
  <glossSeg id="glossSeg_6" gloss="cream">
    <phonemicSeg id="phonemicSeg_5" phonemic="krEm">
      <tok id="tok_6">crème</tok>
    </phonemicSeg>
  </glossSeg>
</posSeg>
```

Fig. 5. Extract from the PAULA-inline representation

elements is broken into smaller units and an attribute `_gid` ('group id') is added to the fragmented elements to explicitly mark elements that belong together. In our example, the overlapping segments, the `<phonemicSeg>` (*dla*) and the `<constSeg>` (*NP*) annotations, are encoded as shown in Fig. 6. The segment (*dla*) is split into two XML elements, the *NP* segment remains intact.

```
<phonemicSeg id="phonemicSeg_4" phonemic="dla"
            _gid="id_21" _type="fragment" _discont="no" _overlap="yes">
   <tok id="tok_4">de</tok>
</phonemicSeg>
<constSeg id="constSeg_1" const="NP">
   ...
   <phonemicSeg _gid="id_21" _type="fragment" _id="id_21_0">
      <tok id="tok_5">la</tok>
   </phonemicSeg>
   ...
</constSeg>
```

Fig. 6. Overlaps in PAULA-inline

Generating the inline representation Since both the standoff source data and the target inline data are XML-based representations, we decided to use XQuery for the generation of the inline data. The process consists of several phases: In the first phase, all nodes are expanded, i.e., the *transitive closure* is computed. This enables us to easily compare the relationship between all segments, i.e., to determine whether one segment includes the other, whether they overlap, etc. In the next phase, we compute the *transitive reduction*, i.e., the smallest graph with the same transitive closure. Here, the relations of direct predecessor and successor in the graph are determined. In the third phase, the target tree is built bottom-up. A simple heuristic determines the segments that result in fragments (due to overlaps), by minimizing the number of crossing edges. Fig. 7 shows that it is more favorable to fragment the left tree because this only involves

cutting one edge, as opposed to six edges if the right tree were fragmented. The fragments are then woven into the tree, and the tree is finally output.

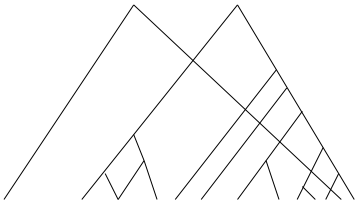


Fig. 7. Two overlapping trees

Today, there are just a few tools that support creating inline versions of standoff annotations, e.g. LT XML[8]. However, LT XML does not allow for conflicting hierarchies. Witt et al. (2005) present a tool that merges exactly two conflicting XML hierarchies by replacing one of the annotations by milestones or fragments. In our application scenario, which involves corpora with multiple annotation layers, merging of more than two XML trees at the same time is crucial.

3 Evaluation

In this section, we examine our formats for their suitability for linguistic querying, focussing here on response time. In particular, the goal is to examine to what extent the intuitively expected performance differences between standoff and inline format are confirmed by practical experience. Our basic approach is to investigate the feasibility of querying linguistic XML data with current off-the-shelf XML technology, without spending effort on optimizing the tools for specific tasks.[9]

3.1 Evaluation Setup

We base our evaluation on two corpora: the *Potsdam Commentary Corpus* (PCC), a newspaper commentary corpus annotated with syntax, co-reference, and discourse structure (Stede, 2004), and the *TIGER Corpus*, annotated with syntax, including part of speech, lemma and morphology (Brants et al., 2004).[10]

[8] `http://www.ltg.ed.ac.uk/software/xml/`

[9] In a similarl fashion, Teich et al. (2001) perform queries on standoff annotations, by applying XQuery expressions to multiple documents. However, they do not report on the performance of the queries.

[10] The PCC consists of 176 articles with a total of roughly 2,200 sentences = 33,000 tokens. Structural annotations (syntax and discourse structure) consist of 18,000 non-terminal nodes, with 50,000 edges; segmental annotations (used in the discourse

Inspired by Bird et al. (2006), our testsuite (see Table 1) consists of 7 queries of varying complexity, with queries involving hierarchical (Q5), pointing (Q7) and overlapping relations (Q6). Some queries differ with respect to the return value: They either return the *reference* to an XML element (Q3) or else the XML subtree dominated by the element (Q4).

For the evaluation, we chose the standard XML query language *XQuery*, and run the XQuery processor *Saxon 8.7.1J* on a 3.06 GHz machine with 4 GB RAM and 2 Intel Xeon processors. Queries were stored in a file and sent to the XQuery processor; the result list was output to a file. Queries were repeated 7 times, and we report here the average evaluation time.

Q1	Find all sentences that include the word "kam".
Q2	Find all sentences that do not include the word "kam".
Q3	Find all NPs. Return the reference to that NP.
Q4	Find all NPs. Return the subtree dominated by that NP.
Q5	Find all NPs dominated by a VP. Return the subtree dominated by that NP.
Q6	Find all constituents whose extensions overlap.
Q7	Find all pairs of anaphors and direct antecedents, where the anaphor is a personal pronoun.

Table 1. Query testsuite

3.2 Results

Table 2 shows the execution times for the query testsuite, for the data sets of the PCC and the TIGER corpus. The results show that the inline version fares better than the PAULA standoff format in all cases. The discrepancy between standoff and inline format especially shows up with "complex" queries, such as Q6, which involves overlapping elements. With large corpora, execution times for the standoff version are unacceptably slow. In principle, much depends on the way the queries are formulated. We did not concentrate on tuning the queries to improve performance, so better results might be achievable. Still, the basic difference remains.

Querying standoff data usually involves considerably complex query expressions, even for apparently simple questions like Q1. The reason is that querying standoff data usually involves following up the links between the different layers

and co-reference annotations) consist of 12,000 markables. The structures and markables are augmented by a total of 220,000 feature annotations.

The complete TIGER Corpus contains 50,500 sentences with a total of 900,000 tokens. Syntactic annotations consist of 375,000 non-terminals, with 1,000,000 edges. In our evaluation, we used only 30,000 sentences of the TIGER Corpus.

(text, markables, features, ...). This results in cumbersome query expressions as well as rather poor performance. One way to deal with the first problem is to modularize the code and use *templates* (parametrized functions). The second problem might be tackled by the using indices.

	PCC176			TIGER Corpus		
Query	standoff	inline	# of hits	standoff	inline	# of hits
Q1	20.6	3.3	8	†	51.2	28
Q2	21.9	3.7	3,276	†	65.7	18,844
Q3	4.5	3.3	3,794	80	52.5	62,710
Q4	19.9	4.9	3,794	†	73.8	62,710
Q5	10.9	4.9	934	†	119.6	18,755
Q6	73.2	12.2	830	NA	NA	NA
Q7	4.5	3.9	370	NA	NA	NA

Table 2. Performance: results of querying, in sec. (†: query did not terminate due to stack overflow; NA: not applicable)

4 Conclusion and Outlook

We presented PAULA, an XML-based standoff representation format which we use as an interlingua for the representation of data annotated on multiple levels. It covers different data structures as well as conflicting hierarchies in a straightforward way. However, it does not support efficient processing. For this purpose, we designed PAULA-inline, which represents the same data, without loss of information, in an inline-version. Our evaluation experiment shows that it performs considerably better than the standoff version. The results also indicate that our queries can be applied to small corpora, whereas the approach does not scale up to larger amounts of data, such as the TIGER Corpus. This is not surprising, though, as we are processing files sequentially, and not with a dedicated database.

Better performance and readibility of XQuery searches could be achieved if the inline version is enriched by certain redundant information. For instance, a simple query like `$node//*:const[@_type="fragment"]` retrieves all fragmented elements that are dominated by `$node`. If we now want to know whether the fragmented elements are contained *completely* in the element `$node`, the entire document has to be searched for "sister fragments" of the respective fragmented elements. If we encode the information explicitly in the inline version (e.g., we explicitly mark start and end of a fragmented constituent), processing will speed up. In general, however, the desire for more promising response times calls for using a proper database, which is one focus of our current work.

Bibliography

Barnard, D., Burnard, L., Gaspart, J.-P., Price, L. A., Sperberg-McQueen, C. M., and Varile, G. B. (1995). Hierarchical encoding of text: Technical problems and SGML solutions. *Text Encoding Initiative: Background and Context. Special Issue of Computers and the Humanities*, 29(211–231).

Bird, S., Chen, Y., Davidson, S., Lee, H., and Zheng, Y. (2006). Designing and evaluating an XPath dialect for linguistic queries. In *Proceedings 22nd International Conference on Data Engineering (ICDE)*, Atlanta, USA.

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Carletta, J., Kilgour, J., O'Donnell, T., Evert, S., and Voormann, H. (2003). The NITE Object Model library for handling structured linguistic annotation on multimodal data sets. In *Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003)*.

Carlson, L., Marcu, D., and Okurowski, M. E. (2003). Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In van Kuppevelt, J. and Smith, R., editors, *Current and New Directions in Discourse and Dialogue*. Kluwer Academic Publishers.

Dipper, S. (2005). XML-based stand-off representation and exploitation of multi-level linguistic annotation. In *Proceedings of Berliner XML Tage 2005 (BXML 2005)*, pages 39–50, Berlin, Germany.

Erk, K., Kowalski, A., Padó, S., and Pinkal, M. (2003). Towards a resource for lexical semantics: A large German corpus with extensive semantic annotation. In *Proceedings of ACL 2003*, Sapporo, Japan.

Ide, N. and Romary, L. (2001). A common framework for syntactic annotation. In *Meeting of the Association for Computational Linguistics*, pages 298–305.

Ide, N., Romary, L., and de la Clergerie, E. (2003). International standard for a linguistic annotation framework. In *Proceedings of HLT-NAACL03 Workshop on The Software Engineering and Architecture of Language Technology*.

Johnson, C. R., Petruck, M. R. L., Baker, C. F., Ellsworth, M., Ruppenhofer, J., and Fillmore, C. J. (2003). Framenet: Theory and practice. Version 1.1.

Jurafsky, D., Shriberg, E., and Biasca, D. (1997). Switchboard SWBD-DAMSL shallow-discourse-function annotation. Coders manual, draft 13. Technical Report 97-02, University of Colorado.

Kingsbury, P. and Palmer, M. (2002). From TreeBank to PropBank. In *Proceedings of LREC 2002*, Las Palmas,Spain.

Laprun, C., Fiscus, J., Garofolo, J., and Pajot, S. (2003). Recent improvements to the ATLAS architecture. Technical report, National Institute for Standards and Technology.

Miltsakaki, E., Prasad, R., Joshi, A., and Webber, B. (2004). The Penn Discourse TreeBank. In *Proceedings of LREC 2004*, Lisbon, Portugal.

Sperberg, C. M. and Burnard, L., editors (1994). *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*. Text Encoding Initiative, Chicago, Oxford.

Stede, M. (2004). The Potsdam Commentary Corpus. In *Proceedings of the ACL Workshop on Discourse Annotation*, pages 96–102, Barcelona.

Teich, E., Hansen, S., and Fankhauser, P. (2001). Representing and querying multi-layer corpora. In *Proceedings of IRCS Workshop on Linguistic Databases*, pages 228–237, University of Pennsylvania, Philadelphia, USA.

Witt, A., Goecke, D., Sasaki, F., and Lüngen, H. (2005). Unification of XML documents with concurrent markup. *Literary and Linguistic Computing*, 20(1):103–116.