An OWL- and XQuery-Based Mechanism for the Retrieval of Linguistic Patterns from XML-Corpora

Georg Rehm
SFB 441 Linguistic Data Structures
Tübingen University
Nauklerstrasse 35
72074 Tübingen, Germany
georg.rehm@uni-tuebingen.de

Richard Eckart
Dept. of English Linguistics
TU Darmstadt
Hochschulstrasse 1
64289 Darmstadt, Germany
eckart@linglit.tu-darmstadt.de

Christian Chiarcos
SFB 632 Information Structure
Potsdam University
Karl-Liebknecht-Strasse 24–25
14476 Potsdam, Germany
chiarcos@uni-potsdam.de

Abstract

We present an approach for querying collections of heterogeneous linguistic corpora that are annotated on multiple layers using arbitrary XML-based markup languages. An OWL ontology is used to homogenise the conceptually different markup languages so that a common querying framework can be established.

Keywords

Corpora, Corpus analysis, XML, querying, XQuery, OWL, ontologies, multi-rooted trees, annotation, multi-level annotation

1 Introduction

Annotated linguistic corpora can be used in several different scenarios: they can be employed in machine learning contexts to serve as training data, they can be used to build language models based on statistical properties, or corpora can serve as a resource in computer-assisted language learning software. In fact, there are so many possible ways in which corpora can be used effectively that their initial purpose has become overshadowed rather quickly. Traditionally, linguists compiled corpora in order to find answers for research questions on the basis of empirical evidence. After a corpus had been compiled using a number of criteria, it could be analysed using statistical methods.

We are concerned with devising a web-based corpus platform for a large collection of more than 60 heterogeneous linguistic corpora. One of the obstacles we are confronted with deals with exploring ways of providing homogeneous means of accessing this very large collection of diverse and complex linguistic resources. The user interface does not only have to generalise over several heterogeneous annotation formats, it has to be intuitively usable for linguists without expertise in XML, querying standards such as XQuery (see, e. g., [15]), or even the original markup languages. In other words, we want to lay a technical foundation for the interoperability and reusability of annotated linguistic corpora. We would like to enable academics who are not interested in the corpus annotation specifics to log onto the platform and to explore as well as to query the available corpora in an efficient and simple way.

Section 2 briefly highlights the most important properties of data formats for linguistic corpora and our generic data model. Section 3 sketches the general approach, our system architecture, and the process flows. The main part of this paper, section 4, discusses the web-platform's query interface: first, we illustrate the technical aspects of querying multi-rooted trees. We subsequently introduce an ontology-based approach for homogenising the heterogeneous markup languages. Finally, we sketch the graphical interface and the output and visualisation modules.

2 A Homogeneous Data Model

Since the late 1990s, practically all corpus annotation formats have been realised as XML markup languages [11, 13, 20]. They come in two different flavours: traditionally, most corpus markup languages form hierarchies that are expressed by nested XML element trees (e. g., for the representation of syntactic constituents or document structures). In stark contrast to hierarchical data formats are markup languages that anchor a data set to a timeline (primarily used for the transcription of spoken language), see [2]. In timeline-based formats such as Exmaralda [18], the annotator can draw an arc from one anchor to another point on the timeline. However, these structures are not represented by nested XML element-trees, but with the help of attribute-value pairs. At the same time, both approaches usually encode several annotation layers concurrently, for example, information on morphological, syntactic, semantic, and pragmatic structures.

In our project we have to deal with both hierarchical and timeline-based corpora and we have to provide the means for enabling users to query both types of resources in a uniform way. In fact, the original annotation format will be irrelevant to the user, as the user interface and the underlying technology will abstract from any idiosyncrasies and peculiarities of the original data formats. We use an approach that is able to cope with the abovementioned difficulties [8, 19, 22] and that can be compared to the NITE Object Model [4]. We developed a tool that semiautomatically splits hierarchically annotated corpora that typically consist of a single XML document instance, into individual XML files, so that each file represents all the information related to a single annotation layer [21]; this approach guarantees that overlapping structures can be represented straightforwardly. Timeline-based cor-

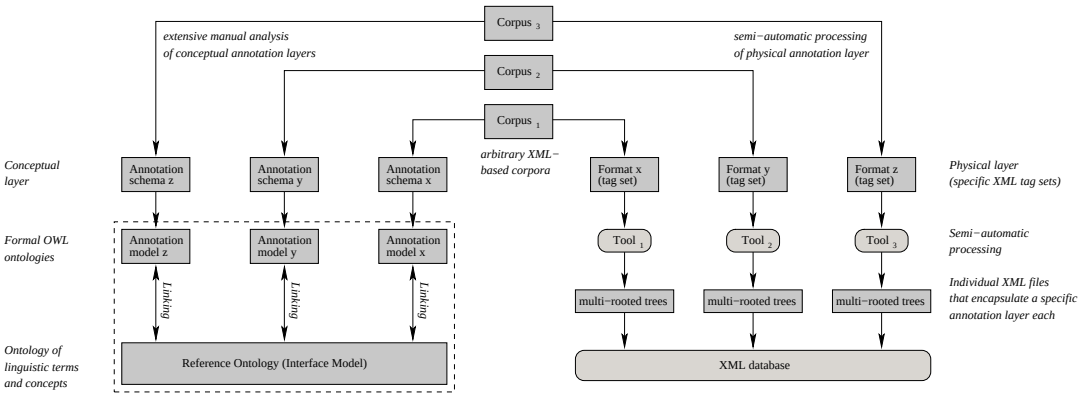

Fig. 1: *The two main corpus processing workflows*

pora are processed using another tool in order to separate the graph annotations that are also stored in individual XML files [21]. Our approach enables us to represent arbitrary types of XML-annotated corpora as individual files, i. e., individual XML element trees. These multi-rooted trees are represented as regular XML document instances, but, as a single corpus comprises *multiple* files, there is a need to go beyond the functionality offered by typical XML tools in order to enable us to process multiple files, as regular tools work with single files only.

3 System Architecture

First, a corpus to be imported into our corpus platform has to be analysed manually (figure 1). Depending on its corresponding markup language, the XML document instance is transformed into multi-rooted trees.

Some corpora can be transformed using simple XSLT stylesheets, while other corpora have to be processed using a custom set of tools: corpora annotated based on the hierarchical model are analysed by a tool that enables us to map XML elements, attributes and textual content onto one or more annotation layers. As soon as this mapping exists, the annotation layers can be exported as XML documents. A second tool can be used to split timeline-based corpora into a set of multi-rooted trees. Finally, these XML files are imported into an XML database (e. g., eXist). A third tool anchors all files to a set of primary data in order to allow query-time coordination between the individual files that represent a single-rooted tree each.

At the same time, the elements and attributes used in the markup languages are analysed and incorporated into an ontology that encapsulates knowledge about linguistic terms and concepts. The ontology is used to generalise over the specific and, at times, idiosyncratic names and labels used in the corpus markup languages and to provide a coherent, unified, and homogeneous perspective on the large set of heterogeneous corpora.

4 The Query Interface

There are several constraints for the web-based query interface we are currently developing. For this paper the two most important issues are the implementation of a mechanism that enables XQuery queries that work on multi-rooted trees (section 4.1) and the integration of the ontology of linguistic annotations into the process of building an XQuery statement (section 4.2). In addition, we want to provide a graphical interface that can be intuitively used by linguists and other interested parties who know neither XML, XQuery, nor the XML-based markup languages used in the original corpora (section 4.3). Figure 2 shows the architecture of the query interface. We modified the XML database eXist so that it is able to cope with directing XQuery queries over multi-rooted trees.

4.1 Querying Multi-Rooted Trees

As each annotation layer is contained in one XML document, a corpus represents a special form of a multi-rooted tree, i. e., a collection of trees that do not share nodes except the leaves containing annotated data. AnnoLab [9] is an XML/XQuery-based corpus query and management framework designed to deal with multi-rooted trees. An abstract data-model for corpus annotation was synthesized from various approaches (e. g., [4], [12], [14]) and consists of four tiers: (i) signal tier (annotated data), (ii) structure tier (annotation structure), (iii) feature tier (annotation features), (iv) location tier (a mapping between signal and structure tiers). XML's data-model itself, however, supports only three of the four tiers: signal (text-nodes), structure (element hierarchy), and feature tier (attributes). Furthermore, it combines the tiers into an ordered tree with non-overlapping leaves, leading to problems regarding projectiveness and overlapping segments. By introducing the location tier as a buffer between signal and structure, these problems can be resolved. In addition, the text-nodes from the XML data-model are replaced by *segments* that serve as placeholders for the signal, thus functioning as stand-off *anchors*. A segment addresses a signal using start and end offsets as well as a *signal identifier*. The rest of the XML data-model remains untouched, so that standard XQuery statements can be used. Assuming that an XML annotation contains the annotated text in document order in its text nodes, the conversion to the AnnoLab format (and back) can be done fully automatically.

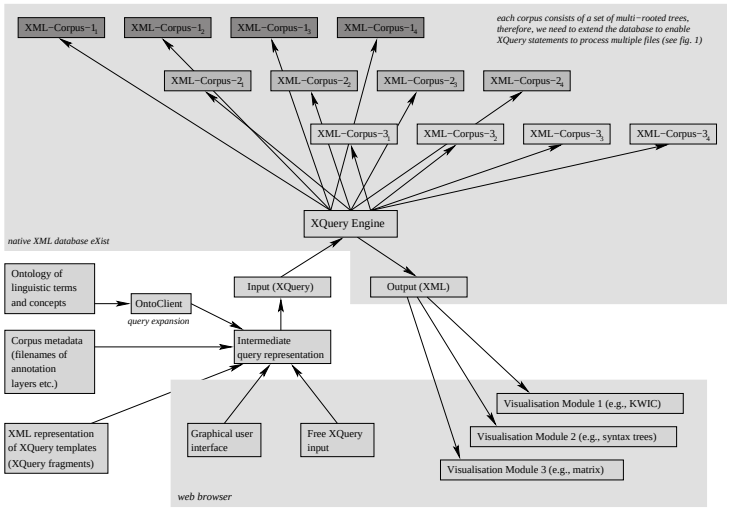

Fig. 2: *Architecture of the web-based query interface*

4.1.1 XQuery Extensions

To access signals and to perform queries across multiple layers, AnnoLab provides a library of XQuery functions that are loaded into eXist as extensions. These extensions fall into two categories: (i) accessing the signal, (ii) coordinating queries across layers.

Signal access – To this category belong functions such as `get-text(N)` and `find-text(N, p)`. The first function takes as an argument a set of elements `N`. It collects all *segments* located under `N` and returns the text they address. The second function takes a set of elements `N` and a pattern `p`. It returns those segments under `N` that address text matching `p`.

Layer coordination – The functions in this category perform comparisons and calculations on *segments*. The function `overlapping(X, Y)` illustrates the general principle: it takes two sets of elements `X` and `Y`. These sets are expanded into two *segment* lists `A = seg(X)`, `B = seg(Y)` that contain all segments under `X` and `Y`. It returns all `a` in `A` that overlap with some `b` in `B`. Analogous functions exist for all 13 temporal relations formalized by Allen [1]. The functions can be used to specify the desired relations between *segments* originating from different annotation layers and, thus, to coordinate different layers.

All extension functions could be implemented in pure XQuery, however, for performance reasons and limitations in eXist, they were implemented in Java.

4.1.2 Query Example

For the following example [9] assume an alignment layer `en_de.align` (see figure 3); its segments refer to two signals `de` (*Deutsch*, German) and `en` (English). Another layer `en.pos` contains `token` elements that have a `pos` feature (part-of-speech data for `en`).

The query (figure 4) yields all verb forms in the English text that are one or two tokens to the left of a determiner along with their translations into German. The query selects all tokens (`$eng`) from the POS layer and all alignments (`$aln`) from the alignment layer. The result set contains those combinations of segments and alignments that fulfill the specified conditions:

- line 4: the English part of the alignment layer has to overlap with a token from the part-of-speech layer,
- line 5: the token from the part-of-speech layer has to be a verb form (`pos` feature starting with `V`),
- lines 3+6: the first or second following token (`$next`) from the part-of-speech layer has to be a determiner (`pos` feature starting with `DT`).

This example demonstrates that using AnnoLab's XQuery extensions results in rather complex query statements that require a certain amount of XQuery knowledge. Each query depends on a consistent set of annotation elements, feature names, and feature values.

```
  <signal id="de">Er schloss das Tor ab</signal>
2 <signal id="en">He locked the gate</signal>
  <layer id="en_de.align">
4  <alignment>
    [...]
6   <align>
     <i role="de">
8     <seg start="3" end="9" sig="de">schloss</seg>
      <seg start="19" end="20" sig="de">ab</seg>
10   </i>
     <i role="en">
12     <seg start="3" end="8" sig="en">locked</seg>
     </i>
14  </align>
    [...]
16 </layer>
```

Fig. 3: *Abbreviated alignment layer and signals*

```
  for    $eng in ds:layer("en.pos")//token,
2        $aln in ds:layer("en_de.align")//align
  let    $next := $eng/following::token[position()<2]
4 where  seq:overlapping($eng, $aln//i[@role="en"])
    and  starts-with($eng/@pos, "V")
6   and  starts-with($next/@pos, "DT")
  return
8   <t>
      <eng>{txt:get-text($eng)}</eng>
10    <ger>{txt:get-text($aln//i[@role="de"])}</ger>
    </t>
12 ------------------------------------------------
    <t>
14    <eng>locked</eng>
      <ger>schloss ab</ger>
16  </t>
```

Fig. 4: *Query aligned signals using pos constraints*

4.2 Creating XQuery Constraints

In order to provide a consistent approach for documentation and to enable a uniform query interface that applies to different annotation formats, we built an ontology that serves as a terminological reference, represented in OWL DL (see [6, 10] for similar approaches). This *reference model* is based on the EAGLES recommendations for morphosyntax, the general ontology for linguistic description [10], and the SFB632 annotation standard [7]. Currently it includes reference specifications for word classes, morphosyntax [5], and will be extended to other linguistic phenomena.

The *reference model* consists of three parts: a taxonomy of linguistic categories (modelled as OWL classes, e. g., NOUN, COMMONNOUN), a taxonomy of grammatical features (OWL classes, e. g., ACCUSATIVE), and relations (OWL properties, e. g., HASCASE). An *annotation model* is an ontology that represents one specific annotation scheme. We built, among others, formalised annotation models for the SFB632 annotation format [7], TIGER/STTS [17, 3], SUSANNE [16], and for the Uppsala corpus tagset. Annotation models include word classes, grammatical features, and relations. However, this structure is independent from the reference model as it relies on the original annotation documentation only. It can be seen as a formal interpretation of the annotation scheme (see figure 1).

In contrast to the reference model, annotation models include instances. Every instance corresponds to a tag or an annotation value in the original annotation scheme. It is augmented with the properties HASTAG and HASTIER, which provide the exact surface form of the corresponding annotation (e. g., `hasTag(VVZv)`) and the conceptual layer (e. g., `hasTier(pos)`). Instances are characterized by the word class they are assigned to (e. g., susa:LEXICALVERB and susa:FINITEVERB) and grammatical properties (e. g., susa:HASPERSON(susa:THIRD), and susa:HASNUMBER(susa:SINGULAR)).

Annotation models and the reference model are linked by RDF descriptions (`rdfs:subClassOf`, `rdfs:subPropertyOf`): an annotation model acts as one specific instantiation of the reference model. This linking mechanism can also be applied to use definitions from external reference ontologies such as GOLD [10] as an optional *upper model* or *external reference model*. The internal reference model's purpose is to mediate between resource or language-specific annotation models and an external upper model. For the specification of queries, definitions provided by an external reference model may decrease the initial reluctance a user might have to work with the ontology.

Ontology-Based Corpus Querying

According to the structure of the ontologies, any tag used in an annotation scheme corresponds to an indirect instance of a class in the reference model, which might be subject to further specification by (sub)properties of the reference model. Accordingly, any tag from an annotation model can be retrieved by a description in terms of OWL classes and properties from the reference model. If multiple annotation models are considered, such a description may be expanded into a disjunction of tags from different tag sets or conceptual layers. OntoClient, a highly configurable query preprocessor implemented in Java, retrieves all individuals which correspond to an ontology-based description and translates them into a disjunction of tags. OntoQueries can be embedded in arbitrary code which remains untouched during query expansion. OntoClient's input as well as the output are specified by formal grammars. In the input, ontology-sensitive sub-queries are marked by curly braces, with the opening parenthesis followed by the CUE, e. g., a variable that describes the element whose attributes and attribute values are defined by the ontological description, the key word `in`, and an expression composed of ontological classes and properties.

RESULT := `(CUE/@TIER="TAG" (or CUE/@TIER="TAG")*)`

For every individual retrieved from the expansion of the OntoQuery expression, TIER and TAG are the values of the corresponding HASTIER and HASTAG properties. CUE is identical to a CUE element in the OntoQuery, thus, it has to be specified by the user.

```
  for    $eng in ds:layer("en.pos")//token,
2        $aln in ds:layer("en_de.align")//align
  let    $next := $eng/following::token[position()<2]
4 where  seq:overlapping($eng, $aln//i[@role="en"])
    and  {$eng in Verb}
6   and  {$next in Determiner}
  return [...]
```

Fig. 5: *Incorporating ontology-driven constraints into a query (modified version of the query shown in fig. 4)*

Figure 5 shows a modified version of the sample query: for `{$eng in Verb}` (line 5), OntoClient searches for the concept VERB in the reference model. Considering the SUSANNE annotation model, susa:VERB is retrieved as a subclass of the reference model concept, with sub-classes susa:LEXICALVERB, susa:MODALVERB, etc., which expand to a total of 44 instances (e. g., susa:VVZV is retrieved as an instance of LEXICALVERB). The value of HASTAG() specifies the surface form of its tag, i. e., `VVZv`, the value of HASTIER() specifies the conceptual layer, i. e., `pos`. OntoClient produces the following constraints:

```
($eng/@pos = "VVZv" or $eng/@pos = "VV0"
or $eng/@pos = "VV0i" [...])
```

Here, `$eng` is the cue from the original query, `pos` is the value of the property `hasTier`, and `VVZv` is the value of `hasTag`. In the additional annotation models, corresponding tags are listed as well, including multiple conceptual layers and a greater variety of tags.

OntoClient was originally developed as a preprocessor for corpus querying languages such as CQP, TIGERSearch, and ANNIS-QL, which are tailored to the needs of corpus linguists. However, OntoClient can be applied as a more general query preprocessor in order to produce XQuery constraints.

4.3 The Graphical Interface

We cannot expect our primary user group (i. e., linguists) to be proficient in XML-related querying languages such as XQuery. Instead, we want to provide an intuitive user interface that generalises as much as possible from the underlying data structures and querying

methods actually used. Our system will make heavy use of Ajax technologies (Asynchronous JavaScript and XML) so that a dynamic, interactive, drag-and-drop-enabled query interface can be provided. As the ontology of linguistic annotations (section 4.2) is a resource for homogenising heterogeneous markup languages, we will be able to provide abstract graphical representations of linguistic concepts (e. g., "noun", "verb", "preposition" etc.) that may have a specific set of features; furthermore, we will provide operands so that the linguistic concepts can be glued together by dragging and dropping these graphical representations onto a specific area of the screen, building a query step by step. In addition, users will be able to enter all kinds of annotated linguistic information, e. g., specific text, feature values, syntactic relations etc. (where possible, the information to be presented to the user will be constructed from the ontology). The abovementioned linguistic concepts as well as the operands are associated with XPath and XQuery fragments so that, after a query has been specified using this graphical interface, the individual fragments can be assembled into the final XQuery statement.

We want to provide several output and visualisation modules for query results, e. g., we will visualise queried corpus subsets that contain syntactic trees as trees, realised as SVG graphics, and we plan to represent data that is modelled using a timeline-based approach in a tabular fashion that highlights overlapping structures. One conceptual obstacle concerns the fact that, just like SQL, XQuery queries specify the output part of a query. We plan to introduce a processing layer that represents complex search result datatypes: as soon as each query template is associated with one such search result datatype (e. g., "syntax tree", "matrix", "kwic" etc.), we are able to map a specific query template onto a specific output or visualisation module so that the search result datatype specifies which output modules can be used.

For the representation of the query templates we will use an XML-based format in order to store all necessary data in one place: (a) the query template itself (i. e., an XQuery fragment, its associated linguistic concepts, and "free" corresponding variables); (b) the search result datatype; (c) function of the query (its linguistic scope); (d) source of the query; (e) the annotation layer the query refers to (e. g., syntax, information structure etc.); (f) instructions or a general description of the query; (g) one or more sample queries built upon the query template (i. e., example variable assignments that can be modified by the user).

5 Concluding Remarks

We presented an approach to querying XML-annotated corpora using standard techniques such as XPath and XQuery. As modern corpora are annotated on several layers, we extended a native XML database so that multi-rooted trees, representing one such annotation layer each, can be queried. One of our goals is to provide an intuitive, modern, flexible, and powerful search interface. As our web-platform has to cope with arbitrary annotation formats, we built an OWL ontology that encapsulates knowledge about the tag sets used in these annotation schemes. The ontology can be used for query expansion, so that knowledge of the underlying data formats is not required.

Acknowledgments

Part of the research on AnnoLab was supported by a grant from *Deutsche Forschungsgemeinschaft* (DFG) within the project *Linguistische Profile interdisziplinärer Register* (Darmstadt University of Technology). Part of the research presented was supported by a grant from *Deutsche Forschungsgemeinschaft* (DFG) within the project *Nachhaltigkeit linguistischer Daten*.

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] S. Bird and M. Liberman. A Formal Framework for Linguistic Annotation. *Speech Communication*, 33(1/2):23–60, 2001.

[3] S. Brants, S. Dipper, P. Eisenberg, S. Hansen-Schirra, E. König, W. Lezius, C. Rohrer, G. Smith, and H. Uszkoreit. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, 2003.

[4] J. Carletta, J. Kilgour, T. J. O'Donnell, S. Evert, and H. Voormann. The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets. In *Proc. of t. EACL Workshop on Lang. Techn. a. t. Sem. Web*, 2003.

[5] C. Chiarcos. An Ontology of Linguistic Annotation: Word Classes and Morphology. In *Proc. of DIALOG 2007*, 2007.

[6] G. A. de Cea, G.-P. Asunción, I. Álvarez de Mon, and A. Pareja-Lora. Ontotag's linguistic ontologies: Improving semantic web annotations for a better language understanding in machines. In *Proc. of ITCC'04*, pages 124–128, 2004.

[7] S. Dipper, M. Götze, and S. Skopeteas, editors. *Information Structure in Cross-Linguistic Corpora: Annotation Guidelines for Phonology, Morphology, Syntax, Semantics, and Information Structure*, volume 7 of *ISIS*. 2007.

[8] S. Dipper, E. Hinrichs, T. Schmidt, A. Wagner, and A. Witt. Sustainability of Linguistic Resources. In E. Hinrichs, N. Ide, M. Palmer, and J. Pustejovsky, editors, *Proc. of LREC 2006 Satellite Workshop Merging and Layering Linguistic Information*, pages 48–54, Genoa, Italy, May 2006.

[9] R. Eckart and E. Teich. An XML-Based Data Model for Flexible Representation and Query of Linguistically Interpreted Corpora. In G. Rehm, A. Witt, and L. Lemnitzer, editors, *Data Structures for Linguistic Resources and Applications*. Gunter Narr, Tübingen, Germany, 2007.

[10] S. Farrar and D. T. Langendoen. A Linguistic Ontology for the Semantic Web. *GLOT International*, 3:97–100, 2003.

[11] N. Ide, P. Bonhomme, and L. Romary. XCES: An XML-based Standard for Linguistic Corpora. In *Proc. of LREC 2000*, pages 825–830, Athens, 2000.

[12] C. Laprun and J. Fiscus. Recent improvements to the atlas architecture. In *Proc. of HLT 2002*, San Diego, 2002.

[13] T. Lehmberg and K. Wörner. Annotation Standards. In A. Lüdeling and M. Kytö, editors, *Corpus Linguistics*, Handbücher zur Sprach- und Kommunikationswissenschaft (HSK). de Gruyter, Berlin, New York, 2007. In press.

[14] W. Lezius. *Ein Suchwerkzeug für syntaktisch annotierte Textkorpora*. Ph.D. thesis, University of Stuttgart, 2002.

[15] J. Melton and S. Buxton. *Querying XML*. Morgan Kaufmann, Amsterdam etc., 2006.

[16] G. Sampson. *English for the Computer. The SUSANNE Corpus and Analytic Scheme*. Clarendon, Oxford, 1995.

[17] A. Schiller, S. Teufel, and C. Stockert. Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical report, University of Stuttgart, University of Tübingen, 1999.

[18] T. Schmidt. Time Based Data Models and the Text Encoding Initiative's Guidelines for Transcription of Speech. *Working Papers in Multilingualism, Series B*, 62, 2005.

[19] T. Schmidt, C. Chiarcos, T. Lehmberg, G. Rehm, A. Witt, and E. Hinrichs. Avoiding Data Graveyards: From Heterogeneous Data Collected in Multiple Research Projects to Sustainable Linguistic Resources. In *Proc. of E-MELD Workshop on Dig. Lang. Doc.*, East Lansing, Michigan, June 2006.

[20] C. M. Sperberg-McQueen and L. Burnard, editors. *TEI P4: Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative Consortium, 2002.

[21] A. Witt, O. Schonefeld, G. Rehm, J. Khoo, and K. Evang. On the Lossless Transformation of Single-File, Multi-Layer Annotations into Multi-Rooted Trees. In B. T. Usdin, editor, *Proc. of Extreme Markup Languages*, Montréal, Canada, 2007.

[22] K. Wörner, A. Witt, G. Rehm, and S. Dipper. Modelling Linguistic Data Structures. In B. T. Usdin, editor, *Proc. of Extreme Markup Languages*, Montréal, Canada, 2006.