Ligt: An LLOD-Native Vocabulary for Representing Interlinear Glossed Text as RDF

Christian Chiarcos
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
http://www.acoli.informatik.uni-frankfurt.de/
chiarcos@informatik.uni-frankfurt.de

Maxim Ionov
Applied Computational Linguistics Lab, Goethe University Frankfurt, Germany
ionov@informatik.uni-frankfurt.de

Abstract

The paper introduces Ligt, a native RDF vocabulary for representing linguistic examples as text with interlinear glosses (IGT) in a linked data formalism. Interlinear glossing is a notation used in various fields of linguistics to provide readers with a way to understand linguistic phenomena and to provide corpus data when documenting endangered languages. This data is usually provided with morpheme-by-morpheme correspondence which is not supported by any established vocabularies for representing linguistic corpora or automated annotations.

Interlinear Glossed Text can be stored and exchanged in several formats specifically designed for the purpose, but these differ in their designs and concepts, and they are tied to particular tools, so the reusability of the annotated data is limited. To improve interoperability and reusability, we propose to convert such glosses to a tool-independent representation well-suited for the Web of Data, i.e., a representation in RDF. Beyond establishing structural (format) interoperability by means of a common data representation, our approach also allows using shared vocabularies and terminology repositories available from the (Linguistic) Linked Open Data cloud.

We describe the core vocabulary and the converters that use this vocabulary to convert IGT in a format of various widely-used tools into RDF. Ultimately, a Linked Data representation will facilitate the accessibility of language data from less-resourced language varieties within the (Linguistic) Linked Open Data cloud, as well as enable novel ways to access and integrate this information with (L)LOD dictionary data and other types of lexical-semantic resources. In a longer perspective, data currently only available through these formats will become more visible and reusable and contribute to the development of a truly multilingual (semantic) web.

2012 ACM Subject Classification Information systems → Graph-based database models; Computing methodologies → Language resources; Computing methodologies → Knowledge representation and reasoning

Keywords and phrases Linguistic Linked Open Data (LLOD), less-resourced languages in the (multilingual) Semantic Web, interlinear glossed text (IGT), data modeling

Digital Object Identifier 10.4230/OASIcs.LDK.2019.3

Funding The research described in this paper was conducted in the project *Linked Open Dictionaries* (LiODi, 2015-2020), funded by the German Ministry for Education and Research (BMBF) as an Early Career Research Group on eHumanities.

1 Background

Interlinear glossed text (IGT) is a notation frequently used in linguistic research and documentation. IGTs combine language utterances with their morphological analysis in order to provide readers with a way to understand linguistic phenomena in languages they do not necessarily know. An important property of IGT examples is that there is an alignment

between morphemes and corresponding grammatical values as in (1)[1]:

(1) Min ter-a ter-gan ezba tau-da bas-kan
I live-ST.IPFV live-PFCT house hill-LOC get_up-PFCT
"I live in the house that is located on the hill" (Tatar, Mishar dialect)

Accessing and analyzing IGT data is vital for a vast amount of linguistic research, especially when dealing with less-resourced languages. However, unlike other types of linguistic resources like corpora or dictionaries, this type of data lacks interoperability and reusability. In practice, linguists rarely use IGTs from outside their research groups. There are two main reasons for this: a conceptual one and a technical one.

From a conceptual point of view, IGT can vary greatly from a researcher to researcher. The Leipzig Glossing Rules [3] define guidelines and best practices for writing glossed examples and texts, but these represent only the basis on which researchers can later build on, introducing more and refined information in their analysis (e.g., a layer with a phonetic transcription or more specific abbreviations for linguistic categories). Another source of variability lies in the list of grammatical categories. In the language documentation community, there is no single inventory for grammatical categories all over the world and their corresponding abbreviations (tags), also, there can be several ways for representing the same category (cf. `AOR` vs. `aor` vs. `aorist`). All these factors make it more difficult to redistribute or reuse IGT data.

From a technical point of view, there is another problem: There is a myriad of ways to encode IGT, each with its advantages and limitations, ranging from printed PDFs to various XML (and pre-XML) formats defined by specific tools such as Toolbox and FLEx. Even in the simple cases it can be difficult to use independently produced IGTs, and it is even more problematic to compare or combine several data sources. One possible way to overcome this problem would be to represent data in a tool- and theory-agnostic format. Several initiatives for this purpose do exist, e.g., TypeCraft or Xigt, based on XML technologies. Another possibility would be to represent this type of data in RDF. Moreover, in order to make it truly interoperable, there should exist a standard vocabulary for this type of data. This paper introduces such a vocabulary, Ligt, an LLOD-native vocabulary for representing Interlinear Glossed Text as RDF data.

The paper is organized as follows. We describe existing approaches to representing IGT data in Section 2. We describe the Ligt core vocabulary in Section 3. Converters between Ligt and several popular formats are described in Section 4. Finally, Section 5 concludes the paper, outlining its main outcomes.

2 Data models for IGT data

2.1 Existing formats

This section describes several widely used formats for storing IGT data, including existing Linked Data representations.

There is a great variation in the ways to store IGT data. Probably, the biggest factor influencing the format is whether the data is stored for research or published alongside the research. In the latter case, it is usually a part of a book or a paper either scattered thorough the text or given as an appendix. This representation is not truly machine-readable, so this format cannot be considered reusable. However, there are initiatives in order to overcome this

[1] An example from the second author's fieldwork materials.

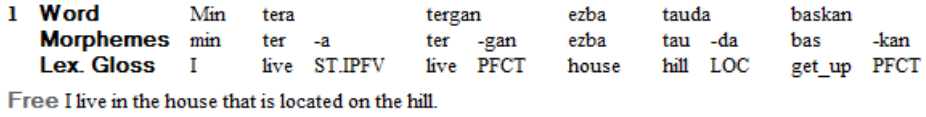


Figure 1 Mishar IGT sample, FLEx print view.

problem by making this data accessible, namely, ODIN, the Online Database of Interlinear Text [7],[2] which was created by parsing scholarly documents on the Web in order to extract such data. For storing the data, ODIN uses Xigt (eXtensible Interlinear Glossed Text).[3] Xigt is an XML-based data model created to simplify the format of IGT in most of the cases, allowing to scale up to accomodate different kinds of annotations. Figure 11 illustrates its XML data model.

IGT data used in research is mostly generated by field linguists working with (native) speakers during their work, so the format depends on what tools do they use. The most widely known applications developed specifically for creating IGT are Toolbox (formerly Shoebox)[4] and FLEx, its successor[5]. They both provide advanced functionality to enter and store IGTs, perform analyses, and build dictionaries. Both have their own advantages, and therefore, both are actively used in the community.

In our previous work, we introduced a shallow RDF representation for both FLEx and Toolbox formats [1]. Below, we briefly describe the respective data models, both in their original XML serialization and in a shallow RDF reconstruction. Other formats do exist, as well, e.g., TypeCraft, and different proposals within the Text Encoding Initiative (TEI).

2.2 RDF reconstruction of FLEx and Toolbox

The FLEx framework stores linguistic data as a set of XML documents: an XML file with all the texts with their markup and a number of auxiliary files: language settings, project settings, etc. The main file consists of a number of `<rt>` elements, each representing a database record. Hierarchy is established by linking records using the attribute `ownerguid` which references the parent record of the element. Records may consist of different elements depending on their `class` attribute.

Figure 1 shows selected glosses in the FLEx graphical user interface and Fig. 2 provides a fragment from its XML representation.

Another way of accessing FLEx data is to export its texts. Unlike the database-like structure of the main XML format, the format for exporting is hierarchical, and its semantics is more clear. FLEx distribution includes a (non-validating) XSD schema that illustrates the basic data structure of these files. Fig. 3 provides a fragment from the XML representation of the same fragment as in Fig. 2.

In [1], we propose a shallow RDF model based on the latter XML representation. Exporting texts from the main project XML leads to information loss which does not allow converting the result back to FLEx projects. Despite that, in this paper we employ this model as a basis for further conversion, while dealing with the main project format is currently under development.

[2] http://depts.washington.edu/uwcl/odin/
[3] https://github.com/xigt/xigt
[4] http://www-01.sil.org/computing/catalog/show_software.asp?id=79
[5] http://fieldworks.sil.org/flex

```
<rt class="StTxtPara"   guid="fa70b76b-5e19-48f6-aa14-9f8d4029ad95"
                        ownerguid="bada85a7-c2cd-47c2-9ad1-be46c3160e5f">
  <Contents>
    <Str>
      <Run ws="tat-Latn-RU-x-mishar">Min tera tergan ezba tauda baskan.</Run>
    </Str>
  </Contents>
  <ParseIsCurrent val="True"/>
  <Segments>
    <objsur guid="a5720126-e1f6-4b19-a5bb-74527f7c57f8" t="o" />
  </Segments>
</rt>
<rt class="Segment" guid="a5720126-e1f6-4b19-a5bb-74527f7c57f8"
                    ownerguid="fa70b76b-5e19-48f6-aa14-9f8d4029ad95">
  <Analyses>
    <objsur guid="248b923a-8d10-49a2-8979-db26d3125ead" t="r" />
    <objsur guid="c2483bb6-6a2d-4fa0-993c-56b538a92b42" t="r" />
    <objsur guid="821aa99b-c8f2-4be0-9634-24797addbb6b" t="r" />
    <objsur guid="c90f7e22-9397-49a7-b727-0cbec202392f" t="r" />
    <objsur guid="fd0fe7c2-b7c0-4b71-bad8-992e1100eb30" t="r" />
    <objsur guid="be7cdaea-8262-4367-a538-b7da11521aa1" t="r" />
    <objsur guid="0534cafe-d520-4548-bc47-962509b77f9f" t="r" />
  </Analyses>
...
```

Figure 2 Mishar IGT sample, FLEx XML.

The RDFS data model that we take as a basis for FLEx data conversion, and the aforementioned data fragment converted with respect to this data model are illustrated in Figs. 5 and 4, respectively.

In earlier work, we have shown that the FLEx conversion can also be applied to Toolbox data [1]. Together with FLEx, Toolbox is the most tool popular for working with IGT data is Toolbox. It is a predecessor of FLEx, although in some aspects it is more powerful than FLEx, so it is still widely used by field linguists. Most notably, it allows creating any number of user-defined "markers" (glossing/annotation layers) such as multiple orthographies or different variants of morphological glossing[6]. Given this, even though there is a process of importing Toolbox data into FLEx, it is not universally possible to do this without information loss.

Toolbox stores its data in an SFM[7] format. It is a text-based format where each line represents a layer defined by its marker at the beginning. Interlinear alignment is achieved by using the precise number of spaces: Each new segment on corresponding lines starts at the same position[8].

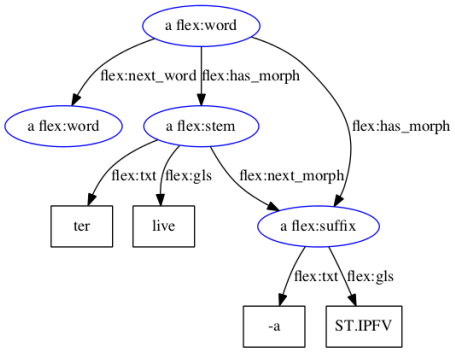
An existing shallow RDF representation for Toolbox resembles the one for FLEx with two key differences:

1. There is no paragraph division in Toolbox data hence `flex:has_phrase` relations can be directly between `flex:interlinear-glosses` and `flex:phrase`.
2. Triples with information regarding toolbox markers are stored in their own namespace since they can differ from the FLEx markers.

[6] The new version of FLEx, which is now available as beta, has similar functionality. However, the current stable version is still widely used.
[7] Standard Format Markers
[8] Due to historical pre-Unicode reasons, the position is calculated in the number of bytes, not in the number of characters.

```
<phrases>
  <phrase guid="a5720126-e1f6-4b19-a5bb-74527f7c57f8">
    <item type="segnum" lang="en">1</item>
    <words>
      ...
      <word guid="c2483bb6-6a2d-4fa0-993c-56b538a92b42">
        <item type="txt" lang="tat-Latn-RU-x-mishar">tera</item>
        <morphemes>
          <morph type="stem" guid="11e7849e-3aad-4c58-bdf4-966f783e6904">
            <item type="txt" lang="tat-Latn-RU-x-mishar">ter</item>
            <item type="gls" lang="en">live</item>
          </morph>
          <morph type="suffix" guid="11e7849e-3aad-4c58-bdf4-966f783e6904">
            <item type="txt" lang="tat-Latn-RU-x-mishar">-a</item>
            <item type="gls" lang="en">ST.IPFV</item>
          </morph>
        </morphemes>
      </word>
      ...
    </words>
    <item type="gls" lang="en">I live in the house that is located on the hill.</item>
  </phrase>
</phrases>
```

Figure 3 Mishar IGT sample, FLEx XML export.




```
:c2483bb6-6a2d-4fa0-993c-56b538a92b42
    a flex:word;
    flex:txt "tera"@tat-Latn-RU-x-mishar;
    flex:has_morph  :11e7849e-3aad-4c58-bdf4-966f783e6904_1,
                    :11e7849e-3aad-4c58-bdf4-966f783e6904_2;
    flex:next_word  :821aa99b-c8f2-4be0-9634-24797addbb6b.

:11e7849e-3aad-4c58-bdf4-966f783e6904_1
    a flex:morph, flex:stem;
    flex:txt "ter"@tat-Latn-RU-x-mishar;
    flex:gls "live"@en;
    flex:next_morph :11e7849e-3aad-4c58-bdf4-966f783e6904_2.

:11e7849e-3aad-4c58-bdf4-966f783e6904_2
    a flex:morph, flex:suffix;
    flex:txt "-a"@tat-Latn-RU-x-mishar;
    flex:gls "ST.IPFV"@en.
```

Figure 4 FLEx IGT sample, generated RDF graph.

2.3 RDF reconstruction of Xigt

In order to provide a generalization over existing, tool-specific data models, Ligt is being defined as a generalization over two data models, FLEx/Toolbox data structures and the data model of the Xigt format. With respect to Xigt, we use a shallow RDF reconstruction of the Xigt data model as a basis, akin to FLEx RDF for FLEx and Toolbox.

Xigt differs from tool-specific formats such as FLEx and Toolbox in that it aims to provide a generic data model for IGT data rather than to provide a serialization for an existing tool. The Xigt format was designed from scratch, it was explicitly intended to be easily extensible for different types of annotations, and thus differs greatly from FLEx or Toolbox formats. Xigt was designed as an XML format, and Fig. 6 illustrates the reconstruction of its structure as an RDFS schema:

- The top-level element of a Xigt document is a `xigt-corpus`, which contains `igt` elements that convey the actual annotation.
- An `igt` contains a number `tier` elements, each corresponding to a single layer of annotation. Each `tier` consists of several `items`.
- An `item` can contain text and carry additional attributes that contain the actual annotation, rendered here as datatype properties of the same name, e.g., `tag`.

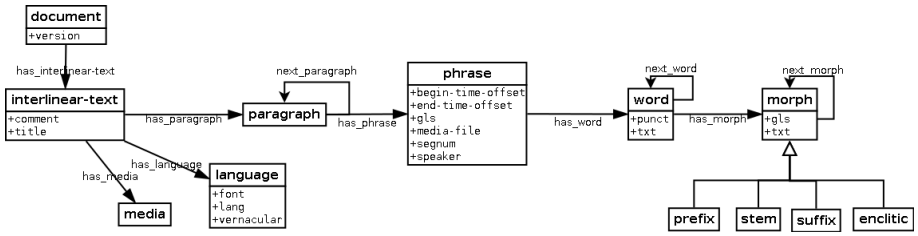


Figure 5 RDF schema fragment for FLEx data.

- Alignment between items is established by *alignment expressions* stored in items' attributes. Those expressions can refer either to one or more items or their parts: `p[1:3]` corresponds to characters 1–3 of the item `p` and `p1+p2[0:2]` corresponds to the full value of item `p1` and characters 0–2 of the item `p2`.
- The sequential order of `igt`, `tier` and `item` is inherent to the XML model, but must be explicated in the RDF rendering. For this purpose, we introduce the property `next`.
- For Xigt XML elements that contain a reference to (the id of) another Xigt XML element, we create an object property of the same name (e.g., `dep` for the annotation of dependency syntax).
- Any `xigt-corpus`, `igt` or `tier` can carry a `metadata` property with a `Metadata` object (corresponding to the `Metadata` element in Xigt/XML).
- The property `meta` assigns a `Metadata` object an XMLLiteral. Normally, this property is not to be used directly, but subproperties are to be created for different types of metadata. These subproperties of `meta` are derived from the `@type` attribute of `meta` elements in Xigt/XML.
- The Xigt RDFS vocabulary does not define subclasses of `igt`, `tier` and `item`, but such subclasses are expected to be defined by different applications, e.g., designated tiers for word segmentation, and morphological segmentation. In Xigt/XML, this is expressed with a `@type` attribute and we expect to derive such more specific subclasses from `@type`.
- In order to ground Xigt/RDF in existing web vocabularies, we define `tier` and `item` as `nif:String`s and postulate a `nif:subString` relation between them [5].
- Xigt elements are identified by a URI. If the Xigt XML element provides an `@id` attribute, this will be adopted as local name and combined with the document/graph URI. Otherwise, URIs are inferred from the structure of the Xigt XML file.

Along with converters for FLEx, Toolbox and other formats, we provide a converter from Xigt to Xigt/RDF as part of our LLODifier suite.[9]

For illustration, we provide a simple sample of the ODIN data base, v.2.3 (from by-doc-id/xigt/10.xml, see appendix for the original XML).

```
:igt10-6 a xigt:igt.
:igt10-6  xigt:metadata :meta1.
:igt10-6 xigt:has_tier :tier_18.

:tier_18 a xigt:odin_tier;
xigt:has_item :item_47, :item_48, :item_49.
```

[9] `https://github.com/acoli-repo/LLODifier`

```
:item_47 a xigt:item;
xigt:line "103";
xigt:tag "L";
xigt:odin_text "Ahmet hizli     ko-uyor-du";
xigt:next :item_48.

:item_48 a xigt:item;
xigt:line "104";
xigt:tag "G";
xigt:odin_text "Ahmet quickly run-PROG-PAST.3sg";
xigt:next :item_49.

:item_49 a xigt:item;
xigt:line "105";
xigt:tag "T";
xigt:odin_text "Ahmet was running quickly.".
```

In more complex examples (omitted here for reasons of space), `item`s are further split into morph(eme)s and the analysis is aligned across different tiers. An advantage of Xigt is that it allows to represent IGTs both in a fine-grained manner (as known from FLEx) and in such a coarse-grained way (as in the ODIN data, adopted here because of difficulties to infer morpheme-level alignment).

Aside from being more scalable with respect to its level of detail, Xigt differs greatly from the FLEx data model described above in the following aspects:

- It lacks any data structures that aggregate IGTs into larger groups such as paragraphs.
- It does not define formal data types for standard components of IGT analysis. Instead, these have to be defined by the data provider (via the `@type` attribute resp. `rdfs:subClassOf`).
- It does provide a complex mechanism for expressing and resolving alignment. In FLEx, this is restricted to substrings.
- It does not provide a vocabulary for metadata properties. Instead, these have to be defined by the data provider (via the `@type` attribute resp. `rdfs:subClassOf`).
- It does provide some properties that exceed the capability of traditional IGTs (e.g., `dep` for depedency syntax).

Like Toolbox (and – to a limited extent – FLEx), Xigt allows to add novel attributes/properties, it is, however, more generic, and allows to include other aspects of linguistic annotation. At the same time, it is underspecified with respect to its concepts: As the comparison with FLEx RDF shows, its data structures are also weaker, in that no vocabulary for essential categories in IGT annotation are provided, most notably words (tokens) and morph(eme)s. We design the Ligt vocabulary as a compromise between both extremes: A vocabulary that provides obligatory IGT data structures (as FLEx), but with the potential for further extensions and underspecification (as Xigt).

3 A native LLOD vocabulary for interlinear glossed text

We motivate Ligt as an abstraction over two application-specific data models, FLEx and Xigt, resp., the RDF vocabularies created for expressing their information in RDF. We see the main contribution of our paper in the formulation of this vocabulary, as a basis for an exchange and publication format for interlinear glossed text in the web of data, and for a

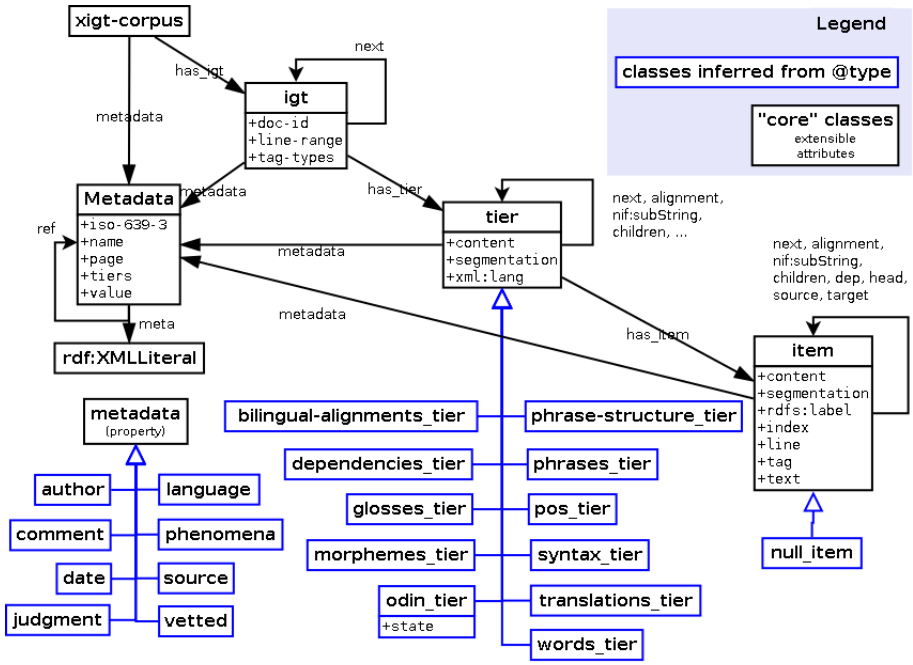


Figure 6 Xigt RDFS data model, also cf. Fig. 11, p. 15 for the underlying XML Schema.

tool chain developed for such data, e.g., converters from classical IGT formats via Ligt to other annotation formats, e.g., tabular formats such as used by ELAN. As part of such a tool chain, we provide a converter suite that generates Ligt from FLEx, Toolbox and Xigt data, and by means of SPARQL, a generic functionality to generate TSV exports from Ligt is already provided by off-the-shelf technology.

In addition, we see an important contribution of this vocabulary as an input to the development of specifications for morpheme-level analyses for W3C vocabularies for lexical data[10] and linguistic annotations.[11]

3.1 Core vocabulary

Ligt vocabulary defines classes and properties to describe the relations between the documents, morphemes and their annotations in texts with an interlinear glossing. This vocabulary is a generalization over two shallow RDF representations introduced in the previous section: a model for data from FLEx or Toolbox, and Xigt RDF model. Other than these, however, it is defined independently from an existing tool chain.

In order to develop an interoperable solution, the base classes are derived from two widely used external vocabularies: Dublin Core [12][12] and the NLP Interchange Format [6, NIF].[13]

[10] Note the development of a morphology module within the OntoLex vocabulary `https://www.w3.org/community/ontolex/wiki/Morphology`.

[11] At the time of writing, morphology is not adequately covered by existing RDF vocabularies for linguistic annotations: NIF [6] and NIF-based vocabularies such as ITS [4] focus on annotations at the level of words or larger, Web Annotation [11] does not provide a designated vocabulary for linguistic annotation at all.

[12] `http://purl.org/dc/terms/`

[13] `http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core/nif-core.html`

`ligt:Document`. A document is a subclass of `dc:Dataset` that represents a collection of interlinear glossed texts. We formulate no constraints on the nature of documents, a document may be the electronic edition of a set of coherent texts, but also an unstructured collection of isolated examples. A `ligt:Document` is equivalent to a `flex:document`, and closely resembles `xigt:corpus`.

`ligt:has_text`. A document must have at least one `ligt:has_text` property that points to a `ligt:Segment`, e.g., an object of type `ligt:InterlinearText`. The property is closely related with `flex:has_interlinear_text`.

`ligt:Segment`. A segment is a `nif:String` that is an abstraction over (interlinear) text, paragraph and utterance. Segments that contain each other are connected by `nif:subString`.

`ligt:InterlinearText`. An interlinear text is a coherent sequence of interlinear glosses, and defined as a `dc:Text` and equivalent with `flex:interlinear-text`. There is no exact pendant of `ligt:InterlinearText` in Xigt. `ligt:InterlinearText` is equivalent to a `ligt:Segment` without another `ligt:Segment` that it is a `nif:subString` of.

`ligt:Paragraph`. A paragraph is a `nif:Paragraph` within a `ligt:InterliearText` that groups together multiple utterances or other segments. It corresponds to `flex:paragraph`, no pendant in Xigt. Paragraph is equivalent with a `ligt:Segment` that is neither `ligt:InterlinearText` nor `ligt:Utterance`.

`ligt:Utterance`. An utterance is a coherent, consecutive sequence of words, as typically produced by a single speaker in a communication situation. The notion of utterance is closely related to a `nif:Sentence`, but we do not require utterances to be sentential in a syntactic sense. We define an utterance as a `ligt:Segment` without further `ligt:Segments` as `nif:subString`. Utterance is equivalent to `flex:phrase`. There is no exact pendant of `xigt:igt` in Ligt, but every `xigt:igt` is both a `ligt:Utterance` and a `ligt:InterlinearText`.

`ligt:hasTier`. This property assigns an utterance a tier that contains its annotations. Corresponds to `flex:has_tier`.

`ligt:Tier`. A tier is a set of annotations that share the same characteristics, and in particular, the same segmentation. Tier corresponds to `xigt:tier`, there is no exact equivalent in FLEx, as FLEx considers tier definitions as being inherent in the notions of `flex:phrase`, `flex:word` and `flex:morph`. Based on FLEx data structures, two subclasses of tier are provided:

`ligt:WordTier`. A tier adopting a segmentation into words (i.e., `flex:words`).

`ligt:MorphTier`. A tier adopting a segmentation into morph(eme)s (i.e., `flex:morphs`).

Note that unlike FLEx, Ligt does not posit a uniqueness constraint on these tiers, but instead supports, for example, to have multiple tiers for morphs at different granularity. Ligt also permits to provide application-specific tiers (as currently by Xigt XML `@type` attributes).

We define a tier as a `nif:String` and its items as the corresponding `nif:subStrings`.

`ligt:item`. Property assigning a `ligt:Tier` a `ligt:Item`. Corresponds to `xigt:has_item`. As both `ligt:Tier` and `ligt:Item` are defined as `nif:Strings`, this property is defined as a subproperty of `nif:subString`.

`ligt:Item`. Abstract class representing elements of a `ligt:Tier`, representing the unit of annotation in an IGT. Equivalent to `xigt:item`, and likewise defined as a subclass of `nif:String`. It is possible to provide application-specific subclasses (as by `@type` in Xigt).

Two following pre-defined subclasses are provided:

`ligt:Word`. A grammatical or orthographic word as the basis for further annotation, equivalent with `nif:Word`. A `ligt:WordTier` is defined as a `ligt:Tier` for which every `item` is a `ligt:Word`. Roughly equivalent with `flex:word`, but note that Ligt (unlike FLEx) does not prohibit concurrent word segmentations of the same utterance.

`ligt:Morph`. We define a morph as a `nif:String` that corresponds to the smallest unit of grammatical analysis applicable to a given word. A `ligt:MorphTier` is defined as a `ligt:Tier` for which every `item` is a `ligt:Morph`. Roughly equivalent with `flex:morph`, but note that Ligt (unlike FLEx) does not prohibit concurrent word segmentations of the same utterance.

An item can be a `nif:subString` of another item at another tier; this is the preferred way to express that a `ligt:Morph` is contained in a `ligt:Word` (cf. `flex:has_morph`).

`ligt:next`. Presents the sequential order of items, corresponds to `xigt:next` and `flex:next_word` and `flex:next_morph`.

Ligt is grounded in the generalization over (the RDF vocabularies inferred for) FLEx and Xigt, but the concept of tiers is exclusive to Xigt, so there is no straightforward way to generalize this concept for both representations. In order to do this, we introduced a base class `ligt:Tier` and two subclasses: `ligt:WordTier` and `ligt:MorphTier` which should correspond to sequences of words and morphs, respectively. Tiers in Ligt must consist of elements on the same level of granularity hence we merge Xigt tiers with identical segmentation. Xigt has been designed for reversible IGT parsing. This means that it provides a standoff mechanism that refer to segments and annotation values rather than providing them. In Xigt RDF, these are resolved, but xigt:content and xigt:alignment are preserved. In the generalization, these are no longer necessary. They should not be deleted, though, as they cannot be easily reproduced. But they provide Xigt-specific information and do not need to be represented in the overarching model.

Both `ligt:Word` and `ligt:Morph` are subclasses of `ligt:Item` and are objects of a property `ligt:item` for the word and morph tiers, respectively. Finally, for compatibility with FLEx, we introduce subclasses of `ligt:Morph` for representing prefixes, suffixes, stems and enclitics.

The data model for text representation in Ligt is illustrated in Fig. 7.

3.2 Metadata

Every concept identified above can be subject to metadata annotations. In order to provide a consistent domain definition for such properties, we introduce `ligt:Element` as an abstract superclass over `ligt:Document`, `ligt:Segment`, `ligt:Tier` and `ligt:Item`.

FLEx metadata is represented by (a fixed set of) simple properties (`flex:version`, `flex:comment`, `flex:title`, `flex:has_media`, `flex:has_language`, etc.) for which no generalization is provided. In opposition to that, Xigt metadata is modelled by means of reification, with a `xigt:Metadata` object mediating between metadata properties and its target.[14] In Ligt, we thus support both mechanisms, but we do not prescribe any specific metadata vocabulary. Instead, any metadata property must be a subproperty of

[14] In Xigt XML, metadata is represented by the container element `metadata` that groups together several `meta` statements. As the `metadata` element can carry its own XML attributes, it has to be rendered as reification in Xigt RDF.

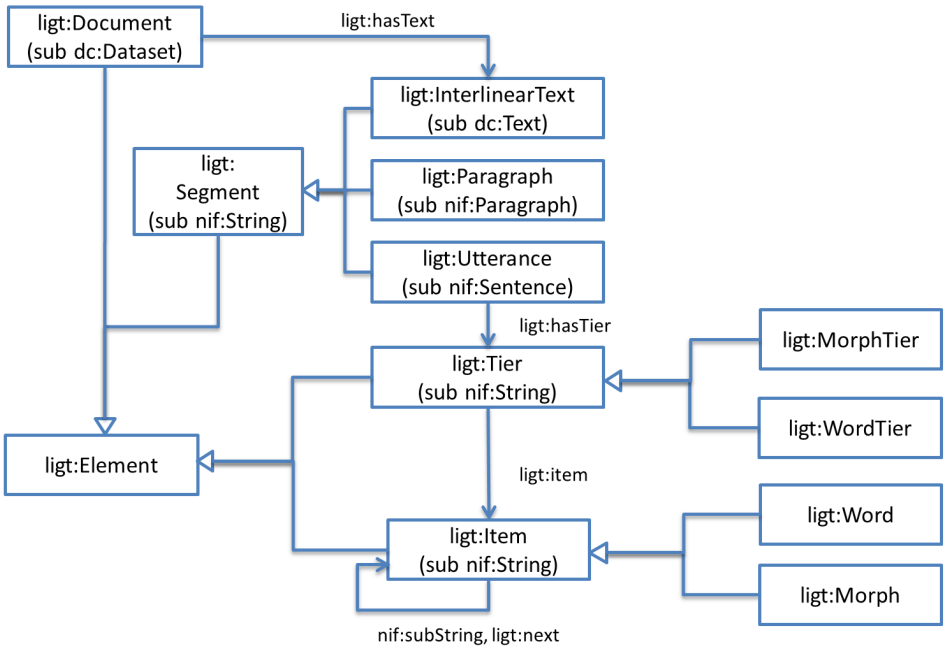


Figure 7 Ligt data model, excluding metadata.

`ligt:metadata`, and any metadata object must be an instance of `ligt:Metadata`. As for the reified representation of metadata, we follow the Web Annotation data model [11].

`ligt:metadata`. Abstract datatype property that assigns a `ligt:Element` a literal value. Superproperty of `flex:version`, etc.
`ligt:Metadata`. Subclass of `oa:Annotation` that represents the reification of `ligt:metadata`.
`ligt:metaTarget`. Subproperty of `oa:hasTarget` that points from a metadata object to the `ligt:Element` that the metadata refers to. Corresponds to the inverse of `xigt:metadata`.
`ligt:metaBody`. Subproperty of `oa:hasBody` that connects a `ligt:Metadata` object with the literal value that contains the metadata. Superproperty of `xigt:iso-693-3`, etc.
`ligt:refBody`. Subproperty of `oa:hasBody` that connects a `ligt:Metadata` object with another (metadata) object. Corresponds to `xigt:ref`.

The Xigt data model allows complex metadata attached to any `corpus`, `igt` or `tier` element. It can be both simple values like language, source or date and complex structures. In the shallow RDF representation this was modeled with reification, where multiple attributes for the same type of metadata are represented as a collection. This approach is powerful, but does not make much sense for atomic metadata properties from FLEx data model. In order not to overcomplicate the model, we decided to use both RDF reification to express the complex Xigt metadata and the more transparently structured FLex metadata. This will keep the model simple but retain its expressivity.

In order to be able to link metadata to elements on different level, we define a top-level concept `ligt:Element`, which is the domain of `ligt:metadata` property. Top-level class `ligt:Document` is defined as its subclass. Atomic metadata elements should be subclasses of `ligt:annotation` whereas complex metadata should be a subclass of `ligt:metadata`. The reified representation is modeled with properties `ligt:metaTarget` and `ligt:metabody`, which are derived from `hasTarget` and `hasBody` properties of the OpenAnnotation vocabulary [10].

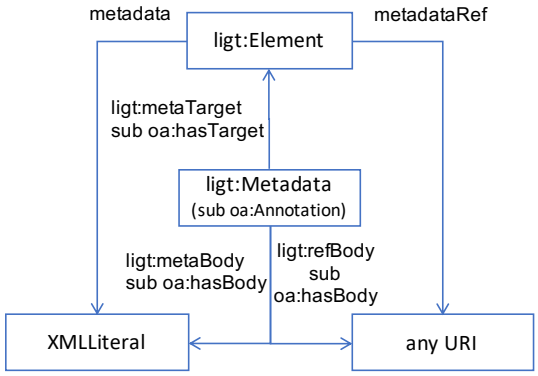


Figure 8 Ligt metadata representation.

The metadata part of the model is illustrated in Fig. 8.

4 Implementation

Here we describe the problems with converters we developed between the shallow RDF representations and Ligt. Mainly, the conversion is performed by a series of SPARQL updates. In the following paragraphs we briefly discuss our decisions and difficulties with the conversion.

Xigt → Ligt. When converting to Ligt, we drop explicit information about segmentation and alignment, since this is already resolved, and we want to keep the data as general as possible. For the same reason we drop tier ordering information (property `xigt:nextTier`), since the ordering of tiers is specific only to Xigt. We also need to convert the metadata information to the OpenAnnotation metadata model.

Xigt ← Ligt. The main problem with the conversion in this direction is that Ligt omits tier ordering information even if the data was originally got converted from Xigt. In order to convert, the order of tiers should be specified manually or left in the default order and then get reordered later by other means.

If the data originally came from FLEx or Toolbox, there may be information about paragraphs or texts, which cannot be expressed in terms of Xigt, which means that this information will be ignored.

FLEx ↔ Ligt. In this conversion, the main difficulty is transforming the data properties into metadata, for instance, language information stored in FLEx data model should become metadata in Ligt representation and vice versa. Another thing is the introduction of fictitious elements within multiple conversions, e.g. putting the text in the paragraph even if there was no paragraphs in the original data when converting to FLEx RDF.

One application of Ligt is to facilitate the integration and the querying of IGT from various sources. The concept is illustrated in Fig. 9. At the moment, we provide converters from FLEx, Toolbox and Xigt to FLEx RDF and Xigt RDF as part of our LLODifier repository,[15] respectively, as well as SPARQL Update scripts to convert that data into Ligt.[16]

[15] `https://github.com/acoli-repo/LLODifier`
[16] `https://github.com/acoli-repo/ligt`

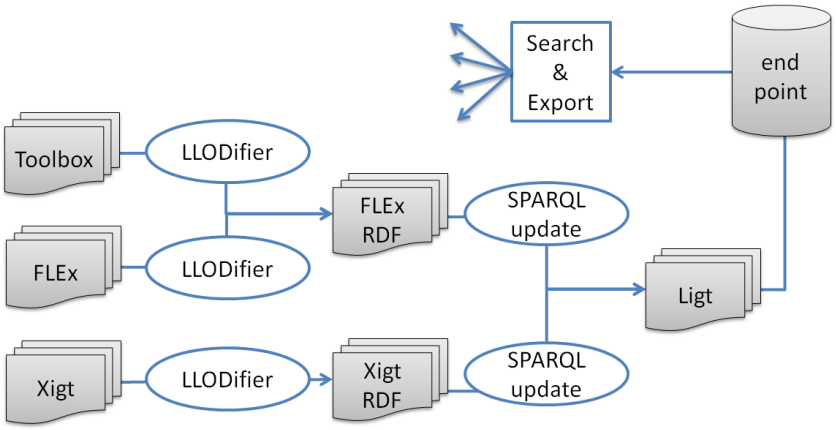


Figure 9 Ligt-based IGT processing workflow.

We have converted the ODIN v.2.3 into Ligt, with a total of 7.5 million triples for 158 007 IGTs for 2 888 language varieties. Interfaces tailored towards the needs of end users (linguists) are still under development.

5 Summary and outlook

In this paper we presented Ligt, the first LLOD-native IGT vocabulary for LLOD-data, based on three formats. This vocabulary is grounded in widely used vocabularies (Dublin Core, NIF and WebAnnotation), but extends them with respect to the coverage of morphology. With Ligt, we aim to achieve the following goals:

Provide a vocabulary for publishing and sharing IGT data via the (Linguistic) Linked Open Data cloud.
Contribute to the extension of W3C vocabularies such as Ontolex-lemon with respect to the coverage of morphology.
Trigger the development of morphology-aware vocabularies for the representation of corpora and linguistic annotations.
Prepare the ground for developing an infrastructure for the integrated querying and processing of IGTs and related linguistic data.

Publishing interlinear glosses as LLOD facilitates their reusability and interoperability, allowing querying several IGT datasets at once, linking them to external resources and more. At the same time, using different shallow data models, each of which inherits conceptual model of the corresponding framework, is not enough to achieve true interoperability. All three frameworks provide slightly different set of functions, and the conceptual model behind their data representation differ greatly. Even though this shallow approach guarantees data structures that are transparent and familiar to their user community, it does not provide the rich semantics of more advanced vocabularies for language resources.

By creating a universal vocabulary for modeling IGT annotations, and creating converters from those three formats to this unified representation should improve interoperability further.

Given this, the main contribution of this paper is a proposal of an RDF-native data model that not only allows to unify IGT data developed under different frameworks to a completely new level, but also allows to generalize to other use cases in linguistics, as well. Beyond

establishing structural (format) interoperability by means of a common data representation, our approach also allows to make use of shared vocabularies and terminology repositories available from the (Linguistic) Linked Open Data cloud, e.g., for representing language varieties [9], linguistic phenomena [2], or lexical information [8].

References

1 Christian Chiarcos, Maxim Ionov, Monika Rind-Pawlowski, Christian Fäth, Jesse Wichers Schreur, and Irina Nevskaya. LLODifying linguistic glosses. In *Proceedings of Language, Data and Knowledge (LDK-2017)*, Galway, Ireland, June 2017.

2 Christian Chiarcos and Maria Sukhareva. OLiA - Ontologies of Linguistic Annotation. *Semantic Web Journal*, 518:379–386, 2015.

3 Bernard Comrie, Martin Haspelmath, and Balthasar Bickel. The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses. `https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf`, 2008.

4 David Filip, Shaun McCance, Dave Lewis, Christian Lieske, Arle Lommel, Jirka Kosek, and Felix Sasaki. Internationalization Tag Set (ITS) Version 2.0. Technical report, W3C Recommendation, 2013.

5 Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP using Linked Data. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013. URL: `http://svn.aksw.org/papers/2013/ISWC_NIF/public.pdf`.

6 Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. Integrating NLP using Linked Data. In *Proc. 12th International Semantic Web Conference, 21-25 October 2013*, Sydney, Australia, 2013. also see `http://persistence.uni-leipzig.org/nlp2rdf/`.

7 William D. Lewis. ODIN: A model for adapting and enriching legacy infrastructure. In *Second International Conference on e-Science and Grid Technologies (e-Science 2006), 4-6 December 2006, Amsterdam, The Netherlands*, page 137. IEEE Computer Society, 2006. `doi:10.1109/E-SCIENCE.2006.106`.

8 John P. McCrae, Julia Bosque-Gil, Jorge Gracia, Paul Buitelaar, and Philipp Cimiano. The Ontolex-Lemon model: development and applications. In *Proceedings of eLex 2017 conference*, pages 19–21, 2017.

9 Sebastian Nordhoff and Harald Hammarström. Glottolog/Langdoc: Defining dialects, languages, and language families as collections of resources. In *First International Workshop on Linked Science 2011-In conjunction with the International Semantic Web Conference (ISWC 2011)*, 2011.

10 Robert Sanderson, Paolo Ciccarese, and Herbert Van de Sompel. Open Annotation Data Model. Technical report, W3C Community Draft, 08 February 2013, 2013.

11 Robert Sanderson, Paolo Ciccarese, and Benjamin Young. Web Annotation Data Model. Technical report, W3C Recommendation, 2017.

12 Stuart Weibel, John Kunze, Carl Lagoze, and Misha Wolf. RFC 2413 - Dublin Core metadata for resource discovery. URL `http://www.ietf.org/rfc/rfc2413.txt` (July 31, 2012), September 1998. Network Working Group.

A Xigt XML data

In the paper, we presented Xigt RDF as one starting point for the development of Ligt. In order to relate this to actually available Xigt data, the interested reader may be interested in the original XML schema and sample data, provided here in Figs. 11 and 10, respectively. Additional documentation can be found under `https://github.com/xigt/xigt/wiki/DataModel`.

```
<igt id="igt10-6" doc-id="10" line-range="103-105" tag-types="L G T">
   <metadata>
     <meta id="meta1">
       <dc:subject olac:code="tur" xsi:type="olac:language">Turkish</dc:subject>
       <dc:language olac:code="en" xsi:type="olac:language">English</dc:language>
     </meta>
   </metadata>
   <tier id="n" type="odin" alignment="c" state="normalized">
     <item id="n1" alignment="c1" line="103" tag="L">Ahmet hizli     ko-uyor-du</item>
     <item id="n2" alignment="c2" line="104" tag="G">Ahmet quickly run-PROG-PAST.3sg</item>
     <item id="n3" alignment="c3" line="105" tag="T">Ahmet was running quickly.</item>
   </tier>
</igt>
```

Figure 10 Xigt XML sample data, ODIN v. 2.3, file by-doc-id/xigt/10.xml.

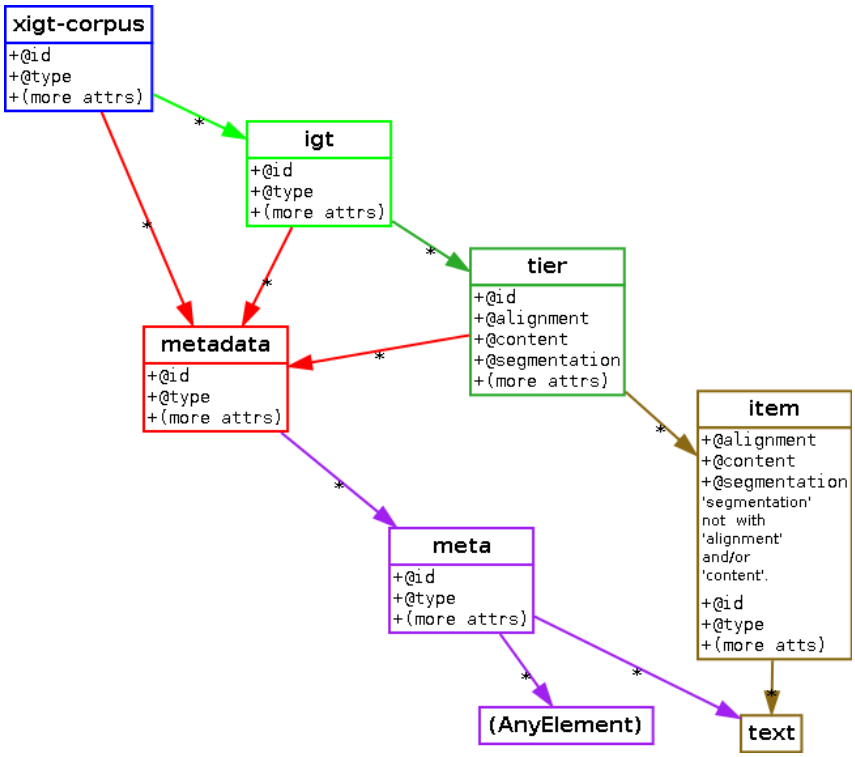


Figure 11 Xigt XML Schema.