Computational Linguistics in the Netherlands Journal 3 (2013) 63-81 Submitted 06/2013; Published 12/2013

FoLiA: A practical XML format for linguistic annotation – a descriptive and comparative study

Maarten van Gompel PROYCON@ANAPROY.NL
Martin Reynaert REYNAERT@UVT.NL

Centre for Language Studies
Radboud University Nijmegen

Tilburg Center for Cognition and Communication
Tilburg University

Abstract

In this paper we present FoLiA, a Format for Linguistic Annotation, and conduct a comparative study with other annotation schemes, including the Linguistic Annotation Framework (LAF), the Text Encoding Initiative (TEI) and Text Corpus Format (TCF). An additional point of focus is the interoperability between FoLiA and metadata standards such as the Component MetaData Infrastructure (CMDI), as well as data category registries such as ISOcat. The aim of the paper is to present a clear image of the capabilities of FoLiA and how it relates to other formats. This should open discussion and aid users in their decision for a particular format.

FoLiA is a practically-oriented XML-based annotation format for the representation of language resources, explicitly supporting a wide variety of annotation types. It introduces a flexible and uniform paradigm and a representation independent of language or label set. It is designed to be highly expressive, generic, and formalised, whilst at the same time focussing on being as practical as possible to ease its adoption and implementation. The aspiration is to offer a generic format for storage, exchange, and machine-processing of linguistically annotated documents, preventing users as well as software tools from having to cope with a wide variety of different formats, which in the field regularly causes convertibility issues and proliferation of ad-hoc formats. FoLiA emerged from such a practical need in the context of Computational Linguistics in the Netherlands and Flanders. It has been successfully adopted by numerous projects within this community. FoLiA was developed in a bottom-up fashion, with special emphasis on software libraries and tools to handle it.

1. Introduction

Linguistic annotation is the enrichment of data with features that describe or analyse this data linguistically. The data often takes the form of written text, which may well be a transcription of an audio/video clip of human speech. A wide variety of such linguistic features can be thought of; examples are part-of-speech annotation, lemmatisation, co-reference relations, semantic roles, and so on. Within the field of computational linguistics we often see tools and data collections such as corpora use ad-hoc formats that are limited to a mere subset of these *annotation types*. When newer annotation types become available at a later point in time, the original format needs to be extended, which often occurs in a similar ad-hoc fashion and raises concerns of backward-compatibility. Moreover, such ad-hoc formats are often not formalised in any schema definition and thus hard to validate. This is all very well if this usage is limited to temporary data formats in a linguistic processing pipeline with specialised tools, but the practice becomes problematic when these formats are used as a data exchange or corpus storage format. This becomes even more of an issue if the ad-hoc formats are the result of the processing of a richer input format but fail to retain parts of the input format or the means to effectively refer to such original input format. Data loss is one of the most undesirable outcomes. A data exchange or corpus storage format requires a

more extensible and a more formalised format, one that does not reach its limits as different project partners or projects include different annotation types, and that never loses data with respect to the original input source.

In this paper we propose and describe FoLiA: Format for Linguistic Annotation, as such a format for data exchange and storage, as well as for processing by tools. The prime motivation for the development of FoLiA stems from the grievances we just sketched, and which we found all too common in practice. Nevertheless, ad-hoc formats are often used because they are simple and practical. If we intend to offer a viable alternative to this we must ensure to strongly focus on practical needs and ease of adoption and usage, as well as make available the necessary tools to work with the format.

We observe and acknowledge that numerous annotation formats exist in the field of Computational Linguistics and Linguistics in general. This raises the question why we have opted to create a new format rather than to use an existing one. FoLiA emerged in the context of the computational linguistics community in the Netherlands and Flanders. It was presented at the Computational Linguistics in the Netherlands (CLIN) conference in 2011 and 2012. A prevalence of the above mentioned ad-hoc formats was observed in many projects and tools, creating problems for data exchange and corpus storage. Development of FoLiA was initiated at an intersection of several projects (Stevin project SoNaR (Oostdijk et al. 2013), CLARIN-NL projects TICCLops and TTNWW, NWO project DutchSemCor)[1] and software projects (Ucto, Frog, Valkuil) that all faced such issues. At that point in time, the largest available Dutch language corpus, the D-Coi corpus, was using a custom XML-based format (Apperloo 2006) that could be considered a de facto standard. However, this format was severely limited in expressivity. FoLiA was built on the foundations laid by the D-Coi corpus format, extending it way beyond the limits this format imposed. Another widely established format in the same community is the Alpino XML format, used for the encoding of (Dutch) syntactic structures and dependency parses by the widely used Alpino parser (Bouma et al. 2000). This format is good at representing what it was designed for, but not expressive enough to act as a more generic format for linguistic annotation.

Existing formats that may compete with FoLiA feature-wise seem not to have been represented well in the aforementioned community, in part maybe simply due to a lack of awareness and lack of interest in formatting issues. Another reason, from a more development-oriented perspective, is the lack of NLP-tools and software libraries that offer direct support for existing formats and thus make it easy to work with them. FoLiA was consciously designed with this in mind, in a very bottom-up fashion, right alongside tools and libraries and within the scope of actual NLP projects. We see this as an important prerequisite to facilitate adoption and, by extension, de facto standardisation. This approach seems successful and noted. Odijk (2012) writes in a paper on recent developments in CLARIN-NL: "the corpus format in use in this project [the TTNWW project], FoLiA, appears to develop into a de facto standard for text corpora annotation for Dutch". We dedicate Section 5 to a more detailed discussion of the usage of FoLiA in current projects and lessons learned.

In Section 4 the differences and compatibility between FoLiA and the main alternatives available will be studied in-depth. For now, before going into an actual description of the FoLiA paradigm in Section 3, we will list several dimensions or criteria on which formats for linguistic annotation can be assessed in Section 2.

2. Criteria for assessment

We draw the dimensions on which we assess formats for linguistic annotation from Ide and Romary (2004). In this study a group of experts drafted requirements for a framework for linguistic annotation. We interpret these as dimensions or criteria along which the characteristics of annotation formats can be defined. Each dimension we paraphrase as a question. Subsequently, we give an

1. We refer to Section 5 for a discussion of FoLiA in practice.

indication of how FoLiA measures along each of these dimensions. This provides an initial overview of FoLiA and presents a foundation for further comparison in Section 4.

1. **Uniformity & Consistency** – To what extent is the format uniform; based upon the same building blocks rather than ad-hoc provisions? Is the format capable of expressing different levels of granularity? We would like to group this category with what Ide and Romary (2004) denote as *consistency*: are the same mechanisms used to indicate the same type of information?

   FoLiA proposes a uniform paradigm to represent linguistic data, this paradigm or data model consists of a finite set of building blocks that are used throughout the format. Section 3 will focus on a detailed description of this model.

2. **Expressivity** – Is the format expressive enough to represent a wide variety of linguistic annotation?

   Linguistic annotation types need to be explicitly supported in FoLiA, and a wide variety indeed is, following a uniform paradigm. FoLiA is a verbose format but tries to not force undesired verbosity on the users if the user is willing to compromise other dimensions such as formality.

3. **Openness** – Does the representation commit to a single linguistic theory, or is it open to multiple theories and label sets? Likewise, we add, is the format specialised for one or more languages or is it language independent?

   FoLiA does not commit to any single linguistic theory and is open to any label set or language. The definition of label sets is separate from the annotated documents. Section 3.3 will discuss this.

4. **Extensibility** – Is the format easily extensible?

   The openness dimension sketched above already allows for a fair amount of flexibility. In addition to this, higher-order annotations (see Section 3.5) in FoLiA allow for additional freedom by being able to freely associate extra features (key/value pairs) with any annotation, without concession to the formalised nature of the format. The main annotation types, however, are a core specialised part of the format and are defined centrally. This distinguishes FoLiA from more abstract formats. FoLiA remains in constant development and subsequent releases often add support for additional annotation types, fitted within the FoLiA paradigm and with full regard for maintaining backward-compatibility. Users who see the need for additional annotation types are always encouraged to contact the authors.

5. **Explicitness & Formality** – Is the annotation scheme explicit and unambiguous enough, not shifting the burden of interpretation to processing software? To this we add what Ide and Romary (2004) denote as *Semantic Adequacy*: do the representational structures and data categories have a proper formal semantics? Are they stored in a shared fashion? In this same spirit we further add: is there a proper schema definition of the format against which instances can be validated?

   FoLiA is a highly verbose format and explicit in its representations. There is a strict separation between the annotated document and the definition of the data categories in a separate FoLiA Set Definition format. This allows for high formality and deep validation. At the same time, this is not forced on users so more informal usage without set definitions is also possible. Shallow validation validates any FoLiA document against a RelaxNG schema (Clark 2001). Each FoLiA element has clearly defined semantics, defined in the documentation (van Gompel 2012b).

6. **Human readability** – To what extent are the representations human readable and editable?

   FoLiA XML files are human readable, though high verbosity may complicate matters. On the other hand, we strongly commit to usage of XML hierarchical notation and inline annotation (where possible), which makes it relatively easier compared to certain stand-off-only formats.

7. **Media independence** – Does the annotation format support different media? Or is it committed to one, such as written text?

   Though different media are supported and proposals are being written for more comprehensive speech annotation, written text is currently best represented.

8. **Incrementality** – Does the format support annotations of alternatives and partial/underspecified results?

   FoLiA supports the annotation of alternatives for all main annotation types. These alternatives can each be assigned confidence scores. FoLiA also supports explicit corrections, or mere suggestions for corrections, on annotations. Untokenised text, and text at different stages of processing can also be encoded. There is an important limitation in this dimension, however. FoLiA presupposes one authoritative tokenisation if linguistic annotations are added. Increased granularity within this tokenisation, through morphemes or phonemes, is always possible.

We add several criteria of our own to the requirements carefully drafted by Ide and Romary (2004). These were not addressed as such in their study, but are considered important nevertheless:

9. **Genericness vs Specialisation** – How generic is the format? Is it perhaps highly specialised for a certain kind of annotation? This is strongly correlated with *expressivity*.

   Our aim is to provide, within reasonable limits, a one-format-fits-all solution. The scope of FoLiA is broad rather than narrow and highly specialised.

10. **Parsability & Efficiency** – Can the format be parsed and processed efficiently? Does it make a big demand on system resources?

    We emphasize expressivity over efficiency. That is, efficiency may never come at the cost of expressivity. Nevertheless, we still distinguish ourselves from several other formats by the way the data is structured as a mix of inline and offset annotation. This is done to facilitate efficiency/parsability and also has a positive effect on human readability. FoLiA is designed also with streaming XML parsers in mind, which do not hold the entire document in memory. Keeping entire XML documents in memory, FoLiA documents included, is often very expensive.

11. **Locality** - Are annotations pertaining to the same sequence(s) in close proximity to one another in the representation, or are they spread out over the full representation? And is the representation spread over multiple files?

    Annotations can be grouped in various ways. In FoLiA we choose to keep information as local as possible, and within a single XML file.

12. **Accessibility** - Is the proposed format available freely (open access), both free of charge as well as free to extend, and are there tools and software libraries available for working with the format, released under open-source licenses?

This for us is an important dimension that is highly emphasized. Our aim is to deliver not just a theoretical framework, but a practical format with tools and libraries under open-source licenses.

13. **Abstraction** - How far does the format abstract from the linguistic concepts familiar in the field?

    Certain formats require high levels of abstraction whilst others are very close to the original source. The usage of XML already implies a higher level of abstraction. We keep the XML elements strongly correlated with meaningful concepts in the field. Annotation types and text-structural elements are explicitly defined and fixed by the format, rather than completely open to the user to define. In this regard, we choose a lower level of abstraction than certain other formats.

Formats in the field will rank differently along each of these dimensions. A precise objective quantification along these dimensions is neither possible nor necessary to illustrate and compare the properties of a format. Moreover, different objectives result in a different prioritisation of the dimensions. This will, in part, account for the variation in formats in existence.

3. Description

An extensive overview of all annotation types FoLiA offers is beyond the scope of this paper; for that we refer to the FoLiA Documentation (van Gompel 2012b) and the validation schema which is available in RelaxNG format (Clark 2001), the latter is ideally suited to validate your own FoLiA documents to make sure they adhere to the standard. In the current section our aim is to clarify the data model used by FoLiA, show the capabilities given this paradigm, and motivate our choices.

3.1 Annotation style

FoLiA is an XML-based format (Bray et al. 1998). XML is a common choice as it is a widely established and accepted standard. Choosing XML also implies that we have powerful technologies such as XPath, XSLT, and XQuery at our disposal for handling FoLiA documents. XML is inherently a hierarchical format which lends itself well for the encoding of text documents. This is proven most of all by the ubiquity and success of (x)HTML on the web. The OHCO thesis, first posited by DeRose et al. (1997) and revised in Renear et al. (1996), asserts that "text is an Ordered Hierarchy of Content Objects". Each structural element that makes up the text may hold deeper, more fine-grained, structures of the text. XML is well suited for modelling such data and FoLiA makes strong use of the hierarchical capabilities of XML. For document structure it fully commits to the OHCO thesis, and inherits most of this from its predecessor D-Coi. As far as possible FoLiA continues in this fashion for linguistic annotation, which implies an *"inline annotation"* style. We can distinguish four categories of annotation or representation in FoLiA, the first two of which are forms of inline annotation:

1. **Structure annotation** – Representation of document structure. This includes *words/tokens, divisions (i.e. chapters, sections, subsections), headers (titles of divisions), paragraphs, sentences, utterances, morphemes, phonemes, lists, list items, figures and their captions, extra linebreaks, gaps and events.*

2. **Token annotation** – Annotation applied to a single token or, by extension, to another structural element. This category encompasses the following annotation types: *part-of-speech tags, lemmatisation, lexical semantic sense annotation, subjectivity annotation, domain annotation.* All are annotations pertaining to a single structural element.

The choice for inline annotation for structure annotation is largely justified by the OHCO thesis and the aforementioned success of hierarchical representations of text on the web. For token annotation the choice for inline annotation maximises the use of the hierarchical expressivity of XML, which results in better *human readability* and *parsability*. The excerpt of FoLiA XML in Figure 1 illustrates a sentence, tokenised, annotated with Part-of-Speech tags and showing structure annotation as well as token annotation.

```
<s xml:id="example.p.1.s.2">
  <w xml:id="example.p.1.s.2.w.1">
    <t>He</t>
    <pos class="PRP" />
  </w>
  <w xml:id="example.p.1.s.2.w.2">
    <t>is</t>
    <pos class="VBZ" />
  </w>
  <w xml:id="example.p.1.s.2.w.3">
    <t>Mr.</t>
    <pos class="NNP" />
  </w>
  <w xml:id="example.p.1.s.2.w.4">
    <t>Smith</t>
    <pos class="NNP" />
  </w>
  <w xml:id="example.p.1.s.2.w.5">
    <t>.</t>
    <pos class="." />
  </w>
</s>
```

Figure 1: An excerpt of FoLiA XML showing a sentence, tokenised, with part-of-speech annotation. The excerpt illustrates the use of in-line annotation for structural elements and token annotation, and the use of identifiers.

FoLiA elements may carry a unique identifier by which they can be easily referenced, both internally and from external sources. Figure 1 illustrates this. The precise convention adopted for naming identifiers is left to the user.

The inline annotation style used in FoLiA's token annotation brings benefits but is not without its limits. In order to attain the high expressivity we desire, stand-off annotation cannot be avoided. In the title of their study, Renear et al. (1996) rightly posit that problems arise as soon as overlapping structures have to be represented. FoLiA's *Span Annotation* remedies this problem by resorting to a stand-off annotation style in separate annotation layers. Span annotation is used for annotation types that span multiple structural elements.

3. **Span annotation** – Annotation applied to a span of multiple tokens or other structural elements: *syntactic structure, named entities, dependency relations, co-reference relations, semantic roles, chunking/shallow parses.*

4. **Higher-order annotation** – Annotations on annotations. See Section 3.5.

Within the scope of a span annotation element, references are made to the structural elements over which the span applies, often word tokens, and refer to their unique identifiers. Span annotation elements of a certain type are grouped in an *annotation layer* corresponding to the annotation type; this layer is in turn embedded *inline* in the structural element that encompasses all spans. An

```
<s xml:id="example.p.1.s.2">
  <w xml:id="example.p.1.s.2.w.1">
    <t>He</t>
    <pos class="PRP" />
  </w>
  <w xml:id="example.p.1.s.2.w.2">
    <t>is</t>
    <pos class="VBZ" />
  </w>
  <w xml:id="example.p.1.s.2.w.3">
    <t>Mr.</t>
    <pos class="NNP" />
  </w>
  <w xml:id="example.p.1.s.2.w.4">
    <t>Smith</t>
    <pos class="NNP" />
  </w>
  <w xml:id="example.p.1.s.2.w.5">
    <t>.</t>
    <pos class="." />
  </w>
  <entities>
        <entity xml:id="entity.1" class="person">
            <wref id="example.p.1.s.2.w.3" />
            <wref id="example.p.1.s.2.w.4" />
        </entity>
  </entities>
</s>
```

Figure 2: An excerpt of FoLiA XML illustrating the use of span annotation, in this example: named entities.

example of this is shown in Figure 2. Span annotation is also geared towards nested, tree-like structures, such as syntax.

A deliberate choice was made to keep information together and as localised as possible. The annotated document encompasses a single XML file rather than multiple files for different annotation types. The annotation layers, using stand-off annotation, are embedded in an in-line fashion at a local level, as we have seen. This choice is motivated largely by parsability and to a lesser extent by human readability. Keeping information together and localised allows parsers to function in a streaming fashion and lowers memory requirements. When a parser retrieves for instance a sentence, it can be sure to have contained within it all relevant annotations that are limited to the scope of that sentence. Moreover, holding everything in a single file ensures the integrity of the document and its annotations is maintained as a whole and can be easily used in data exchange or storage, also with regard for versioning control. This focus on high locality distinguishes FoLiA from various other formats, as shall be discussed later in Section 4.

3.2 Text content

The textual content of a structural element is considered an annotation just as well, albeit with certain special properties. The `t` element represents the text content in FoLiA and is shown in Figures 1 and 2. Text content can be embedded on various levels of structure annotation. Any text content outside of the word/token level is by definition untokenised. Likewise, text content within the word/token level by definition is tokenised. Links between the two can be retained by using the optional *offset* attribute, pointing to the unicode *character* (as opposed to byte offset) where the text fragment is embedded in the higher level. These character offset references *by default* point

to the text pertaining to the structure element "one level up", where a level also corresponds to a structure annotation element. Figure 3 illustrates this.

The same figure also illustrates that a token is not the structure element of smallest granularity, there is also notation for representing morphemes and phonemes. These are embedded within an annotation layer in a word/token and can make use of the same offset attribute to relate their text to the full token.

```
<p xml:id="example.p.1">
 <t>He hits Mr. Smith. That came quite expected!</t>
 <s xml:id="example.p.1.s.1">
  <t offset="0">He hits Mr. Smith.</t>
  <w xml:id="example.p.1.s.1.w.1"><t offset="0">He</t></w>
  <w xml:id="example.p.1.s.1.w.2"><t offset="3">hits</t>
    <morphology>
        <morpheme class="lexical" function="lexical">
         <t offset="0">hit</t>
        </morpheme>
        <morpheme class="suffix" function="inflectional">
         <t offset="3">s</t>
        </morpheme>
    </morphology>
   </w>
  <w xml:id="example.p.1.s.1.w.3"><t offset="8">Mr.</t></w>
  <w xml:id="example.p.1.s.1.w.4" space="no"><t offset="10">Smith</t></w>
  <w xml:id="example.p.1.s.1.w.5"><t offset="15">.</t></w>
 </s>
  ..
</p>
```

Figure 3: Text content with references on multiple levels: paragraph, sentence and word. Also included is an example of morphological annotation.

3.3 Sets, classes and common attributes

Central to the FoLiA paradigm is the notion of *sets* and *classes*. A set represents a label set for a particular annotation type; the elements of this set are referred to as classes and represent the individual data categories. FoLiA does not predefine any sets or classes, these are left to the user to define, and in doing so we commit to a high level of openness. Set definitions are defined outside of the annotated FoLiA documents, using a different format: *FoLiA Set Definition format*[2].

It is the task of the FoLiA Set Definition format to define all possible classes in a set, or to declare a set as "open" if the classes do not form a closed vocabulary. Whereas the classes are short identifiers, the set definition format can map each of these to fully human-readable labels. A mapping may also be included to an external data category registry such as ISOcat (Kemps-Snijders et al. 2009) which may take over a lot of the role of the set definition format.

The set definition format may also pose constraints on how classes of *subsets* may be combined. The notion of subsets is explored in Section 3.5. The validity of a FoLiA document can be validated against a RelaxNG schema which checks whether the document correctly adheres to the proper FoLiA syntax; this we call "shallow validation". Set definition files allow us to take a step further and do "deep validation" as well, in which the actual classes are checked.

Figures 1 and 2 illustrate the use of classes in FoLiA. The actual Part-of-Speech tags are the classes and they form part of a set (in this case the Penn Treebank Tagset). In these excerpts, the

2. Alternative set definition formats may be supported as well in later versions of FoLiA.

sets are not made explicit. Sets can be referred to using the `set` XML attribute that any token annotation element or span annotation element may take. The value of this attribute is a URL where the set definition resides. Each annotation type with the used set must be *declared* in the metadata portion of the FoLiA document. If only one set is declared for a given annotation type, as is often the case, then the `set` attribute needs not be specified on each of the relevant annotation elements. These mandatory declarations will be further discussed in Section 3.4.

This paradigm is applied consistently to annotation elements of all types. The set points to a URL holding a file in FoLiA Set Definition format that declares all possible classes of the set. It may also declare a set as being open. Human readable labels can be associated with each class in a set and constraints can be posed on combinations of classes in subsets, see Section 3.5. Moreover, references to a data category registry such as ISOcat (Kemps-Snijders et al. 2009) can be made at this level as well.

The FoLiA paradigm furthermore states that annotation elements may always take certain common XML attributes; these are never mandatory:

- **annotator** – A name or ID of the person or system responsible for this annotation, i.e. the creator or editor.
- **annotatortype** – "auto" or "manual", indicating whether the annotator is an automated system or whether the annotation is the result of manual human labour.
- **confidence** – A value between 0 and 1 expressing the confidence placed in this annotation by the annotator.
- **n** – A string representing a numeral that indicates the position of this element in a certain order. Used in, for example, chapter numbers, section numbers, enumerations.

For structural elements in a speech context, additional common attributes are available:

- **src** – A link to a sound or video clip containing the speech.
- **begintime** — A timestamp indicating the begin time of this speech act.
- **endtime** — A timestamp indicating the end time of this speech act.

Figure 4 also shows that FoLiA has provisions for encoding alternatives: an alternative sense is encoded alongside the actually selected sense. This example illustrates alternatives for token annotation, yet a similar construct is available for span annotation.

3.4 Metadata

A FoLiA document always starts with a metadata specification. FoLiA introduces a simple native $field \rightarrow value$ map for metadata, leaving it up to the user to invent fields. However, more sophisticated metadata specifications such as CMDI (Broeder et al. 2011) can be used with FoLiA and are highly recommended. These are independent of FoLiA and can be either embedded in the FoLiA document, or be kept in an external file to which an explicit reference is made. The motivation is to keep FoLiA specialised on actual representation of annotations and leave metadata to more capable formats that are already being used in the field.

A mandatory part of the metadata that cannot be delegated to other metadata formats yet is the declaration of annotation types and sets. This enables users and parsers alike to know whether certain annotations are present in the document, without resorting to a full scan.

The common FoLiA attributes `annotator` and `annotatortype` can arguably be considered metadata as well, but they are so closely related to the annotations that their inclusion in FoLiA is justified in our view. In fact, standardised facilities to easily make explicit the annotators of an annotation are lacking in certain other formats, whilst they make a format more suited for use in annotation tools.

```
<w xml:id="example.p.2.s.1.w.2">
  <t>amazing</t>
  <pos set="http://someserver/claws"
    class="jj" annotator="Maarten van Gompel"
    annotatortype="manual" confidence="0.9"/>
  <lemma set="http://someserver/lemmas"
    class="amazing" annotator="C3PO"
    annotatortype="auto" />
  <sense set="http://someserver/wordnet3"
    class="amazing%3:00:00:surprising:00"/>
  <alt auth="no">
   <sense set="http://someserver/wordnet3"
     class="amazing%3:00:00:impressive:00"
     annotator="Martin Reynaert" annotatortype="manual"
     confidence="0.23" />
  </alt>
</w>
```

Figure 4: Excerpt of a word with token annotations in (fictitious) sets, with common attributes. Also shown is an example of an alternative.

3.5 Higher order annotation

The paradigm of sets and classes allows for further granularity by introducing *features* and *subsets*. A feature in FoLiA is a form of higher-order annotation: it is essentially an annotation on an annotation. Where token and span annotations take a set, features take a subset. This subset is defined within the scope of a set, and thus in the same set definition file. Like sets, subsets consist of classes.

An illustrative example of the usage of features and subsets can be made using part-of-speech annotation, as shown in Figure 5. The part-of-speech annotation element in this label set happens to be assigned a class that encodes all features in one string; this is a property of the label set and not of FoLiA, as FoLiA never prescribes classes. The feature notation in FoLiA in turn allows you to explicitly encode each part-of-speech feature.

```
<pos class="N(soort,ev,basis,zijd,stan)" head="N">
    <feat subset="ntype" class="soort" />
    <feat subset="number" class="ev" />
    <feat subset="degree" class="basis" />
    <feat subset="gender" class="zijd" />
    <feat subset="case" class="stan" />
</pos>
```

Figure 5: Example of a Part-of-Speech tag with features.

Figure 5 additionally demonstrates that annotation types may *predefine* certain subset names (but never their classes): for part-of-speech annotation, the subset "head" is predefined; this then allows shorter notation using an XML attribute instead of the `feat` element, but it is semantically identical within FoLiA. Likewise, Figure 3 included an example of morphology, which predefines the subset "function". There is no obligation on sets to actually use these predefined subsets and any subset can be invented by the user, fully committing to the idea of openness.

Other forms of higher-order annotation which we shall not address further are:

- *Alignments* – Links (DeRose et al. 2010) between arbitrary annotations or structure elements, also across FoLiA documents.

- *Descriptions* – Extra human-readable descriptions can be associated with any annotation.

- *Corrections* – FoLiA also implements extensive support for storing and tracking corrections or suggestions for correction.

4. Comparison

In this section we will provide a more in-depth comparison between FoLiA and other formats for linguistic annotation.

4.1 The Linguistic Annotation Framework (LAF, GrAF, MAF, SynAF)

The Linguistic Annotation Framework is developed by the ISO/TC37/SC4 subcommittee of the International Organization for Standardisation (ISO). This committee has developed "principles and methods for creating, coding, processing and managing language resources, such as written corpora, lexical corpora, speech corpora, dictionary compiling and classification schemes" (Ide and Romary 2004).

LAF cannot be seen as a format as such, but rather as a comprehensive meta-model or paradigm. The scope of the paradigm is wide, and the levels of abstraction, generalisation, uniformity and consistency are high. Various user formats or representations can in their own way make use of this data model. This is in line with the observation that different constraints are posed on formats by different uses. There is an explicit aim at maximum flexibility.

Ide and Romary (2004) state that the LAF data model is essentially a feature structure graph. An instantiation of this idea is found in the GrAF format (Ide and Suderman 2007), an actual XML-based format that can encode the full LAF data model. This format consists in essence of nodes and edges. GrAF is intended to act as the LAF pivot format for conversion between user-defined formats.

The LAF paradigm is stand-off. It first of all lays a base segmentation (tokenisation) by referencing to offsets in the original data document, a separate file. Unlike FoLiA, multiple segmentations are possible. Annotations (graph nodes) can then refer (using graph edges) to this base segmentation. Annotation can also be drawn over other annotations, a more abstract and generic version of what higher-order annotation in FoLiA is.

LAF-compatible representation of tokenisation and tagging is covered by the Morpho-syntactic Annotation Framework (MAF) (de la Clergerie and Clément 2005). The Syntactic Annotation Framework (SynAF) focusses on the representation of syntactic structures (Declerck 2008), and part of its focus is a definition of data categories, which is outside the scope of FoLiA. SynAF is still a meta-model and does not commit to a particular XML serialisation. However, SynAF is inspired by Tiger XML (König and Lezius 2000), the successor of which, Tiger2 (Bosch et al. 2012), is a full SynAF-compatible serialisation format.

LAF and GrAF offer great flexibility but are of less direct practical use for end-users seeking a hands-on format. With regard to this Bański and Przepiórkowski (2010) remark: "A tendency may be observed of increasing abstractness and generality of proposed standards, esp., SynAF and LAF. This leads to their greater formal elegance, at the cost of their actual usefulness". We concur and conclude that this is not a format end-users can just pick up and use with readily available tools, despite the availability of some libraries for GraF. Due to the level of abstraction, the learning curve is fairly steep and leaves a lot to the user to define. Moreover, accessibility is greatly hindered by the fact that the full and latest specification of LAF, MAF and SynAF are not freely accessible, but only after purchase from ISO. This is in sharp contrast with the emphasis on open access and open source in FoLiA.

The question that arises in this context is to what extent FoLiA is compatible with LAF, i.e. to what extent FoLiA can be seen as a format adhering to the LAF data model and whether it can be converted to and from the pivot format proposed by LAF.

The fact that FoLiA in part commits to an inline annotation style is not an obstacle for conversion to any fully stand-off representation desired by LAF; more important is the fact that the notation employed by FoLiA is consistent and unambiguous. The reverse conversion, from LAF to FoLiA, will be somewhat more constrained. FoLiA will not be able to encode the full extent of LAF; most notably it will not handle annotations on multiple mutually incompatible tokenisations of the source text, unless one produces various and mutually divergent FoLiA versions of the same document. The high generalisation and flexibility in LAF may not always translate to FoLiA, as FoLiA is by design more specific, less abstract, and facilities for annotation types have to be explicitly present. Nevertheless, we think a large subset can still be covered in a variety of applications, as many features are present in FoLiA and many annotation types are supported. A large extent of the higher-order graph-based annotations in LAF could also be represented in FoLiA through *alignments* and *subsets/features*. Furthermore, FoLiA commits to a high level of openness and, like LAF, takes standards such as ISOcat (Kemps-Snijders et al. 2009) into consideration.

4.2 The Text Encoding Initiative (TEI P5)

The Text Encoding Initiative has established itself as a de facto standard for the encoding of a wide variety of text documents and is frequently used in the Digital Humanities and beyond. The project represents a large community-based effort which produces *guidelines* for text encoding (Burnard and Bauman 2007). It defines a vast amount of tags for the encoding of all kinds of texts, in which different views on a text can be taken for different uses and applications. Like FoLiA, TEI is used for data storage, exchange, and processing. Furthermore, TEI can be considered a direct ancestor leading to LAF and its children: "the current standards that have been or are being established by ISO TC 37/SC4 committee ..., known together as the LAF (Linguistic Annotation Framework) family of standards, ... descend in part from an early application of the TEI." (Bański and Przepiórkowski 2010).

FoLiA's predecessor D-Coi states to be loosely based on a subset of TEI (Apperloo 2006). FoLiA inherits this, which results in certain similarities in structure annotation. Moreover, like FoLiA, TEI largely commits to an inline annotation style, but also has support for stand-off notation where necessary. When it comes to the encoding of textual structure, TEI clearly offers more expressivity than FoLiA.

TEI also provides facilities for linguistic annotation. Certain linguistic properties such as lemmas can be represented as XML attributes. Morphemes are also well supported in TEI. A further set of generic analysis elements is available, supporting spans too. Generic elements for *feature structures* are also available. These feature structures follow the same de jure standard as the Morpho-syntactic Annotation Framework. Stührenberg (2012) defines feature structures as general-purpose data structures consisting of a named feature and its value(s). Complex feature structures contain a group of individual features allowing for a representation of various kinds of information. FoLiA's token and span annotation elements, optionally enriched with FoLiA's *features*, are essentially (instantiations of) feature structures; at a higher level of specificity and lower level of abstraction, each annotation-type with proper semantics and following the same uniform paradigm.

Przepiórkowski (2009) argues for TEI P5 as a suitable format for the encoding of treebanks. He does observe that, at the time of the paper, the impact of the P5 version for linguistically annotated corpora is limited: "Ideas useful for linguistically annotated corpora are scattered over the 1350-odd pages of the Guidelines, and usually there is more than one way of representing any given annotation, so designing a coherent and constrained TEI-conformant schema for linguistic corpora is a daunting task". It is unclear to what extent TEI has been more enthusiastically adopted for

linguistic annotation in recent years in spite of this. Still, this is in sharp contrast to FoLiA's higher specificity.

4.3 The D-SPIN Text Corpus Format (TCF 0.4)

The D-SPIN text corpus format (Heid et al. 2010) was developed in the context of the Deutsche Sprachressourcen-Infrastruktur (D-SPIN) project, which is the German contribution to the European Common Language Resources and Technology Infrastructure (CLARIN) project. In comparison, FoLiA is used by several projects in the Dutch version of CLARIN, i.e. CLARIN-NL (Odijk 2010).

The TCF format aims at a representation that supports interoperability between different web services in a corpus processing pipeline. It is strongly linked to the WebLicht software which offers a webservice infrastructure and employs TCF as interchange format (Hinrichs et al. 2010).

Of all the formats compared in this section, the TCF format is most similar to FoLiA with regard to its intended use and audience. The levels of abstraction in TCF and FoLiA are very similar, and both share a practical focus.

TCF is inspired by the Linguistic Annotation Framework (Ide and Romary 2004), and unlike FoLiA, the TCF format is primarily a stand-off format. The corpus text, its tokens, and its annotations are all encoded in different sections. This implies that TCF differs from FoLiA in the dimension of *locality* and that *parsability* of TCF may be somewhat more complicated than FoLiA as more dereference operations are needed.

Like FoLiA, TCF assumes one basic tokenisation underlying the data. TCF also employs globally unique identifiers, which are extensively used due to the stand-off nature.

TCF differs from FoLiA in *expressivity*, being less expressive than FoLiA as FoLiA offers more annotation types. This makes FoLiA generally more verbose. Annotation types in FoLiA are also more uniform than in TCF, as TCF does not introduce as strict a paradigm as FoLiA does with its notion of sets, subsets, classes and common attributes[3]. All in all, FoLiA offers a higher degree of formality, through set definitions and deep validation.

TCF is developed with more regard for the compatibility with the Linguistic Annotation Framework (Ide and Romary 2004) and sees more effort towards providing converters to this end than FoLiA. For FoLiA this remains open for future work, but such conversions from and to LAF standards should in principle be no less feasible than for TCF.

TCF has some tools and libraries available, including validators. There is a Java software library as well as a Java pilot desktop application (TIEWER) for viewing and limited editing. An online web application for viewing TCF is also available.

4.4 Other formats

We compared against two major formats or frameworks, and one more practical format which is more on par with FoLiA. In this last section of our comparison, we will briefly sketch some comparisons with other common formats in the field.

The XCES (XML Corpus Encoding Standard) format is an older format that is derived from its non-XML predecessor CES (Ide et al. 2000) and which aims to follow the TEI guidelines. The format is fairly old, even in its second version, and currently seems not well maintained; the last update of the xces.org site dates from June 2008. It is criticized for its lack of documentation (Bański and Przepiórkowski 2010). Przepiórkowski (2009) anticipates this and adds that XCES lacks specific recommendations for any linguistic levels, resorting instead to general feature structure mechanisms. FoLiA in contrast has explicit support for various annotation types.

PAULA (Potsdamer Austauschformat Linguistischer Annotationen) is an XML format geared towards data exchange, used and developed within the SFB 632 project in Germany[4]. The format

3. Features such as support for encoding who were the annotators do not seem to be available in TCF.
4. `http://www.sfb632.uni-potsdam.de/`

is fairly well documented and comes with DTD schemas for validation. It uses a strict stand-off style in which different layers of linguistic annotation are encoded in different files. A primary data text file forms the lowest level of resource representation (Zeldes et al. 2013), and is always mandatory. A tokenisation file posits a segmentation, by verbose references, *marks* in PAULA, to the data file. Further marks or the more powerful and abstract *structs* can be used akin to span annotation in FoLiA. Stand-off feature structures hold the actual linguistic annotations. PAULA is more generic and less specific than FoLiA. The stand-off nature makes it generally more verbose, less human-readable and less easily parsable than FoLiA. It commits to a high level of openness, but in this regard it lacks in formality as it does not seem to provide facilities to define data categories or relate them to standards such as ISOcat. Unlike FoLiA, PAULA does not specify any real structure notation.

PAULA is inspired on early drafts of LAF and uses the same graph paradigm, but is of more direct practical use. An open-source search and visualisation tool, ANNIS2[5], is available with support for PAULA.

5. Practice

FoLiA emerged at the intersection of different projects, several still ongoing, within the Computational Linguistics community in the Netherlands and Flanders. As part of the general overview of FoLiA in this paper, we in this section present a short overview of projects and tools in which FoLiA is used.

Especially notable is the SoNaR corpus (Oostdijk et al. 2013), the 500 million word reference corpus for contemporary written Dutch, which is delivered in FoLiA format. Other corpora in which FoLiA is used are DutchSemCor, a corpus annotated with lexical semantic senses (Görog and Vossen 2010); Basilex[6], a corpus consisting of Dutch texts young children would typically be exposed to; VU-DNC, a 2 million word diachronic corpus for Dutch offering both sentiment annotations and a gold standard for OCR post-correction. Moreover, several projects within CLARIN-NL's TTNWW project are also employing FoLiA for their representational needs. NWO project Political Mashup[7] embeds FoLiA in its own XML format and has extended the format with elements for "semanticising" the texts of the Dutch Acts of Parliament[8]. In view of the focus on Dutch in all these projects, it has to be stressed again that FoLiA in fact is fully language independent. This is exemplified by the various third-party projects at research group LT3[9], Ghent University, which have adopted the format for the projects AMiCA, PARIS and SubTLe [10]. In so far as these projects are collaborations between the various Flemish universities and research groups, FoLiA seems set to find strong footing over the full Dutch language area comprising both the Netherlands and Flanders.

One of the emphasized assets of FoLiA is the availability of tools to work with the format; we believe this is a key aspect for projects considering adoption of our format. Other factors are the fact it is actively supported, used, and developed and the familiarity with its predecessor format D-Coi. There are currently two complete FoLiA libraries available for programmers, a Python library and a $C++$ library[11]. Furthermore, a suite called *"FoLiA tools"*[12] is available; it contains several converters to other formats (including plain text, CSV and HTML for visualisation), a validator, a tool for merging multiple FoLiA documents and a command-line tool for limited querying/searching

5. `http://www.sfb632.uni-potsdam.de/d1/annis/`
6. `http://www.basilex.nl`
7. `http://politicalmashup.nl`
8. E.g. Dutch parliamentary proceedings 1930-2012, semanticised: `https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:51848`
9. `http://lt3.hogent.be`
10. `http://lt3.hogent.be/en/projects/`
11. Both libraries can be obtained from `http://proycon.github.io/folia`
12. `http://pypi.python.org/pypi/FoLiA-tools`

of FoLiA documents. The visualisation stylesheet is popular and has already been adapted by several third party projects such as NederLab, Basilex, and Political Mashup to suit their specific needs.

Some further notable open-source tools[13] that use FoLiA are:

- Ucto[14] - A multilingual rule-based tokeniser (van Gompel et al. 2012). This tool is a good stepping stone for setting up an initial FoLiA document from scratch, it allows for tokenising plain-text input.
- Frog[15] - An integration of various memory-based NLP modules developed for Dutch. Allows for plain-text Dutch input to be converted into richly annotated FoLiA documents. Uses FoLiA as its internal format.
- TICCL - Text-Induced Corpus Clean-up. A system geared at fully automatic lexical normalization and correction of typographical and Optical Character Recognition (OCR) misrecognition errors in possibly very large corpora (Reynaert 2010).
- CLAM[16] - CLAM allows the quick building of fully fledged RESTful webservices by wrapping around any command-line tool (van Gompel 2012a). Though CLAM can be used with any format, it is very suitable for use with FoLiA as format for data exchange between webservices. This setup is found for instance in CLARIN-NL's and CLARIN Flander's joint TTNWW project.
- Valkuil[17] - A spelling corrector for Dutch.
- Fowlt[18] - A spelling corrector for English.
- FoLiA-stats[19] - Frequency list generation and n-gram extraction on FoLiA documents, works on word forms, lemmas and POS-tags.
- BlackLab[20] A corpus retrieval engine built on top of Apache Lucene. It allows fast, complex searches with accurate hit highlighting on large, tagged and annotated, bodies of text. Has support for both FoLiA as well as TEI. It will form the back-end for OpenSonaR.
- Brat[21] - Brat Rapid Annotation Tool. This web-based tool for structured text annotation is being extended at LT3, Ghent University, with support for importing, visualizing and exporting FoLiA information.

At the moment, off-the-shelf annotation environments are not yet available for FoLiA and remain much desired future work; promising work to this extent is being performed in the context of the Brat Rapid Annotation Tool. Corpus search environments that also include FoLiA support are under development in the newly started projects NederLab[22] and OpenSoNaR[23].

From the usage of FoLiA in practice we can already draw some lessons. FoLiA and XML are by nature very verbose formats, which results in sizable XML files. This makes a strong demand on

13. We list the tools in chronological order of their development and/or adaptation to FoLiA. The authors were not personally involved in the development of the last two listed.
14. `http://ilk.uvt.nl/ucto`
15. `http://ilk.uvt.nl/frog`
16. `http://proycon.github.io/clam`
17. `http://www.valkuil.net`
18. `http://www.fowlt.net`
19. `http://tilc.uvt.nl`
20. `https://github.com/INL/BlackLab`
21. `http://brat.nlplab.org`
22. `http://www.nederlab.nl/`
23. CLARIN-NL Call 4 project which started October 2013 in which the Stevin SoNaR corpus is to be made available online with various user interfaces for users with different research interests.

disk space as well as on memory if a document is to be fully loaded in memory. The former problem can be remedied by compression, which the FoLiA libraries support natively[24]. The latter problem can only be remedied by using streaming parsers; the principle of locality ensures that such parsers can be suitably used on FoLiA, which cannot be said for most other formats.

Another cautionary lesson we can draw is that third party users are often quickly inclined to either create ad-hoc extensions to the format or embed a subset of FoLiA within different formats. Especially the former is problematic and discouraged as FoLiA is a format high in specificity. Flexibility is offered in the form of the set/class paradigm and corresponding set definition files, features in subsets offer even more flexibility. If annotation categories need to be added, then this has to be a more centralised development effort in FoLiA as a whole. This proves not always sufficiently clear, and it is the task of the authors and developers to make this clearer.

Embedding FoLiA in other formats, or even other formats in FoLiA, may be a valid tactic on the strict condition that properly distinguishable XML namespaces are used. However, in the former case, the resulting documents can not be considered FoLiA documents as-such and cannot be processed by the FoLiA validator and other FoLiA tools; specialised parsers will be required. Moreover, special care is in order not to lose the annotation declarations required by FoLiA.

Another issue is the fact that the limits of any XML-based format are quickly reached in applications of real-time corpus querying. Real time iteration over millions of XML documents is far too costly in terms of processing time. For such applications, smart indexing and storage in some sort of database system is indispensable. The aforementioned Nederlab and OpenSoNaR projects aim for this.

6. Conclusion & future work

There is plenty to choose from for anybody deciding on a format for linguistic annotation. However, the abundance of choice does not necessarily make things easier and it greatly depends on the particular goals and requirements of the project.

Our aim in this paper was twofold. Firstly, we have provided a descriptive overview of the Format for Linguistic Annotation (FoLiA). Secondly, we have conducted a comparative study with other formats.

FoLiA distinguishes itself from most other formats by choosing to represent all annotations in a single XML file, following an inline annotation style where possible, and embedding stand-off annotation layers locally. It furthermore introduces a strict notion of sets and classes, and applies this consistently across all annotation types. This proves to be a powerful paradigm.

Comparisons were drawn with the Linguistic Annotation Framework and its children, the de jure standard for linguistic annotation. LAF offers a far higher level of abstraction and genericness than FoLiA. Undoubtedly more thought and debate has gone into establishing these theoretical foundations than is the case in the development of FoLiA, which follows a completely reversed bottom-up approach. FoLiA however aims at a far more practical level and is more easily accessible than LAF, which makes it far more easy to adopt in software tools.

Similar conclusions can be drawn from the comparison with TEI. FoLiA offers higher specificity for linguistic annotation[25] and is again at a more practical level when off-the-shelf solutions are needed.

On a practical level, TCF and PAULA are most comparable with FoLiA. One thing that sets FoLiA apart from all these formats is its commitment to locality, by following a largely in-line and single-document setup. This eases parsability and human readability. A second distinguishing characteristic of FoLiA is its higher specificity; each annotation type is an explicit part of the format

24. GZip and BZip2 compression is supported, achieving very high compression factors.
25. TEI though offers higher specificity and expressivity for structure annotation.

and carries proper semantics. This implies a slightly lower level of openness and genericness. The class/set and common attributes paradigm balances this and ensures the necessary flexibility.

Future work on FoLiA focusses on one hand on increased expressivity of the format, by adding specifications for more annotation types where current facilities prove insufficient; and on the other hand on development of further tools and libraries to work with the format. Current developments are underway for better representation for linguistic annotation of speech, and for reducing the currently present bias towards written text. Further focus is on finalising the more formal aspects of FoLiA; the set definitions and resulting ability for deep validation. These are not yet embraced in current practice, but are necessary for formal closure.

In line with the development of further tools is also the development of converters to other formats. Especially increased connectivity with the Linguistic Annotation Framework may be a notable point for future work.

Acknowledgments

Maarten van Gompel acknowledges CLARIN-NL for funding FoLiA support and development within the scope of various projects, starting with the TTNWW project. Martin Reynaert acknowledges support from NWO in projects Political Mashup and Nederlab. FoLiA and TEI interoperability is being extended with funding from CLARIN-NL in project @PhilosTEI (CLARIN-NL-12-006), while FoLiA support in corpus search environments is further being developed in OpenSoNaR (CLARIN-NL-12-013). We also acknowledge Ko van der Sloot as key developer for FoLiA, contributing greatly to its flourishing, and Prof. Antal van den Bosch for his continuing support for FoLiA. Last, we thank numerous partners in the Dutch-Flemish NLP community for thinking with us about the format and for enthusiastically adopting it.

References

Apperloo, Wilko (2006), XML basisformaat D-Coi: Voorstel XML formaat presentational markup, *Technical report*, Polderland Language and Speech Technology.

Bański, Piotr and Adam Przepiórkowski (2010), TEI P5 as a text encoding standard for multilevel corpus annotation, *Digital Humanities 2012 Conference Abstracts*, pp. 98–100.

Bosch, Sonja, Key-Sun Choi, Eric Villemonte de la Clergerie, Alex Chengyu Fang, Gertrud Faass, Kiyong Lee, Antonio Pareja-Lora, Laurent Romary, Andreas Witt, Amir Zeldes, and Florian Zipser (2012), <tiger2/> as a standardized serialisation for ISO 24615 - SynAF, *TLT11 - 11th International Workshop on Treebanks and Linguistic Theories - 2012*, Ediçoes Colibri, Lisbon, Portugal, pp. 37–60.

Bouma, Gosse, Gertjan van Noord, and Rob Malouf (2000), Alpino: Wide-coverage computational analysis of Dutch, *Eleventh annual meeting of Computational Linguistics in the Netherlands*, Vol. 37 of *Language and Computers - Studies in Practical Linguistics*, Rodopi, pp. 45–59.

Bray, Tim, Jean Paoli, and C.M. Sperberg-McQueen (1998), Extensible Markup Language (XML) 1.0, *Technical report*, W3C. http://www.w3.org/TR/1998/REC-xml-19980210.

Broeder, Daan, Oliver Schonefeld, Thorsten Trippel, Dieter Van Uytvanck, and Andreas Witt (2011), A pragmatic approach to XML interoperability – the Component Metadata Infrastructure (CMDI), *Balisage: The Markup Conference 2011*, Vol. 7.

Burnard, Lou and Syd Bauman, editors (2007), *TEI P5: Guidelines for Electronic Text Encoding and Interchange*, Text Encoding Initiative Consortium, chapter A Gentle Introduction to XML.

Clark, James (2001), RELAX NG specification. Organization for the Advancement of Structured Information Standards (OASIS), Committee Specification.

de la Clergerie, Eric and Lionel Clément (2005), MAF: a Morphosyntactic Annotation Framework, *Actes de LTC* pp. 90–94.

Declerck, Thierry (2008), A framework for standardized syntactic annotation, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008*, European Language Resources Association, pp. 3025–3028.

DeRose, Steven J., David G. Durand, Elli Mylonas, and Allen H. Renear (1997), What is text, really?, *SIGDOC Asterisk J. Comput. Doc.* **21** (3), pp. 1–24, ACM, New York, NY, USA.

DeRose, Steven J., Eve Maler, David Orchard, and Norman Walsh (2010), XML linking language (XLink) version 1.1, *Technical report.* http://www.w3.org/TR/2010/REC-xlink11-20100506/.

Görog, Attila and Piek Vossen (2010), Computer assisted semantic annotation in the DutchSemCor project, *Proceedings of the Seventh International Conference on Language Resoruces and Evaluation, LREC 2010*, European Language Resources Association, pp. 1220–1226.

Heid, Ulrich, Helmut Schmid, Kerstin Eckart, and Erhard Hinrichs (2010), A corpus representation format for linguistic web services: The D-SPIN Text Corpus Format and its relationship with ISO standards, *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*, European Language Resources Association, pp. 494–499.

Hinrichs, Marie, Thomas Zastrow, and Erhard W. Hinrichs (2010), WebLicht: Web-based LRT services in a distributed eScience infrastructure, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010*, European Language Resources Association, pp. 489–493.

Ide, Nancy and Keith Suderman (2007), GrAF: a graph-based format for linguistic annotations, *Proceedings of the Linguistic Annotation Workshop*, LAW '07, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–8.

Ide, Nancy and Laurent Romary (2004), International standard for a linguistic annotation framework, *Natural Language Engineering* **10** (3-4), pp. 211–225.

Ide, Nancy, Patrice Bonhomme, and Laurent Romary (2000), XCES: An XML-based encoding standard for linguistic corpora, *Proceedings of the Second International Language Resources and Evaluation Conference, LREC 2000*, European Language Resources Association, pp. 825–830.

Kemps-Snijders, Marc, Menzo Windhouwer, Peter Wittenburg, and Sue Ellen Wright (2009), ISOcat: remodelling metadata for language resources, *IJMSO* **4** (4), pp. 261–276.

König, Esther and Wolfgang Lezius (2000), A description language for syntactically annotated corpora, *Proceedings of the 18th Conference on Computational Linguistics, COLING '00*, Vol. 2, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1056–1060.

Odijk, Jan (2010), The CLARIN-NL project, *Proceedings of the Seventh International Conference on Language Resources and Evaluation, LREC 2010*, European Language Resources Association, pp. 48–53.

Odijk, Jan (2012), Recent developments in CLARIN-NL, *Proceedings of the Eight International Conference on Language Resources and Evaluation, LREC 2012*, European Language Resources Association, pp. 1055–1060.

Oostdijk, Nelleke, Martin Reynaert, Véronique Hoste, and Ineke Schuurman (2013), The construction of a 500-million-word reference corpus of contemporary written Dutch, *Essential Speech and Language Technology for Dutch: Results by the STEVIN-programme*, Springer Verlag, chapter 13.

Przepiórkowski, Adam (2009), TEI P5 as an XML standard for treebank encoding, *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories, TLT 8*, pp. 149–160.

Renear, Allen, Elli Mylonas, and David Durand (1996), Refining our notion of what text really is: The problem of overlapping hierarchies, *in* Ide, Nancy and Susan Hockey, editors, *Research in Humanities Computing*, Vol. 4, Clarendon Press, pp. 263–280.

Reynaert, Martin (2010), Character confusion versus focus word-based correction of spelling and OCR variants in corpora, *International Journal on Document Analysis and Recognition* **14**, pp. 173–187, Springer Berlin / Heidelberg.

Stührenberg, Maik (2012), The TEI and current standards for structuring linguistic data: an overview, *Journal of the Text Encoding Initiative* (3), pp. 1 – 14.

van Gompel, Maarten (2012a), CLAM: Computational Linguistics Application Mediator. Documentation, *Technical report*, Tilburg Centre for Cognition and Communication, Tilburg University and Radboud Centre for Language Studies, Radboud University Nijmegen. http://ilk.uvt.nl/downloads/pub/papers/ilk.1202.pdf.

van Gompel, Maarten (2012b), FoLiA: Format for Linguistic Annotation. Documentation, *Technical report*, Tilburg Centre for Cognition and Communication, Tilburg University and Radboud Centre for Language Studies, Radboud University Nijmegen. http://ilk.uvt.nl/downloads/pub/papers/ilk.1203.pdf.

van Gompel, Maarten, Ko van der Sloot, and Antal van den Bosch (2012), Ucto: Unicode Tokeniser. Reference Guide, *Technical report*, Tilburg Centre for Cognition and Communication, Tilburg University and Radboud Centre for Language Studies, Radboud University Nijmegen. http://ilk.uvt.nl/downloads/pub/papers/ilk.1205.pdf.

Zeldes, Amir, Florian Zipser, and Arne Neumann (2013), PAULA XML Documentation, *Rapport de recherche*. http://hal.inria.fr/hal-00783716.