Language Resources and Evaluation manuscript No.
(will be inserted by the editor)

Xigt: Extensible Interlinear Glossed Text for Natural Language Processing

Michael Wayne Goodman · Joshua Crowgey · Fei Xia · Emily M. Bender

Received: date / Accepted: date

Abstract This paper presents Xigt, an extensible storage format for interlinear glossed text (IGT). We review design desiderata for such a format based on our own use cases as well as general best practices, and then explore existing representations of IGT through the lens of those desiderata. We give an overview of the data model and XML serialization of Xigt, and then describe its application to the use case of representing a large, noisy, heterogeneous set of IGT.

Keywords Interlinear glossed text (IGT) · Annotation · Storage format

1 Introduction

In this paper, we propose a new model for representing a distinctive datatype of linguistics, Interlinear Glossed Text (IGT), and its extension *enriched* IGT which consists of the original IGT plus additional layers of annotation (e.g. word alignment and parse trees). IGT is a datatype that emerged from linguistic research as a way of compactly displaying a range of annotations which are of interest to linguists working in a variety of subfields, including structural subfields (most notably syntax, but also others), typology, and descriptive and documentary linguistics. As a display format, IGT serves to make linguis-

This material is based upon work supported by the National Science Foundation under Grant No. BCS-1160274 and BCS-0748919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

University of Washington
Dept. of Linguistics
Box 352425
Seattle WA 98195-2425
E-mail: {goodmami,jcrowgey,fxia,ebender}@uw.edu

tic examples more comprehensible by pairing source language strings[1] with translations into a language of wider communication and word-by-word (often morpheme-by-morpheme) annotations of the source language string. This helps other linguists reading the discussion of the example understand how the language under study is expressing the meaning indicated in the translation line. A simple example from Shona (adapted from Toews (2009)) is shown in (1):

(1)	Ndakanga	ndakatenga	muchero
	Ndi-aka-nga	ndi-aka-teng-a	mu-chero
	SBJ.1SG-RP-AUX	SBJ.1SG-RP-buy-FV	CL3-fruit
	‘I had bought fruit.’ [sna]		

We will refer to the first line of this example as the source language line, the second as the morpheme-segmented line, the third as the gloss line, and the fourth as the translation.[2]

Recent years have seen much interest in the standardization of storage formats (in contrast to display formats) for IGT. This interest has been driven in large part by the needs and goals of the documentary linguistics community. In particular, the representations proposed so far have been developed in the context of either tools supporting the production of large collections of IGT (such as are produced in many documentary linguistics projects) or the long-term preservation of such collections. Our focus, somewhat in contrast, is on the reuse of IGT through automated processing and accommodating additional layers of annotation in the enriched IGT. §2 below provides some further background on IGT as a datatype and on enriched IGT. In §3, we describe our use cases and the desiderata they motivate regarding the design of a representation format for IGT. In §4, we review existing IGT representation formats through the lens of these desiderata. We then present our new format, Xigt, in §5 and discuss a case study of importing a large, heterogeneous collection of IGT to Xigt in §6. Finally, §7 concludes with a discussion of future work.

2 Interlinear Glossed Text and Enriched IGT

In order to support the following discussion, we draw a distinction between the *display formats* and *storage formats* of IGT. A display format specifies conventions of lay-out on a (virtual) page for human consumption. The IGT example in (1) conforms to a common display format for IGT, where the order of lines and the white space between items serve as cues to human readers as to the import of each symbol in the representation. A storage format is meant for machine consumption and typically uses devices other than lay-out to make the role of each part of the IGT explicit. The representation we propose

[1] Languages that have non-Roman scripts will frequently be transliterated, but sometimes IGT will include both the original orthography and the transliteration as separate tiers.

[2] It is in fact more typical to find three-line IGT in linguistics papers, with only one of either the source or morpheme-segmented lines.

here, Xigt, is intended as a storage format. In our use cases (see §3 below) we translate IGT from other storage formats (those of tools presently used by documentary linguists developing IGT collections) or in fact display formats (the IGT collected from PDFs by the ODIN project (Lewis and Xia, 2010)) to Xigt for the purposes of further processing.

From a formal point of view, IGT involves aligned tiers of hierarchically nested structure: The sentence consists of words, and words of morphemes (and in some cases clitics). When the source and morpheme-segmented lines are both present, there is an implicit alignment between them. Likewise, there is an implicit alignment between the gloss line and the source and morpheme-segmented lines.

The de facto standard for the content of IGT is the Leipzig Glossing Rules (LGR; Bickel et al, 2008), which specify conventions on the representation of morpheme boundaries in the morpheme-segmented and gloss lines as well as the representation of grammatical information ('grams') in the gloss line. A complete and well-annotated IGT instance contains information about aspects of the string not directly under discussion. Therefore, the Leipzig Glossing Rules serve to improve both scientific accountability and productivity. For the former, LGR-compliance allows readers and reviewers to potentially notice correlations among linguistic facts and suggest possible alternative analyses, even if they do not know the language in question. For the latter, standardized representations make examples more amenable to reuse by facilitating comparison across descriptions of the same language and across languages.

As it is used as a display format, IGT generally contains only the information discussed above. The most typical departures from the format illustrated in (1) are the addition of brackets to show partial constituent structure and/or the incorporation of symbols displaying the putative position of empty elements. If IGT data are to be used for further automated processing, however, enriching the representation can increase its value for NLP tasks (Xia and Lewis, 2009; Georgi et al, 2012). In particular, enriched IGT can include both explicit representations of the implicit alignments, as well as further information gleaned by parsing the translation line (Lewis and Xia, 2008). The desiderata explored in the next section are meant to apply to both 'standard' and enriched IGT.

3 Motivation/Desiderata

The Xigt format has been developed in the context of the AGGREGATION project,[3] a project whose goal is to combine two sources of linguistic knowledge to create precision implemented grammars. The knowledge sources are the LinGO Grammar Matrix (Bender et al, 2002, 2010), a language-independent resource which maps relatively simple linguistic descriptions to working grammar fragments, on the one hand, and IGT on the other. We aim to infer these

[3] `http://depts.washington.edu/uwcl/aggregation/`

kinds of linguistic descriptions by applying and extending the methodology of Lewis and Xia (2008), who glean information about the structure of the source languages by parsing the translation line and projecting the information through the gloss line to the source line.

The purpose of creating the implemented grammars is to assist in the development of further documentary linguistic resources, including such ambitious goals as creating grammar-derived treebanks (in the style of Oepen et al 2004, see also Bender et al 2012) and using the implemented grammars to comb through collected texts to identify as-yet unaccounted for phenomena (see Baldwin et al, 2005). In order to fulfill this purpose, the system we build must be able to work with IGT collections as actually produced by current and future language documentation projects.

To this end, we need an internal representation format and associated import facilities that can map from the representations produced by various IGT production (and collection) systems to our internal representation. Similarly, as the results of our processing are meant to eventually enrich the documentary resources, we should be able to map from our internal representation to standard display formats at the very least.

Thus our primary use case involves the processing of collections of IGT, looking at both properties of individual items and comparing/aggregating across items, in order to extract hypotheses about linguistic properties of both individual items and whole languages. This use case also raises considerations of import and export: even if we are not using all of the information encoded in a particular collection of IGT, we should be able to translate formats in a lossless fashion.

A second use case involves the aggregation and enrichment of IGT for use in processing by others. In particular, we take the data curated by the ODIN project (Lewis and Xia, 2010) as a specific test case. The ODIN project collects IGT from linguistics papers available as PDFs on the web, and then puts the IGT through several cleaning and enriching steps. The enriched IGT includes 1) word alignment among source language line, morpheme line, gloss line, and English translation; 2) a parse tree for the English translation; 3) a projected parse tree for the source language line; 4) the alignment between nodes in the two parse trees; and so on. The result is a resource that is accessible through a web page[4] with a search interface, but also one that users might want to download in bulk for processing akin to what we describe above. This use case shares with the primary one the focus on amenability to processing, but also adds considerations of encoding provenance of individual annotations. Xia et al (2013) introduce the new ODIN corpus[5] that has been encoded in Xigt, and detail how the original and cleaned IGT are stored with enriched annotations as stand-off to the cleaned form. The initial release includes the original and cleaned text data, basic structural alignments (e.g. aligning the gloss line to the original language line), and metadata for source-document

[4] `http://odin.linguistlist.org`

[5] Available for download at: `http://uakari.ling.washington.edu/corpus/odin/`.

provenance and language names. Further annotations, such as more detailed structural alignments, bilingual word alignments (between the translation line and original language or gloss tokens), syntax trees, and dependency structures are left to future work and will be distributed as additional annotation layers. A brief description of the way this data is encoded is given in §6.

These use cases have lead us to identify seven characteristics that we would like to see in a representation of IGT.

Stand-off: Ide et al (2003) underscore the importance of the ability to deploy an annotation without doing violence to the original. This promotes reuse of linguistic data and constitutes a best practice. In this case, we want to keep the original IGT lines unchanged, while adding the results produced by subsequent processing steps as additional tiers.

Incrementality: Ide et al (2003) also note that representations of annotations must allow for incremental development of analyses. This is equally true of annotations in the form of IGT: On the one hand, documentary linguists develop analyses of data over time, and on the other hand, automatic processing of existing data of the kind done in the ODIN project provides further annotations to existing records. Furthermore, we wish to accommodate IGT instances that may include errors or omissions of various sorts, as these still may be useful for linguistic studies.[6] For instance, morpheme and gloss lines in some items in the ODIN database are not alignable but the grams in the gloss lines may still be informative for studies of the case system or inflection of a language.

Extensibility: The format should not be so rigorous as to disallow annotation beyond that envisioned by the Leipzig Glossing Rules. While it should be possible to encode (and perhaps validate compliance with) any convention defined by the LGR, the LGR rules do not cover every possible variation among the world's languages, particularly with regard to grammatical category labels. But we don't just anticipate extensions for basic interlinear annotation. Indeed, as discussed in our use cases, we envision alignment of syntactic trees to IGT, bilingual word alignments, and more, so we require our IGT representation to allow this and other extensions beyond the common analytical levels.

Complex Alignments: Related to the need for an extensible format is our desire to represent complex alignments between annotation tiers. Some examples of complex alignments are many-to-one (as a series of elements on one tier may align to a single one on another, such as morphemes to a word), one-to-zero (e.g. a case marker in the gloss line may not align to anything in the translation line), and mapping to several discontiguous text spans (e.g. a gloss of an Arabic templatic radical aligns to discontiguous text spans in the

[6] In the case of the ODIN collection, the error could stem from the data entry on the part of the linguist who wrote the paper the item was harvested from, from noise introduced in the process of conversion from PDF to the text format, or from noise introduced by the automatic methods used in the enrichment process.

source language line). Representing syntactic structure also requires a type of complex alignment in which items on a phrase structure tier have overlapping alignments to items on a tier of syntactic elements. Similarly, semantic notions such as coreference structure require complex alignments.

ID-Reference Annotations: In many linguistic theories, certain levels of analysis are nested in a tree-like structure within others (morphemes within words, for example). In some of the existing formats for IGT (e.g. Hughes et al, 2003), this constraint is captured by XML nesting of elements, and in others, such as Toolbox (discussed in §4.2), it is captured by the vertical alignment of textual columns. We choose to use an ID-reference method of annotation instead, where alignable tokens are first assigned an identifier, then annotations can refer to the identifiers in order to establish alignments. Palmer and Erk (2007) point out that the nested XML technique leads to difficulty modifying and extending the analysis. If a new annotation tier is to be added, all elements on the tier which contains the new tier must have their structure modified. The vertically aligned text technique has its own drawbacks: the tiers that are aligned must be specified separately, as presumably not every tier vertically aligns to the one above it; complex alignments are very difficult to capture; and so on. In contrast, the ID-reference technique do not suffer these drawbacks at the cost of some additional verbosity (the assignment and referencing of identifiers). We therefore echo Palmer and Erk (2007) in calling for a referential annotation scheme. This scheme allows a new tier to be added alongside existing tiers without their modification, and it simplifies the representation of complex alignments between tiers.

Applicability to Automatic Processing: Because our use cases are dedicated to mining and enriching large collections of IGT, a Xigt corpus should be easy to query and manipulate with automatic methods. For us, this means that IGT instances in the corpus can be inspected or modified independently, i.e. without requiring or affecting other IGT instances, and that suitable methods exist to retrieve the information contained within an IGT. This desideratum also implies reasonably efficient methods to iterate over IGT instances and extract information from tiers. Much of this efficiency depends on the software used to process Xigt, which is outside the scope of this paper, but the data format itself can have an effect, too. For example, IGT-XML (Palmer and Erk, 2007), described in more detail below, uses a tier-focused model, where the same tiers for all IGT (e.g. all morpheme tiers) are grouped together and separate from groupings of other tiers. While this is adequate if you wish to retrieve or modify all tiers of the same type without the context of the IGT they come from, reassembling an IGT instance requires the computer to read the entire corpus. Similarly, Annotation Graphs (Bird and Liberman, 2001), also defined in more detail below, list the source signals (audio segment, text span, etc.) for all IGT together at the beginning of a corpus and then list the annotations later, so in order to reassemble an IGT from its source signal and aligned annotations, the computer must either read through the entire corpus or hold the signals for all IGT in memory while later reading the annotations. In contrast to these two methods, Xigt groups all information specific to an IGT within the same structure. Within this IGT structure, tiers of annotation are not nested, but flat, and annotation alignments are notated through ID-reference. This approach provides a balance between expressive power and computational processability. Furthermore, for the XML serializations of Xigt, we use a small inventory of element names with variation expressed as the value of a "type" attribute. The flat structure and element naming scheme helps tools in the XML ecosystem like XPath (Berglund et al, 2007) and XSLT (Kay et al, 2007) to be flexible and efficient in processing the IGT data.[7]

Predictable Representation of Tiers: Many of the existing formats (e.g. Maeda and Bird, 2000) provide a generalized framework which is expressive enough in principle to accommodate extremely complex annotation schemes on text and audio/video. However, the greater generality of these models presents a trade-off which limits the range of assumptions that can be made about input data, limiting in turn the ability to automatically process that data. This is true both for computational linguists looking for new data sources and for software-tool makers who want to render and visualize data. Therefore, we suggest that a storage format for IGT should have a standard representation for at least those levels of analysis formalized in the Leipzig Glossing Rules plus the common levels of annotation in enriched IGT (e.g. word alignment and parse trees).

In this section, we have identified seven desiderata for a representation of IGT given our use cases. While some of these are in fact very general, pertaining to any kind of linguistic annotation (*Stand-off*, *Incrementality*, *Extensibility*), others are more specific to IGT (*Complex Alignments*, *ID-Reference Annotations*) and yet others specific to the processing of IGT in bulk (*Applicability to Automatic Processing*, *Predictable Representation of Tiers*). In the next section, we will review existing data models for linguistic annotation in general IGT through the lens of these specific desiderata.

4 Previous Work

In reviewing previous work on representations of IGT (and other annotations), we find it helpful to distinguish between *data models* and *serialization formats*. A data model, in this context, is a specification of which entities exist in the annotations and underlying data and the relationships they can stand in with each other. A serialization format, on the other hand, is a means of representing data encoded according to a data model in terms of strings. Some previous work is focused on data models (and may or may not describe possible

[7] If we had specialized element names for tiers, the XPath query or XSLT selector may need to be modified for any newly defined tiers, especially if there are non-tier elements at the same level.

serialization formats for their proposed models). Other work is focused on formats or on tools (with associated input/output formats) leaving the data models implicit. When the data models are implicit, they can be relatively strictly defined and inferable from the discussed formats—or relatively loosely defined (allowing more or less ad-hoc user extensions) and/or non-inferable but nominally encoded in the associated software. In this review, we focus on data models and group them into two relevant classes: those proposed for generalized annotation (linguistic or nonlinguistic features of potentially multimodal data artifacts) and those specific to the representation of IGT.

4.1 Data models for general annotation

Annotation Graphs/Sets: Bird and Liberman (2001) introduce Annotation Graph (AG) as a general formalism for providing annotations on any one-dimensional, serialized data. In this model, the original data to be annotated is termed a 'signal'. This generalized definition of signal allows text or audio to be annotated within the same formalism. An AG consists of a set of nodes (N), a collection of arcs (A), and a function (τ) that maps nodes to timepoints. The arcs in A are labeled with linguistic annotations which provide information about features of the signal occurring between the two time points. The example in Figure 1 illustrates this formalism for a Portuguese *O Pedro baixou a bola* "Pedro calmed down" (lit. "Pedro lowered the ball"). Here, the signal is the textual representation of the sentence.[8] N is a set of names that refer to character offsets and the mapping from the names to the offsets are specified in τ. The arcs in A mark various sections of the signal as *morph* or *word*.[9]

$$\text{Signal: } \begin{array}{cccccccccccccccccccccc} \text{O} & \_ & \text{P} & \text{e} & \text{d} & \text{r} & \text{o} & \_ & \text{b} & \text{a} & \text{i} & \text{x} & \text{o} & \text{u} & \_ & \text{a} & \_ & \text{b} & \text{o} & \text{l} & \text{a} & \\ {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 2} & {\scriptstyle 3} & {\scriptstyle 4} & {\scriptstyle 5} & {\scriptstyle 6} & {\scriptstyle 7} & {\scriptstyle 8} & {\scriptstyle 9} & {\scriptstyle 10} & {\scriptstyle 11} & {\scriptstyle 12} & {\scriptstyle 13} & {\scriptstyle 14} & {\scriptstyle 15} & {\scriptstyle 16} & {\scriptstyle 17} & {\scriptstyle 18} & {\scriptstyle 19} & {\scriptstyle 20} & {\scriptstyle 21} \end{array}$$

$$N = \{\alpha,\ \beta,\ \gamma,\ \delta,\ \epsilon,\ \eta,\ \theta,\ \iota,\ \kappa,\ \lambda\}$$

$$A = \left\{ \begin{array}{lll} <\alpha, morph, \beta>, & <\alpha, word, \beta>, & \\ <\gamma, morph, \delta>, & <\gamma, word, \delta>, & \\ <\epsilon, morph, \zeta>, & <\zeta, morph, \eta>, & <\epsilon, word, \eta>, \\ <\theta morph \iota>, & <\theta word \iota>, & \\ <\kappa morph \lambda>, & <\kappa word \lambda> & \end{array} \right\}$$

$$\tau = \left\{ \begin{array}{lll} \alpha \to 0, & \beta \to 1, & \\ \gamma \to 2, & \delta \to 7, & \\ \epsilon \to 8, & \zeta \to 12, & \eta \to 14, \\ \theta \to 15, & \iota \to 16, & \\ \kappa \to 17, & \lambda \to 21 & \end{array} \right\}$$

Fig. 1 Formal representation of Annotation Graphs for the Portuguese sentence *O Pedro baixou a bola* "Pedro calmed down" (lit. "Pedro lowered the ball")

[8] The indices 0–21 are the offsets of the characters in the sentence and they are not part of the signal.

[9] This example is offered for illustration; the AG formalism does not restrict the possible units of time referred to in the range of τ.

In Bird et al (2000), the Annotation Graph formalism is used as the core of a three-layer database platform: ATLAS (Architecture and Tools for Linguistic Analysis Systems). Within the ATLAS project, the notion of annotation graphs is generalized to annotation sets. This extension adds the ability to annotate multidimensional signals instead of just linear ones (i.e. video and image data).

Maeda and Bird (2000) apply the Annotation Graph format as a basis for IGT representation. In order to capture the hierarchical layers of annotations which are characteristic of IGT (i.e. words within sentences, morphemes within words, etc.) as well as the alignments which hold between certain layers (e.g. alignment between morphemes and glosses), they introduce the notions of annotation groups and subgroups (within an annotation graph). With this extension, AG provides an elegant way to represent legacy IGT. However, its application to our goal of enriching IGT structures with nonlinear structures is less straightforward. Although the AG formalism generally captures the fundamental properties of annotation, one of our goals is to provide a straightforward way to represent complex alignments with potentially overlapping or non-contiguous spans; it is not clear how AG can be used to represent those layers.

Abstract Corpus Model (ACM): ACM provides the underlying model for the ELAN annotation tools (Brugman and Russel, 2004). While the ACM model can in principle be used for annotated text, the focus is on audio and video annotation. ACM defines `tier` objects which can contain two types of annotations: alignable annotations and reference annotations. Alignable annotations refer directly to two "time slots" of the media being annotated while reference annotations inherit their alignment information indirectly from an ancestor annotation. Constraints placed upon tiers include that annotations on a given tier cannot overlap in the timespan they refer to and that a mix of alignable and reference annotations on the same tier is not allowed.

Tiers can also be constrained such that the annotations they contain are required to fall into specific relationships with the annotations on a parent tier. For example, in 'symbolic subdivision' annotations referring to the same parent are ordered. This sort of tier constraint is appropriate for capturing the hierarchical relationships found in IGT such as the decomposition of words into morphemes. Another relevant type of tier constraint is called 'symbolic association', where a one-to-one association exists with a parent annotation. This constraint type is appropriate whenever items in the parent tier are being tagged according to some attribute, such as POS tags on words, or glosses on morphemes.

The ACM model provides a powerful framework for capturing many other annotation relationships beyond the ones mentioned here. Because any tier can be independent or constrained in several predefined ways, the format provides flexibility. As future work, Brugman and Russel (2004) describe a tier constraint type which allows annotations on a dependent tier to reference annotations on a specific parent tier in order to allow construction of syntactic trees, so in principle the model does support complex annotations. However, the model enforces constraints against overlapping time spans which we do not want for our purposes. For example, with manual languages there may be overlapping gestures, such as with gaze and hand signs, and these become very difficult to annotate with non-overlapping time spans.

LAF/GrAF: Graph Annotation Framework (GrAF; Ide and Suderman, 2007), extending Linguistic Annotation Format (LAF; Ide et al, 2003), provides an abstract data model which aims to be universal in its scope of linguistic annotations. The format relies on annotations being translatable to a generalized feature structure model.

Within the LAF model, two types of annotations are defined: segmentation and linguistic. The former type provides reference points to which the latter type refer. Like ATLAS, LAF aims to provide an ecosystem around which application services can be developed and with the extension to GrAF, the model's export format can be processed by generic graph processing tools.

Both the ACM and LAF/GrAF model provide an expressive framework which could in principle be used to represent enriched IGT, but for our purposes these models are too general. That is, we seek a data model that is more tailored to IGT and our use cases (while still being extensible). It should be possible to create a translation between Xigt and LAF/GrAF, such that Xigt annotations can be translated into a common format with other annotations that can be mapped to GrAF, and thus gain the interoperability benefits of that more general model.[10] We intend to explore this in future work.

4.2 Data models specific to IGT

Shoebox/Toolbox: Shoebox/Toolbox is both a tool to aid linguists in the creation of an IGT corpus and an eponymous storage format.[11,12] This format serves as the basis for linguistic databases of IGT examples and other materials such as lexica and dictionaries. The format specifies database items expressed as blocks of labeled lines in text files, as in Figure 2. Line names are preceded by a backslash. These ad-hoc identifiers are not fixed by the software, but rather freely defined by the linguist (user).

The Shoebox software does provide the ability to define types of tiers and relationships between them in additional files. However, the storage format doesn't support this directly; the ability is a property of the software and not made explicit in the storage of the IGT.

The Shoebox format provides simplicity and flexibility, which explains its popularity and longevity. But the open-ended set of possible line names and lack of predefined semantics or predefined tiers means that no assumptions can

[10] Mapping from LAF/GrAF to Xigt would necessarily be limited to the kinds of annotations that Xigt is designed to handle.

[11] Also called SIL Standard Format

[12] `http://www-01.sil.org/computing/shoebox/`

```
\id   1
\tx   O          Pedro   baixou                 a          bola
\me   O          Pedro   baix    -ou            a          bola
\gl   the.M.SG   Pedro   lower   -PST.IND.3.SG   the.F.SG   ball.F.SG
\tr   Pedro calmed down.
\lt   Pedro lowered the ball.
```

Fig. 2 Example of the Toolbox format for the Portuguese sentence *O Pedro baixou a bola* "Pedro calmed down" (lit. "Pedro lowered the ball")

be made about alignment of tokens from line to line in a given Shoebox corpus without consulting metadata about the corpus' construction. This restricts automatic processing of alignment between tiers because which tiers are to be aligned is unknown. For example, if a user wants to add a new tier of annotation (say syntactic structure), she could use the label name "phs" and an ad-hoc system to indicate alignment spans on the "wds" tier, as shown in Figure 3. Another user with the same goal could choose a different label name and a different system. Without the knowledge of the meanings of those label names or the workings of the ad-hoc systems, the ability to combine IGT data from different Shoebox corpora is lost.

```
\wds i saw a car
\phs np=0,1 vp=1,4 np=2,4
```

Fig. 3 Ad-hoc encoding of syntactic structure in the Toolbox format

EMELD/BHB: Hughes et al (2003), in the context of EMELD (Electronic Metastructure for Endangered Languages Data[13]), introduce an XML format specifically for capturing IGT. They provide a four-level XML containment structure to capture hierarchical nesting of standard annotations. That is, the four levels of the model: **interlinear-text**, **phrase**, **word** and **morpheme** are represented as elements which are contained within the value of an XML element at the level above. Figure 4 shows our Portuguese example in this format.

This model generalizes content to a single element type `item`, which allows for general processing rules to be written to access content from an arbitrary level of annotation. However, as discussed above, direct nesting of hierarchical levels leads to difficulty adding new tiers of annotations beyond the four proposed in this model. Furthermore, the direct nesting of elements rather than the use of ID-reference means that complex alignments are not supported.

IGT-XML: Palmer and Erk (2007) point out that the use of XML containment in annotation elements leaves little flexibility to add or remove layers. Thus the representation they propose, IGT-XML, provides a similar model to

[13] `http://emeld.org`

```
<interlinear-text>
  <phrases>
    <phrase>
      <item type="number">1</item>
      <item type="gls">Pedro calmed down.</item>
      <words>
        <word>
          <item type="txt">O</item>
          <morphemes>
            <morph><item type="gls">the.M.SG</item></morph>
          </morphemes>
        </word>
        <word>
          <item type="txt">Pedro</item>
          <morphemes>
            <morph><item type="gls">Pedro</item></morph>
          </morphemes>
        </word>
        <word>
          <morphemes>
            <morph>
              <item type="txt">baix</item>
              <item type="gls">lower</item>
            </morph>
            <morph>
              <item type="txt">ou</item>
              <item type="gls">PST.IND.3.SG</item>
            </morph>
          </morphemes>
        </word>
        <word>
          <item type="txt">a</item>
          <morphemes>
            <morph><item type="gls">the.F.SG</item></morph>
          </morphemes>
        </word>
        <word>
          <item type="txt">bola</item>
          <morphemes>
            <morph><item type="gls">ball.F.SG</item></morph>
          </morphemes>
        </word>
      </words>
    </phrase>
  </phrases>
</interlinear-text>
```

Fig. 4 Example of the EMELD/BHB format for the Portuguese sentence *O Pedro baixou a bola* "Pedro calmed down" (lit. "Pedro lowered the ball").

BHB/EMELD, but uses ID-reference between annotations rather than direct containment. This is a useful step forward, but IGT-XML also focuses on tiers; it groups all similar tiers in a document under one XML element, leading to processing inefficiency when operating on IGT instances instead of solitary tiers. For example, consider using an XSLT transformation to visualize the IGT instances on a webpage. If the IGT-XML data is sent by a stream (e.g. over an internet connection), the viewer would have to wait until the most of the document is transferred before seeing the first IGT instance. Furthermore, IGT-XML only allows one-to-one alignments, and not sub-segmental (e.g. character spans) or one-to-many (e.g. for discontiguous selections) alignments, although it does allow for many-to-one alignments (e.g. for portmanteau morphemes). In order to define segmentation, multiple annotations can refer to the same identifier (e.g. multiple morphemes refer to the same word identifier), which gives an implicit segmentation as it is not explicit where one segment ends and another begins.

4.3 Discussion

In the preceding review of the existing data models for IGT, we find that they are either too general or too specific. On the one hand, there are models that are designed to be flexible enough to handle any type of linguistic annotation, but aren't really tailored to IGT. On the other hand, there are models built around legacy IGT that don't readily extend to the use cases we have in mind. In designing Xigt we have endeavored to combine the features of the various previous proposals that respond to our desiderata, creating a model that is specific to automatic processing of IGT, extensible in the ways we require, and (in principle) interoperable—via translation—with more general annotation storage formats.

5 The Xigt Specification

We aim for Xigt to be expressively adequate for both coarse- and fine-grained annotation, while keeping IGT as the core interest. A coarse-grained IGT annotation might align an entire tier of morphemes—as an atomic string—to a sentence. A fine-grained annotation might align words to sub-spans of the sentence, and then morphemes to sub-spans of the words. We consider an IGT to be multiple tiers of linear data with hierarchical relationships among the tiers. With this conception in mind, we give an overview of the Xigt data model in §5.1, and an XML serialization format in §5.2.[14] We chose XML to make use of tools in the XML ecosystem (XPath; Berglund et al 2007, XSLT; Kay et al 2007, etc.), and for its general familiarity. The Xigt data model could be serialized into other formats, such as JSON, or even used for describing and populating a relational database, but in this paper we only describe XML serialization.

[14] For more information on the project and to download code and resources, please refer to the project website at `http://depts.washington.edu/uwcl/xigt`.

```
<text id="T1" lg="por">
<metadata idref="T1">
  <!-- OLAC metadata here -->
</metadata>
  <body>
    <phrases>
      <phrase id="T1.P1">
        <plaintext>O Pedro baixou a bola</plaintext>
        <word id="T1.P1.W1" text="O" />
        <word id="T1.P1.W2" text="Pedro" />
        <word id="T1.P1.W3" text="baixou" />
        <word id="T1.P1.W4" text="a" />
        <word id="T1.P1.W5" text="bola" />
      </phrase>
    </phrases>
  <morphemes>
    <phrase idref="T1.P1">
      <morph idref="T1.P1.W1" id="T1.P1.W1.M1" text="O" />
      <morph idref="T1.P1.W2" id="T1.P1.W2.M1" text="Pedro" />
      <morph idref="T1.P1.W3" id="T1.P1.W3.M1" text="baix" />
      <morph idref="T1.P1.W3" id="T1.P1.W3.M2" text="ou">
        <type l="suf" />
      </morph>
      <morph idref="T1.P1.W4" id="T1.P1.W4.M1" text="a" />
      <morph idref="T1.P1.W5" id="T1.P1.W5.M1" text="bola" />
    </phrase>
  </morphemes>
  <gloss>
    <phrase idref="T1.P1">
      <gls idref="T1.P1.W1.M1" text="the.M.SG" />
      <gls idref="T1.P1.W2.M1" text="Pedro" />
      <gls idref="T1.P1.W3.M1" text="lower" />
      <gls idref="T1.P1.W3.M2" text="PST.IND.3.SG" />
      <gls idref="T1.P1.W4.M1" text="the.F.SG" />
      <gls idref="T1.P1.W5.M1" text="ball.F.SG" />
    </phrase>
  </gloss>
  <translations>
    <phrase idref="T1.P1">
      <trans id="T1.P1.Tr1" lg="en">Pedro calmed down.</trans>
      <trans id="T1.P1.Tr2" lg="en">Pedro lowered the ball.</trans>
    </phrase>
  </translations>
  </body>
</text>
```

Fig. 5 Example of the IGT-XML format for the Portuguese sentence *O Pedro baixou a bola* "Pedro calmed down" (lit. "Pedro lowered the ball").

5.1 The Xigt Data Model

Here we explain the basics of the data model, including the structures involved and the methods of aligning them. The manifestation of the conceptual data model is the open-source, core code library of the Xigt project.[15]

[15] The public repository is available from the project website.

5.1.1 Structures

The Xigt data model uses just four structures to encode a corpus of IGT data and annotations: 1) a `xigt-corpus` groups IGT instances and metadata relevant to all items, 2) an `igt` contains all of the tiers represented by an IGT instance, 3) a `tier` groups data or annotations of the same type, and 4) an `item` contains the actual data or annotations. This organization keeps the overall structure mostly flat; a new type of annotation would not require another level of nested structure, but just a new `tier` appended to the list of tiers on an `igt`. This core model is kept simple and predictable so processing tools can reliably parse—even if they cannot accurately interpret—new types of information added to a Xigtcorpus via extensions. A fifth, optional structure called `metadata` may appear under `xigt-corpus`, `igt`, or `tier`, and is used to describe the properties of the data encoded by IGT instances, such as the language or author. (2) is a sample IGT for Icelandic,[16] and a visualization of the primary data structures involved in modeling it is given in Figure 6.

(2)	Ég	hjálpa-ð-i	þeim.
	1SG.NOM	help-PAST-1SG	3PL.DAT
	‘I helped them.’ [isl]		

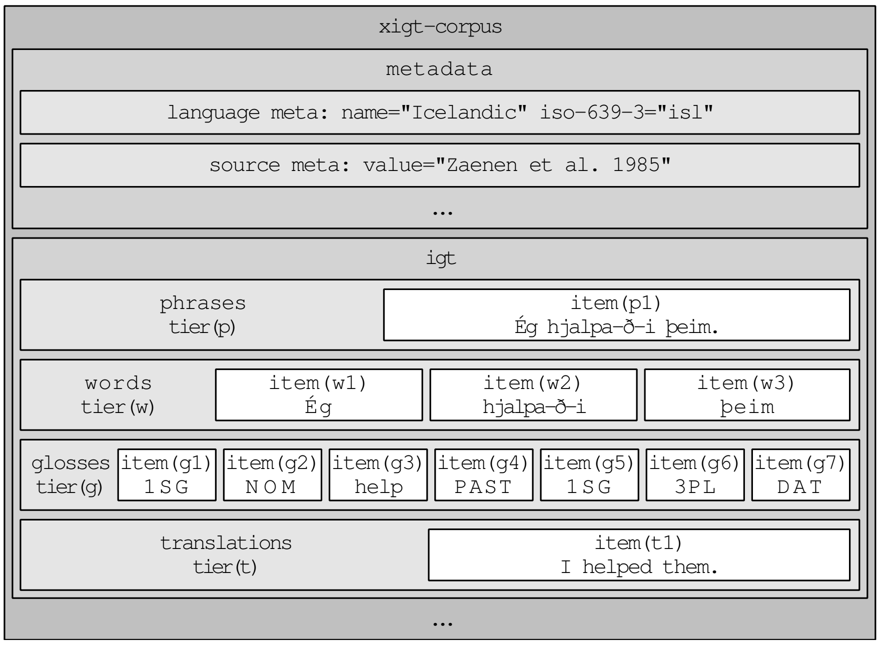

Fig. 6 Example structures used for modeling (2)

[16] The example in Figure 2 is from Zaenen et al (1985) via Bakker and Siewierska (2007) in ODIN.

5.1.2 Alignments

To represent the relationships (i.e. alignments) encoded in IGT, Xigt allows any structure (but most interestingly `tier` and `item`) to declare an `id` that can be referenced. There are two ways to draw references: **basic alignments** and **alignment expressions**. Basic alignments are one-to-one references from an object to a single `id`. In Figure 6, a basic alignment would align tier `w` to tier `p`. Basic alignments are also used for referring to metadata, such as for stating that an IGT comes from a book that has been described in the corpus-level metadata (thus preventing the need to redefine the book for every IGT). Alignment expressions are one-to-many references used solely by `item` objects. They are able to select sub-spans of the content of a referenced `item`. In Figure 6, an alignment expression would align item `w1` to a sub-span (the first two characters) of item `p1`.

Alignment expressions Alignment expressions, which both unify and extend the methods of alignment from previous models, are a novel contribution of the Xigt model. Annotation Graphs, for example, segment data by specifying anchor points in a signal (text, audio, etc.), possibly with offsets (characters, milliseconds, etc.), and annotations select spans by specifying start and end anchor points. In Xigt, data segmentation and annotation alignment (i.e. span selection) are both accomplished by alignment expressions.[17] There are advantages and disadvantages to each approach. The anchor points for Annotation Graphs are useful when later adjustments are necessary, for example when one reanalyzes a speech signal and wants to change the endpoints of a word or morpheme, because all tiers that use the anchor will refer to the new selection without needing to be changed. Alignment expressions, however, are more suited to our goals because of their ability to select discontiguous spans and how they can readily select from different tiers.

Alignment expressions are designed to be compact and expressive. We define a small grammar for alignment expressions in Figure 7. A single **selection** is an item identifier (e.g. `w2`) with an optional content **range** (e.g. `[0:6]`). The range specifies **spans** of content (e.g. `0:6` selects characters between positions 0 and 6) to select from the referenced item. Identifiers are strings beginning with a letter, and in general they must be unique within an `igt`.[18] Spans on text data use 0-indexed pivot points, as shown in Figure 8, and are similar to slicing operations in common programming languages like Perl or Python. If the range is omitted from a selection, it is interpreted as selecting the entire content of the referenced item. The alignment expressions, with their resolved

[17] As we are mostly concerned with textual data, this paper only discusses the segmentation and alignment of character spans. Xigt is capable of representing annotations of audio data, but explicit support for such annotations is relegated to future work. Support for non-linear data (such as images) is beyond the current scope of the project.

[18] As the scope of alignment is a single IGT, `tier` and `item` identifiers do not need to be unique within a `xigt-corpus`.

content, for each item in the glosses tier of Figure 6 are given in Table 1. Some more illustrative examples of alignment expressions are given later in Figure 9.

⟨*AlignExpr*⟩	:=	⟨*Selection*⟩ | ⟨*Selection*⟩ ⟨*Delim*⟩ ⟨*AlignExpr*⟩
⟨*Selection*⟩	:=	⟨*identifier*⟩ ⟨*Range*⟩
⟨*Delim*⟩	:=	'`,`' | '`+`'
⟨*Range*⟩	:=	'`[`' ⟨*Spans*⟩ '`]`' | ϵ
⟨*Spans*⟩	:=	⟨*Span*⟩ | ⟨*Span*⟩ ⟨*Delim*⟩ ⟨*Spans*⟩
⟨*Span*⟩	:=	⟨*integer*⟩ '`:`' ⟨*integer*⟩

Fig. 7 A grammar for alignment expressions

```
     É  g  _  h  j  a  l  p  a  -  ð  -  i  _  þ  e  i  m  .
p1| 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
w1| 0  1  2
w2|          0  1  2  3  4  5  6  7  8  9  10
w3|                                           0  1  2  3  4
```

Fig. 8 Character pivot positions for items `p1` and `w1`–`w3` of Figure 6

Item	Alignment expression	Resolution
`w1`	`p1[0:2]`	Ég
`w2`	`p1[3:13]`	hjálpaði
`w3`	`p1[14:18]`	þeim
`g1`	`w1`	Ég
`g2`	`w1`	Ég
`g3`	`w2[0:6]`	hjálpa
`g4`	`w2[7:8]`	ð
`g5`	`w2[9:10]`	i
`g6`	`w3`	þeim
`g7`	`w3`	þeim

Table 1 Alignment expressions for the words and glosses tier of (2) as represented in Figure 6.

Multiple selections or multiple spans may be combined with one of two delimiters: "`+`" or "`,`". By default, "`+`" adjoins them directly, and "`,`" joins them with an intervening separator. These are useful when selecting discontiguous elements. For example, consider the English sentence *Pick the book up* with a words tier where item `w1` is "Pick", item `w2` is "the", and so on. Now consider creating a word-senses tier where "pick up" should be given a single, unique sense. In order to select the words to align to the sense, an alignment expression of `w1,w4` would yield "Pick up", which maintains the separation between the two words, but groups them as a single unit to be annotated. If, instead, `w1+w4` were used, it would yield "Pickup"; a word for a kind of vehicle instead of the intended verbal sense.

We recognize that different kinds of data may require idiosyncratic separators, such as a space character, a zero-width space character (Unicode `U+200B`;

for languages in which a space is not defined as a word-separator, like Chinese, Japanese, or Thai), or gaps of silence (for audio data). Because alignments are likely to occur on annotation tiers rather than tiers with native orthography or audio data, "`,`" joins selections with a space character as a convenient default. Both "`+`" and "`,`" may be redefined in an extension.

Some example alignment expressions with their resolved strings are given in Figure 9. Note that different alignment expressions can resolve to the same string. The invalidity of the final expression is explained below.

`p1`	→	Ég hjalpa-ð-i þeim.
`p1[0:2]`	→	Ég
`p1[3:13]`	→	hjalpa-ð-i
`w2`	→	hjalpa-ð-i
`w2[0:6+7:8+9:10]`	→	hjalpaði
`w2[0:6]+w2[7:8]+w2[9:10]`	→	hjalpaði
`w1,w2,w3`	→	Ég hjalpa-ð-i þeim
`w1,g1`	→	(invalid)

Fig. 9 Example alignment expressions and their resolutions using items from Figure 6

Constraints on alignment expressions There is a structural constraint for alignment expressions: an `item` may only refer to an `item` within the `tier` its own `tier` refers to. That is, a tier first draws a relationship to another tier, then items in the first tier may annotate those in the second, but no others. Looking back at Figure 6, if the glosses tier `g` refers to the words tier `w`, this constraint would ensure that gloss items (`g1`–`g7`) may refer only to word items (`w1`–`w3`), and not to items on the phrases tier or translations tier. This constraint is artificial, but—we think—well motivated, as it would be strange for some annotation to apply to multiple tiers (i.e. different kinds of data) simultaneously.

There are times when a tier must refer to more than one other tier, albeit in different ways. For example, in order to capture bilingual word alignments between the translation and original language lines, we need a way to select multiple words on either the translation line or original language line, since words don't always translate one-to-one. If we just put the alignments on one of these tiers (say, on the translation tier, selecting the original language words), we could not get alignments involving more than one translation word. Instead, we create a third tier with items having two alignment expressions each: a source and a target. With this new tier, we can capture many-to-many alignments. In order for this configuration to still be valid with respect to the constraint given above, we give references names, and the constraint only applies to references with the same name. In XML these names are those of the attributes that specify the references. Figure 10 illustrates what the bilingual word alignment tier would look like if added to Figure 6, where the references are shown in angle brackets.

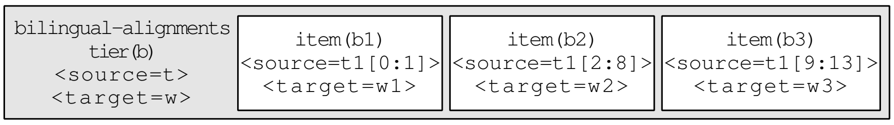

Fig. 10 Example tier for bilingual word alignment)

Floating alignments If multiple items share the same alignment, they have **floating alignments**. This is useful for phenomena like portmanteau morphemes, but also for relaxed (i.e. less granular) annotation where the user does not want to be specific about the alignments. In Figure 6, glosses `g1` and `g2` are portmanteau morphs and would be in a floating alignment with word `w1`. Even though `g1` and `g2` both align to `w1`, they maintain their relative order, so if one were to (re-)serialize them as a string, the result would be "I.NOM" and not "NOM.I".

5.1.3 Items and Annotation Content

Most `items` will have some content associated with them, whether it is source data, annotations, or something else. Xigt does not make any distinction between source data (such as sentences or words) and subsequent annotations, although in practice the source data do not often align to anything else. For all items, content can be specified in two ways: it can be specified directly on the item, or it can be selected with a special alignment. Content selected via alignments is dynamic—it will update when the source data changes—but the ranges of sub-span selections may need to be adjusted.

It is also possible to both select existing content and introduce new content. In this case, the content introduced directly on the item **shadows** the selected content, meaning that the selected content is not used. The purpose is to trace where data comes from; the alignment for the selection points to the source of the data, but the actual content is overridden with the introduced data. This technique is useful for smoothing over OCR or ASR errors (so annotations target the clean form and not the erroneous one), or for providing the underlying form of morphemes rather than the overt phonological realization (as in (1), where "Ndakanga" is aligned to the underlying morphemes "Ndi", "aka", and "nga").

5.1.4 Primary Tier Types and Alignment Types

Different kinds of data are encoded in subtypes of the primary data structures. Xigt's core model provides five basic `tier` subtypes: `phrases`, `words`, `morphemes`, `glosses`, and `translations`. Xigt also provides several `item` subtypes, such as `clitic` and `affix` to distinguish multiple kinds of morpheme elements, or `transliteration` to distinguish phrases in the source language orthography from a transliteration in another script.

All tiers and items have three basic kinds of references: `alignment`, `content`, and `segmentation`. The `alignment` reference selects the target of the annotation, while the `content` reference is used for selecting item content. The `segmentation` reference is the combination of the other two (it both specifies the target of the annotation and uses that target as the content of the annotation), and is convenient for specifying subparts of other items, such as selecting words from phrases and morphemes from words. As such, the `segmentation` reference is incompatible with the other two.

5.2 The Xigt XML Data Format

The XML serialization format for Xigt closely resembles the data model. It was designed to optimize predictability of the data without sacrificing expressibility or extensibility. We therefore have a basic, fixed core format, and extensions can add information but not remove or alter the core format. The implementation of this format consists of two parts: 1) a RelaxNG (Clark and Murata, 2001) schema for validating the XML documents, and 2) code for the serialization of XML from virtual data structures.[19]

Data Structure The XML format follows the data model naming conventions: a `<xigt-corpus>` root element contains `<igt>` elements, which in turn contain all `<tier>` elements relevant to an IGT. The `<tier>` elements then contain the primary or annotation data, which are encapsulated in `<item>` elements. All four of these kinds of elements are ID-bearing (via an `id` attribute), meaning they may establish an ID that other elements can reference. The `<igt>`, `<tier>`, and `<item>` elements may specify subtypes with a `type` attribute. These subtypes can place constraints on their child elements' subtypes and attributes. `<xigt-corpus>`, `<igt>`, and `<tier>` elements may have zero or more `<metadata>` children, which describe the data contained in any `<item>` elements under them, such as the language or script used, author of the IGT, grammaticality of the example, etc. References appear as attributes on the referencing object. To specify these references, an `<item>` may have an `alignment` or a `content` attribute, or both, or it may have a `segmentation` attribute.

Figure 11 is a simple example of the Xigt XML format based on the IGT in (3) that showcases some of Xigt's features, such as alignment expressions, reference-based content selection, floating alignments, and data shadowing. In this example, alignment expressions segment the Spanish word *cocinas* 'you cook' into two morphemes, *cocin* and *as*. The glosses tier then annotates these morphemes. The four glosses (2nd person, singular number, present tense, and indicative mood) are all in a floating alignment with the same morpheme, *as*, meaning they all simultaneously annotate it. Because the glosses tier introduces annotation content (*cook*, *2*, etc.), the contents shadow the aligned content in the morphemes tier, such that if another tier annotates on top of

[19] These, too, are available at the project's public repository.

the glosses, it will annotate the glosses and not the morphemes the glosses align to.

(3) cocinas
cocin-as
cook-2.SG.PRS.IND
'(You) cook' [spa]

```
<?xml version="1.0" encoding="utf-8"?>
<xigt-corpus>
  <igt id="i1">
    <tier type="words" id="w">
      <item id="w1">cocinas</item>
    </tier>
    <tier type="morphemes" id="m" alignment="w">
      <item id="m1" alignment="w1[0:5]"/>
      <item id="m2" alignment="w1[5:7]"/>
    <tier type="glosses" id="g" alignment="m">
      <item id="g1" alignment="m1">cook</item>
      <item id="g2" alignment="m2">2</item>
      <item id="g3" alignment="m2">SG</item>
      <item id="g4" alignment="m2">PRS</item>
      <item id="g5" alignment="m2">IND</item>
    </tier>
    <tier type="translations" id="t" alignment="w">
      <item id="t1" alignment="w1">(You) cook</item>
    </tier>
  </igt>
</xigt-corpus>
```

Fig. 11 Simple example of the Xigt format for (3)

5.3 Extensions

While the core Xigt framework is likely sufficient for encoding and operating on IGT data, some users may need specialized logic for certain applications. As mentioned, the data model has a small, fixed inventory of structures and the XML format similarly fixes the element names. These constraints allow for extensions that are predictable in structure and thus easy for processors to handle. Simple extensions, such as adding a required attribute or a new basic tier type, only need an updated schema for validation, because the code can handle an unknown attribute or tier type as long as it doesn't need new logic for interpreting the new data. Structural extensions, such as adding a new reference attribute for additional kinds of annotation alignment, require changes to the serialization code. This code needs to be aware that the new attribute is indeed a reference and not merely a descriptive attribute (e.g.

one that says whether the content is manually transcribed or automatically generated).

In either case, we make both the schema and the code easy to extend. Symbols can be redefined so only the changes need to be added, and the default behavior is used everywhere else. We provide some example extensions at the project's website, and explain one such extension in §6.

6 Case Study: Importing ODIN to Xigt

In developing Xigt, one key use case has been the importing of IGT from ODIN (Lewis and Xia, 2010) for processing in the AGGREGATION project[20] (Bender et al, 2013). As the ODIN data is initially text extracted from PDFs, an important sub-task is the conversion of the ODIN textual data into the Xigt XML format. Our motivations for doing this are twofold: On the one hand, we want to encode the raw ODIN text data into a format that is easier for a computer to interpret and process, and on the other hand, we want to enable the enrichment of IGT with bilingual word alignments and syntax trees. In this section, we will describe the extensions for storing ODIN data, and briefly cover those for enriching the IGT.

6.1 ODIN Textual Data

ODIN researchers have identified and extracted myriad IGT examples from PDF documents found on the web. The lines are annotated with the line number they came from in the initial PDF extraction, and tags that explain their content, such as “L” for the source text (or “language line”), “G” for glosses, “T” for translation, and occasionally metadata from the example (like the language name or author). Each block of lines representing an IGT instance is preceded by generated metadata, such as the language—which has been identified automatically from the source line—and the ID of the document the example came from. The result of this preprocessing, shown in Figure 12,[21] is what we work with in converting to the Xigt XML format.

```
doc_id=397 959 961 L G T
language: korean (kor)
line=959 tag=L:   1 Nay-ka ai-eykey pap-ul mek-i-ess-ta
line=960 tag=G:     I-Nom child-Dat rice-Acc eat-Caus-Pst-Dec
line=961 tag=T:     ‘I made the child eat rice.’
```

Fig. 12 Example ODIN text data for Korean

[20] http://depts.washington.edu/uwcl/aggregation/
[21] The example in Figure 12 is from Bratt (1996).

Figure 12 is a particularly clean example. Note that even this clean example has some extraneous data, such as the example number on the language line, the quotes on the translation line, and leading spaces on all three lines. Some more challenging examples may have PDF extraction errors (e.g. diacritics not appearing on their respective glyphs); OCR errors; two-column IGT; long, wrapped lines; multiple (i.e. alternate) gloss lines or translations; differing numbers of tokens on the language and gloss lines (which makes alignment difficult); and so on. For example, Figure 13[22] has PDF extraction errors (the language line is split into two; the final word should be "sıkıyor"), which causes a difference in token counts in the language and gloss lines, and inline alternations ("O" ideally aligns to both "He" and "It" as alternates, but "He/It" is treated as a single token).

```
doc_id=3645 10753 10756 L+CR L+CR G T
language: turkish (tur)
line=10753 tag=L+CR:  21)   O     ben-i     si i o r
line=10754 tag=L+CR:                          ky
line=10755 tag=G:         He/It me-ACC bores
line=10756 tag=T:         'He/It bores/frustrates me'
```

Fig. 13 Example of noisy ODIN text data for Turkish

We perform some cleaning steps to try and account for predictable kinds of noise, but we are unable to get a clean representation of every example. For this reason, Xigt's ability to annotate at different levels of granularity (related to our desideratum of incrementality) helps us encode as much information as is available in each case. For example, the IGT in Figure 12 has the same number of space-separated tokens in the language and gloss lines, and each token has the same number of hyphen-separated morphemes. We can, therefore, fairly confidently align individual gloss tokens to words, and even gloss grams to morphemes. The IGT in Figure 13, however, has an unequal number of gloss and word tokens (assuming we cannot automatically recover the word with PDF extraction errors), so we cannot confidently align glosses to words. Instead, we align the entire gloss line to the language line, without segmenting tokens.

The algorithms we deploy for cleanup and interpretation of the textual ODIN data are potentially lossy. To prevent any actual loss of information, we include the original text lines in a tier,[23] and the cleaned text lines in a second tier. The contents of the first tier remain unchanged as much as possible, but occasionally we had to replace control characters—introduced during PDF

[22] The example in Figure 13 is from Cagri (2005).

[23] Not a tier in the traditional sense, but rather a container of data related to the IGT. We could alternatively put these lines in a `<metadata>` element, but then getting alignments to work would require a more complicated extension, as Xigt is already set up to align tiers to one another, but not tiers to metadata. Thus it is a practical decision to put the information in a tier.

extraction—that are invalid in XML, such as the form feed character. We do not directly use the original text data in the AGGREGATION project, but we keep it for the sake of storing and distributing the Xigt-formatted ODIN corpus as stand-off annotation, for which the second tier of cleaned text is the target.[24]

6.2 Xigt Extensions

The ODIN extension to Xigt consists of several changes to the schema, but no special serialization code is necessary. We define two new tiers, `odin-raw` and `odin-clean`, both with optional attributes for the line number and tag. These ODIN text tiers are only used for establishing a link to the original data so the structural annotations are stand-off, and do not affect the actual IGT structure at all. In other words, the extension is a monotonic addition and can be ignored entirely without disrupting the relationships in the IGT extracted from the text. Removing the extension (i.e. encoding the data without the stand-off tiers), however, would result in a non-stand-off, potentially lossy representation of the ODIN textual IGT.

Figure 14 shows an XML-formatted[25] Xigt corpus with the textual Korean IGT from Figure 12. The `odin-raw` tier contains the original text from ODIN, and the `odin-clean` tier has the cleaned version (with the example number, quotes, and extraneous spaces removed). The IGT annotations (specification of words, glosses, relationships between them, etc.) follow, selecting their content with the `content` attribute, which makes them stand-off annotations, rather than introducing it themselves. In the interest of space, the tier for morphemes is not shown, but for this example it is possible to confidently align sub-glosses to morphemes. Note that only one tier needs to select a unique line from the ODIN text; derivative tiers can simply segment the first one, as is done with the words tier from phrases. Annotation alignments, via the `alignment` attribute, target only the IGT structures, and not the ODIN text tiers. This separation of concerns makes it easy to compare IGT utilizing stand-off or stand-alone annotations, and also makes it easy to convert to stand-alone annotation by resolving the content selections and removing the ODIN tiers.

Figure 15 is an example using the noisy Turkish data from Figure 13. We are less confident about how to segment the data, so instead we align the glosses tier directly to the phrases tier without an intervening words tier, and

[24] Regarding stand-off annotation, we could have similarly pointed to character offsets in an external file, but as the textual data is extracted from PDFs and not stable, we chose to copy the relevant lines into the XML file.

[25] Regarding the strange line wrapping; the `>` character of the opening `<item>` tags on the ODIN text tiers are on a new line to help show the initial spaces on each line (or absence thereof).

```xml
<?xml version="1.0" encoding="utf-8"?>
<xigt-corpus alignment-method="auto">
  <metadata type="xigt-meta">
    <meta type="language" name="korean" iso-639-3="kor"
          tiers="phrases words"/>
  </metadata>
  <igt id="i1" doc-id="397" line-range="959 961" line-types="L G T">
    <tier type="odin-raw" id="o">
      <item id="o1" line="959" tag="L"
          >   1    Nay-ka ai-eykey pap-ul mek-i-ess-ta</item>
      <item id="o2" line="960" tag="G"
          >     I-Nom child-Dat rice-Acc eat-Caus-Pst-Dec</item>
      <item id="o3" line="961" tag="T"
          >     'I made the child eat rice.'</item>
    </tier>
    <tier type="odin-clean" id="c">
      <item id="c1" line="959" tag="L"
          >Nay-ka ai-eykey pap-ul mek-i-ess-ta</item>
      <item id="c2" line="960" tag="G"
          >I-Nom child-Dat rice-Acc eat-Caus-Pst-Dec</item>
      <item id="c3" line="961" tag="T"
          >I made the child eat rice.</item>
    <tier type="phrases" id="p" content="c">
      <item id="p1" content="c1[5:40]"/>
    </tier>
    <tier type="words" id="w" segmentation="p">
      <item id="w1" segmentation="p1[0:6]"/>
      <item id="w2" segmentation="p1[7:15]"/>
      <item id="w3" segmentation="p1[16:22]"/>
      <item id="w4" segmentation="p1[23:35]"/>
    </tier>
    <tier type="glosses" id="g" alignment="w" content="c">
      <item id="g1" alignment="w1" content="c2[5:10]"/>
      <item id="g2" alignment="w2" content="c2[11:20]"/>
      <item id="g3" alignment="w3" content="c2[21:29]"/>
      <item id="g4" alignment="w4" content="c2[30:46]"/>
    </tier>
    <tier type="translations" id="t" alignment="p" content="c">
      <item id="t1" alignment="p1" content="c3[6:32]"/>
    </tier>
  </igt>
</xigt-corpus>
```

Fig. 14 Example of the Xigt XML format with an extension for stand-off alignment to a clean textual representation

further we align the entire line as one string instead of segmenting it into separate gloss grams.[26]

Whether we get fine-grained alignments from clean data or coarse-grained alignments from noisy data, the alignments are all generated automatically and are not vetted by a human. Because of this situation, we set the corpus-

[26] The glosses line on its own is fairly clean, so we could alternatively segment it and align each gloss to the phrase in a floating alignment, but we don't do that here for the sake of illustrating coarse-grained annotation.

```
<?xml version="1.0" encoding="utf-8"?>
<xigt-corpus alignment-method="auto">
  <metadata type="xigt-meta">
    <meta type="language" name="turkish" iso-639-3="tur"
          tiers="phrases words"/>
  </metadata>
  <igt id="i1" doc-id="3645" line-range="10753 10756"
       line-types="L+CR L+CR G T">
    <tier type="odin-raw" id="o">
      <item id="o1" line="10753" tag="L+CR"
          >  21)   O     ben-i     si i o r</item>
      <item id="o2" line="10754" tag="L+CR"
          >                         ky</item>
      <item id="o3" line="10755" tag="G"
          >        He/It me-ACC bores</item>
      <item id="o4" line="10756" tag="T"
          >        'He/It bores/frustrates me'</item>
    </tier>
    <tier type="odin-clean" id="c">
      <item id="c1" line="10753 10754" tag="L+CR"
          >O     ben-i     sikyi o r</item>
      <item id="c2" line="10755" tag="G"
          >He/It me-ACC bores</item>
      <item id="c3" line="10756" tag="T"
          >He/It bores/frustrates me</item>
    <tier type="phrases" id="p" content="c">
      <item id="p1" content="c1[0:25]"/>
    </tier>
    <tier type="glosses" id="g" alignment="p" content="c">
      <item id="g1" alignment="p1" content="c2[0:18]"/>
    </tier>
    <tier type="translations" id="t" alignment="p" content="c">
      <item id="t1" alignment="p1" content="c3[0:25]"/>
    </tier>
  </igt>
</xigt-corpus>
```

Fig. 15 Example of the Xigt XML format with an extension for stand-off alignment to a noisy textual representation

level attribute `alignment-method` to `auto` so users of the corpus can be aware of the quality of the alignments. Other possible values for this attribute are: `gold` for manually-created annotations, and `vetted` for automatically created annotations that have been checked by a human.

6.3 Statistics for the Basic Corpus

The first step of creating the Xigt-formatted ODIN corpus is just conversion from the textual corpus, so here we will present some statistics of the results. The basic corpus has the `odin-raw` and `odin-clean` tiers; coarse, line-based alignments between `phrases`, `glosses`, and `translations` tiers, where they exist; and basic metadata for the language and source document of each IGT. Fine-grained alignments from tokenization of the `phrases` and `glosses` tiers will be distributed as an extension to the basic corpus, as will the enriched tiers for syntax trees and bilingual alignments (described in §6.4). The basic corpus is a truer representation[27] of the original ODIN data than it would be with the advanced tiers, because the advanced tiers introduce information not explicitly present in the textual corpus, and moreover because the advanced tiers, being automatically created, are more likely to have errors than the coarse-grained alignments.

In the basic corpus, there are 157,144 IGT instances representing 1,493 languages,[28] and 2,025 source documents. Of the 157,144 instances, 146,128 have a `phrases` tier created from the language ("L") line, 138,547 have a `glosses` tier created from the glosses ("G") line, and 126,253 have a `translations` tier created from the translation ("T") line. Table 2 shows the distribution of IGT instances by the tier types that were present. Each row represents a mutually exclusive set; e.g. the first row represents instances that have an "L" line, but no "G" or "T" lines. Other line types, such as metadata lines, are not considered. The last row, "Other", represents items that either had none of "L", "G", or "T" lines, or had them in a form that we cannot currently detect, such as when two line types share a single line (e.g. "L"/"T" lines are common when an author provides a small inline glossary).

IGT instance count	(%)	Tiers present
749	0.5%	L
611	0.4%	G
155	0.1%	T
19750	12.6%	LG
7912	5.0%	LT
469	0.3%	GT
117717	74.9%	LGT
9751	6.2%	Other
157114	100.0%	Total

Table 2 Counts of IGT instances according to the tiers present.

6.4 Enriching IGT

Above we have described how we are encoding ODIN IGT into Xigt, but for the AGGREGATION project we also need to process the IGT and enrich them with bilingual word alignments and parse trees. This work is ongoing, and the decisions of how to best encode such information are left to future work, but we present here possible solutions based on the capabilities of Xigt.

[27] But not a *perfect* representation, because even the basic corpus can be improved with better cleaning steps and more accurate extraction of complex IGTs, such as those with line wrapping or inline alternations.

[28] The number of languages is calculated by the assigned ISO-639-3 code.

For bilingual word alignments, we need a way to do many-to-many alignments, as it may be possible for multiple translation words to map to a single original-language word, vice-versa, or even multiple translation words to multiple original words. We accomplish this with a tier that utilizes two alignment expressions. As the alignment itself is the annotation, the items on this tier do not need content. For the AGGREGATION project, we actually align translation words to gloss tokens rather than to original words, as the gloss tokens share a similar form to translation words and thus make a better (i.e. easier) target for an automatic aligner. Moreover, they share the same word order as the original words, so it's possible to project the alignments. Therefore, we obtain the raw alignments by extracting the gloss line and translation line from an IGT and processing them with an external word aligner, then encode the results into Xigt in a `bilingual-alignments` tier with the `source` alignment expression aligned to the translation tier and the `target` alignment expression to the glosses tier. An example is given in Figure 16 with data taken from Figure 14. For clarity, we only show the relevant tiers and we've resolved the `content` alignments. We see that the aligner covered most words, but some others (*made* and *the*) were left unaligned (a more sophisticated aligner might relate *made* to the 'Caus' gram in the glosses tier).

```
...
<tier type="glosses" id="g" alignment="w">
  <item id="g1" alignment="w1">I-Nom</item
  <item id="g2" alignment="w2">child-Dat</item
  <item id="g3" alignment="w3">rice-Acc</item>
  <item id="g4" alignment="w4">eat-Caus-Pst-Dec</item>
</tier>
<tier type="translations" id="t" alignment="p">
  <item id="t1" alignment="p1">I made the child eat rice.</item>
</tier>
<tier type="bilingual-alignments" id="a" source="t" target="g">
  <item id="a1" source="t1[0:1]" target="g1"/>
  <item id="a2" source="t1[11:16]" target="g2"/>
  <item id="a3" source="t1[17:20]" target="g4"/>
  <item id="a4" source="t1[21:25]" target="g3"/>
</tier>
...
```

Fig. 16 Example of word-alignment with Xigt

Syntax trees require entities not present in the original IGT, namely nodes on the parse tree, so this extension also requires a second alignment expression. To get the raw parse tree, we extract the translation line from the IGT and process it with an external parser, then encode the results into Xigt. There are some precedents for XML-based encodings of syntax trees (Mengel and Lezius, 2000; Brants et al, 2002), and one possibility for us is to integrate their basic structure with our alignment expressions. An example of such an encoding is given in Figure 18, which represents the tree in Figure 17. Again, for clarity we only show the relevant tiers, and resolve `content` alignments. The `syntax` tier has two alignment expressions, `lex` and `node`, which may not both appear on the same item. Items with `lex` select the lexical unit(s) at the leaves of the tree. Items with `node` represent non-terminal (i.e. phrasal) nodes in the tree, and select items (either `lex` or `node`) from the same tier.[29] The item contents are the node labels of the syntax tree. Encoding syntax trees in this way allows us to easily give only partial tree structures, which would be useful when creators of IGT annotate syntactic structure for only a portion of the sentence, or when an automatic parser fails to give a full representation. It can also handle packed-representations of multiple parses (e.g. with an item attribute that specifies for which parses a node is used) for when there is syntactic ambiguity. Furthermore, it is straightforward to refer to any node in the tree, since they can all have identifiers, thus allowing annotation on top of the syntax tree.

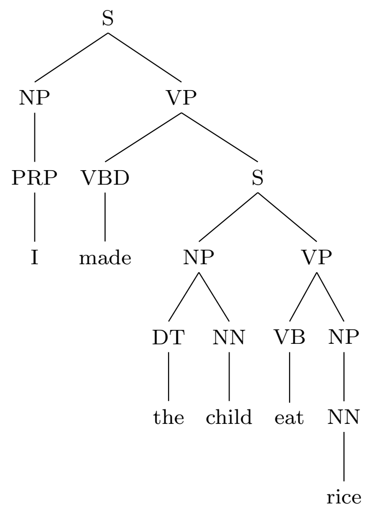

Fig. 17 Syntax tree for sentence from Figure 14

6.5 Summary

In this section we have described our process for importing data from ODIN to Xigt, including the Xigt extensions required for this dataset. We also described our tentative plans for encoding bilingual word alignments and syntax trees. The ODIN use case illustrates how Xigt satisfies our design desiderata, including especially *Stand-Off*, *Incrementality*, *Extensibility* and *Complex Alignments.*

[29] The extension may need some validation code to ensure there are no cycles.

```
...
<tier type="translations" id="t" alignment="p">
  <item id="t1" alignment="p1">I made the child eat rice.</item>
</tier>
<tier type="syntax" id="x" lex="t" node="x">
  <item id="x1" lex="t1[0:1]">PRP</item>
  <item id="x2" lex="t1[2:6]">VBD</item>
  <item id="x3" lex="t1[7:10]">DT</item>
  <item id="x4" lex="t1[11:16]">NN</item>
  <item id="x5" lex="t1[17:20]">VB</item>
  <item id="x6" lex="t1[21:25]">NN</item>
  <item id="x7" node="x1">NP</item>
  <item id="x8" node="x3,x4">NP</item>
  <item id="x9" node="x6">NP</item>
  <item id="x10" node="x5,x9">VP</item>
  <item id="x11" node="x8,x10">S</item>
  <item id="x12" node="x2,x11">VP</item>
  <item id="x13" node="x7,x12">S</item>
</tier>
...
```

Fig. 18 Example of representing the syntax tree in Figure 17 in Xigt

7 Conclusion and Future Work

This paper has presented Xigt, an extensible representation for interlinear glossed text. The schema, code, examples, and documentation are all open-source and available from the project website:

`http://depts.washington.edu/uwcl/xigt`

Our motivation for creating Xigt stems from work repurposing IGT data collected from various sources for NLP applications, including specifically applications of NLP to language documentation. We reviewed existing formats before creating Xigt but found that they did not answer to our specific needs. In particular, the formats that came closest were either not optimized for efficient processing or too general. We defined Xigt with automatic processing as its primary purpose, and such that it answers to our design desiderata:

1. *Stand-off*: Through ids and references (including alignment expressions), it can be as stand-off as one needs.
2. *Incrementality*: There is no fixed nesting of types, which allows flexibility in the presence and order of tiers of annotation. Moreover, as shown in §6, enriched IGT do not structurally change the core IGT—that is, an extension generally does not alter the original IGT, but monotonically adds to it.
3. *Extensibility*: The core of Xigt is designed to be simple, and to support extensions which are also simple.
4. *Complex alignments*: Xigt's alignment expressions allow for one-to-many relationships, and floating alignments allow for many-to-one relationships.

5. *ID-reference annotations*: IDs and references allow complex relationships in a flat structure (i.e. a list of tiers and items), and make it easy to add new information.
6. *Applicability to Automatic Processing*: Our data model groups all relevant data for an IGT under a single element, which helps with streaming or piped processing, such that a processor can operate on each IGT without having to read the whole corpus first.
7. *Predictable representation*: We use a relatively flat structure of elements and fixed element names following the constraints of the data model to help ensure forward-compatibility with processors. Variation in the kinds of data is represented as subtypes of these elements.

We present as a test case the import of ODIN data into Xigt. The ODIN data is of interest as a test case because it encompasses wide variation in the quality and quantity of annotation as well as additional sources of noise (PDF extraction and OCR errors). This test case illustrates how Xigt allows us to flexibly represent as much information as is available on an item-by-item basis.

Work is currently underway to import Abui (a Papuan language of Eastern Indonesia) data from Toolbox for the purpose of automatically discovering phenomena and linguistic patterns. František Kratochvíl, the linguist who created and maintains the Toolbox corpus, is closely supervising our efforts at conversion, and can verify when our methods are accurate. Once we have successfully converted the Abui Toolbox data into Xigt, the next step (for Xigt's purposes) is to make a language-independent conversion routine for Toolbox data.

Our future goals for Xigt itself include adding to the ODIN extension to represent parse trees, developing a translation layer between Xigt and LAF/-GrAF, and further extensions relating to content standards in the general IGT space (e.g. the Leipzig Glossing Rules and OLAC metadata). We are also interested in importing to Xigt from sources beyond ODIN, and in particular from tools that support the creation of IGT including Shoebox/Toolbox and TypeCraft (Beermann and Mihaylov, 2009).

Acknowledgment We would like to thank Glenn Slayden and Ryan Georgi for general discussion; Luís Morgado da Costa for the example Portuguese IGT; and Francis Bond, František Kratochvíl, and anonymous reviewers for helpful feedback.

References

Bakker D, Siewierska A (2007) Another take on the notion subject. In: Hannay M, Steen G (eds) Structural-Functional Studies in English Grammar: In Honour of Lachlan Mackenzie, Studies in Language, vol 84, John Benjamins Publishing Company, pp 141–158

Baldwin T, Beavers J, Bender EM, Flickinger D, Kim A, Oepen S (2005) Beauty and the beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar — and the corpus. In: Kepser S, Reis M (eds) Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives, Mouton de Gruyter, Berlin, pp 49–69

Beermann D, Mihaylov P (2009) TypeCraft: Linguistic data and knowledge sharing, open access and linguistic methodology, paper presented at the Workshop on Small Tools in Cross-linguistic Research, University of Utrecht. The Netherlands

Bender EM, Flickinger D, Oepen S (2002) The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In: Carroll J, Oostdijk N, Sutcliffe R (eds) Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics, Taipei, Taiwan, pp 8–14

Bender EM, Drellishak S, Fokkens A, Poulson L, Saleem S (2010) Grammar customization. Research on Language & Computation pp 1–50, URL `http://dx.doi.org/10.1007/s11168-010-9070-1`, 10.1007/s11168-010-9070-1

Bender EM, Ghodke S, Baldwin T, Dridan R (2012) From database to treebank: Enhancing hypertext grammars with grammar engineering and treebank search. In: Nordhoff S, Poggeman KLG (eds) Electronic Grammaticography, University of Hawaii Press, Honolulu, pp 179–206

Bender EM, Goodman MW, Crowgey J, Xia F (2013) Towards creating precision grammars from interlinear glossed text: Inferring large-scale typological properties. In: Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities, Association for Computational Linguistics, Sofia, Bulgaria, pp 74–83, URL `http://www.aclweb.org/anthology/W13-2710`

Berglund A, Boag S, Chamberlin D, Fernandez MF, Kay M, Robie J, Siméon J (2007) XML path language (XPath) 2.0. W3C recommendation 23

Bickel B, Comrie B, Haspelmath M (2008) The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses, URL `http://www.eva.mpg.de/lingua/resources/glossing-rules.php`, max Planck Institute for Evolutionary Anthropology and Department of Linguistics, University of Leipzig

Bird S, Liberman M (2001) A formal framework for linguistic annotation. Speech communication 33(1):23–60

Bird S, Day D, Garofolo J, Henderson J, Laprun C, Liberman M (2000) Atlas: A flexible and extensible architecture for linguistic annotation. In: In Proceedings of the Second International Conference on Language Resources and Evaluation. Paris: European Language Resources Association

Brants S, Dipper S, Hansen S, Lezius W, Smith G (2002) The TIGER treebank. In: Proceedings of the Workshop on Treebanks and Linguistic Theories, pp 24–41

Bratt EO (1996) Argument composition and the lexicon: Lexical and periphrastic causatives in Korean. PhD thesis, Stanford University

Brugman H, Russel A (2004) Annotating multi-media/multi-modal resources with ELAN. In: Proceedings of the Fourth International Conference on Language Resources and Evaluation

Cagri I (2005) Minimality and turkish relative clauses. PhD thesis, University of Maryland

Clark J, Murata M (2001) {Relax NG} specification. Tech. rep., The Organization for the Advancement of Structured Information Standards (OASIS)

Georgi R, Xia F, Lewis W (2012) Improving dependency parsing with interlinear glossed text and syntactic projection. In: Proceedings of COLING 2012: Posters, Mumbai, India, pp 371–380

Hughes B, Bird S, Bow C (2003) Encoding and presenting interlinear text using XML technologies. In: Proceedings of the Australasian Language Technology Workshop 2003, Melbourne, Australia, pp 61–69

Ide N, Suderman K (2007) GrAF: A graph-based format for linguistic annotations. In: Proceedings of the Linguistic Annotation Workshop, Prague, pp 1–8

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

Kay M, et al (2007) XSL transformations (XSLT) version 2.0. W3C Recommendation 23

Lewis W, Xia F (2010) Developing odin: A multilingual repository of annotated language data for hundreds of the world's languages. Journal of Literary and Linguistic Computing (LLC) 25(3):303–319

Lewis WD, Xia F (2008) Automatically identifying computationally relevant typological features. In: Proceedings of the Third International Joint Conference on Natural Language Processing, Hyderabad, India, pp 685–690

Maeda K, Bird S (2000) A formal framework for interlinear text. In: Proceedings of the Workshop on Web-Based Language Documentation and Description, URL http://www.ldc.upenn.edu/exploration/expl2000/papers/

Mengel A, Lezius W (2000) An XML-based representation format for syntactically annotated corpora. In: LREC

Oepen S, Flickinger D, Toutanova K, Manning CD (2004) LinGO Redwoods. A rich and dynamic treebank for HPSG. Journal of Research on Language and Computation 2(4):575 – 596

Palmer A, Erk K (2007) IGT-XML: An XML format for interlinearized glossed text. In: Proceedings of the Linguistic Annotation Workshop, Association for Computational Linguistics, Prague, Czech Republic, pp 176–183, URL http://www.aclweb.org/anthology/W/W07/W07-1528

Toews C (2009) The expression of tense and aspect in Shona. Selected Proceedings of the 39th Annual Converence on African Linguistics pp 32–41

Xia F, Lewis W (2009) Applying NLP technologies to the collection and enrichment of language data on the web to aid linguistic research. In: Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education (LaTeCH – SHELT&R 2009), Association for Computational Linguistics, Athens, Greece, pp 51–59, URL http://www.aclweb.org/anthology/W09-0307

Xia F, Lewis W, Goodman MW, Crowgey J, Bender EM (2013) Enriching ODIN. In: Proceedings of LREC 2014, Reykjavik, Iceland, to appear

Zaenen A, Maling J, Thráinsson H (1985) Case and grammatical functions: The icelandic passive. Natural Language & Linguistic Theory 3(4):441–483