Ligt: Towards an Ecosystem for Managing Interlinear Glossed Texts with Linguistic Linked Data

Maxim Ionov
University of Zaragoza, Spain
`mionov@unizar.es`

Abstract

Ligt is an RDF vocabulary developed for representing Interlinear Glossed Text, a common representation of language material used in particular in field linguistics and linguistic typology. In this paper, we look at its current status and different aspects of its adoption. More specifically, we explore the questions of data conversion, storage, and exploitation.

We present *ligttools*, a set of newly developed converters, report on a series of experiments regarding querying Ligt datasets, and analyse the performance with various infrastructure configurations.

1 Background

Interlinear glossed text (IGT) is a notation commonly used to represent language examples in descriptive and typological linguistics. It is designed to provide an intuitive way of showing language material so that it could be understood without needing to know that language. IGT data may consist of any number of layers added under the original text (hence *interlinear*): word-by-word translation, grammatical meaning of morphemes, transliteration, free translation, etc. Some layers have morpheme-by-morpheme alignment between each other, e.g. morpheme segmentation and grammatical meaning of morphemes. Consider the following example from the Adyghe language:[1]

(1)

adəgjejə-m	jə-q^{w}əŝhe-xe-r,
Adyghea-OBL	POSS-mountain-PL-ABS
jə-psəχ^{w}e-čer-xe-r	daxe-x
POSS-river-tumultuous-PL-ABS	beautiful-PL

‘The mountains of Adyghea, its tumultuous rivers are beautiful.’

This example consists of three layers: morphological segmentation, glosses aligned with the transcription layer, and free translation. Often, baseline (i.e. unsegmented source text) and its transcription are included as well. Leipzig Glossing Rules (Comrie et al., 2008) provide the set of guidelines and recommended glosses for common grammatical categories (e.g. PL), however it is a short list and it covers only a small subset of the grammatical categories. Generally, datasets and published works provide a list of abbreviations used for glossing.

The variability of this representation grants a level of flexibility that makes it applicable across disciplines and theoretical frameworks. This, however, also hinders its interoperability given that two different authors might use different ways to encode the same grammatical category.[2] Another, more technical hurdle is the large amount of different formats in which IGT can be represented, ranging from non-unicode plain text and XML to relational databases. All these factors make it more difficult to redistribute and reuse the data, or combine and compare several data sources. A solution to this would be to use an interoperable representation to which all the data sources can be converted in a lossless way. An obvious contender for such a representation is RDF.

In our previous research, we introduced Ligt, an RDF-native vocabulary for representing IGT data (Chiarcos and Ionov, 2019), a generalisation over commonly used formats at the time, namely ToolBox, FLEx and Xigt.[3] Later, we showed the applicability of this vocabulary on a dataset with 76 pidgin and creole languages (Ionov, 2021). Additionally, Nordhoff (2020) and Nordhoff and Krämer (2022) successfully applied it to sev-

[1]Source data and attribution: `https://imtvault.org/b/336/ex/langsci336-38caad062e.htm`.

[2]A common but relatively harmless example is the variation is the encoding of past tense: PST and PAST, which sometimes happens even in the same example: `https://imtvault.org/b/323/ex/langsci323-af787e1cef.htm`.

[3]Description of these formats and their limitations can be found in the paper.

eral hundred languages from endangered language archives and linguistic literature.

In this paper, we take a step forward and focus on the next steps that ensure the vocabulary usability and potential for adoption now that its applicability has been proven. For this, we need to generalise over how LD resources are created. According to a commonly used methodology for publishing multilingual Linked Data (Vila-Suero et al., 2014), the process consists of the following steps:

1. Specification: Analyzing and describing data (data sources and RDF data) characteristics;

2. Modelling: Creating/selecting vocabularies to describe the RDF resources;

3. Generation: Transforming the data sources to RDF;

4. Linking: Connecting the RDF dataset;

5. Publication: Making the dataset available and discoverable on the Web.

To understand which steps should be taken on the way from a vocabulary to a usable ecosystem, we need to put ourselves in the shoes of linguists, archivists and other users who might want to use Ligt and analyse how they could approach each of these steps. Since we know the nature of the data and the modelling, we consider the first two steps solved. This means that the focus of this paper is on the remaining three steps: generation (or **conversion**), **linking** and **publication**.

The linking step has also received some attention in the previous research (Ionov, 2021; Nordhoff and Krämer, 2022), so instead of exploring linking *per se*, we are going to focus on how to employ these links, i.e. the infrastructure and performance for federated queries across several data sources.

The rest of the paper is organised as follows: in Section 2 we give a brief overview of the current data model and present newly developed converters, Section 3 describes a series of case studies on querying different volumes of Ligt data linked to external vocabularies with different infrastructure configurations. Finally, in Section 4 we provide a summary and outline future directions.

2 Ligt: Data model and converters

2.1 Ligt Data Model

Before going further, we first outline the main parts of the Ligt model (Fig. 1).[4]

The central element in any Ligt dataset is `ligt:Document`, a subclass of `dc:Dataset`. Depending on the source material, it can either consist of sets of utterances, i.e. examples from different chapters of a typological database like GramBank[5] (`ligt:InterlinearCollection`) or texts (`ligt:Text`). Both consist of one or more `ligt:Utterance`, which roughly correspond to a sentence or an elicitation.

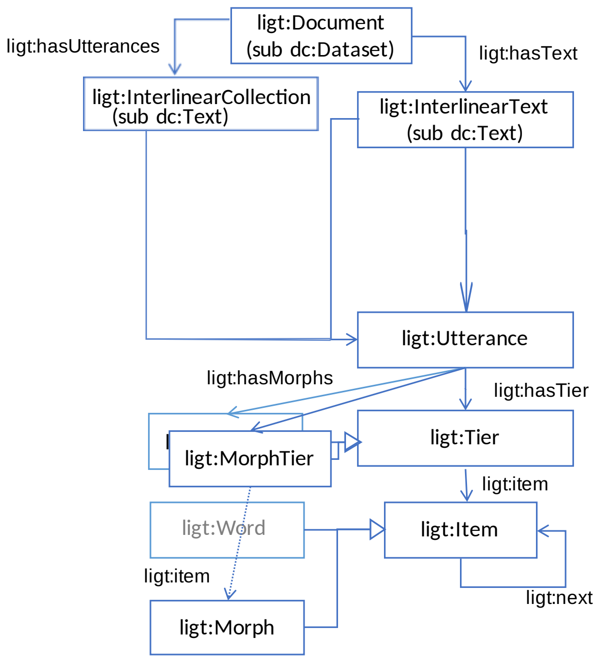

Figure 1: A simplified Ligt data model

Class `ligt:Tier` and its two subclasses `ligt:WordTier` and `ligt:MorphTier` represent annotation layers, i.e. sequences of words and subword elements, respectively. Each tier consists of `ligt:Items` that are connected to each other with the property `ligt:next` (Fig. 2).

The current model has several changes compared to the one described previously in Ionov (2021): Most importantly, the model is decoupled from the NIF vocabulary (Hellmann et al., 2013). The original motivation behind using it was to reuse `nif:String` and `nif:subString` to represent an annotation tier and a single annotations

[4]Full description can be found in the documentation: https://ligt-dev.github.io/ligt.

[5]https://grambank.clld.org/.

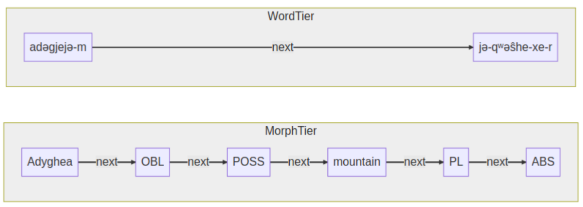

Figure 2: Structure of an utterance

in it, respectively which created a degree of interoperability with NIF-based corpus annotations. However, in practice this proved to be problematic, since it is common in IGT data to have layers with alternative orthographies which cannot be split into substrings equal to the ones presented on other layers, as shown in (2).

(2)

haste	nich	gesehen
hast=du	nicht	ge-seh-en
have=2ps	NEG	PTCP-see-PTCP

Instead, a new property, `ligt:utterance` is introduced, which can link either a text or a set of examples to individual utterances. In the future, this part will likely be integrated with the consolidated linguistic annotation vocabulary currently developed by LD4LT.[6]

The second important change is introduced to add versatility: in addition to the elements in a tier being connected with the `ligt:next` property, `ligt:Tier` is now a subclass or `rdfs:Seq`, which allows data providers to explicitly set the order or elements to make it easier to query and control the order of the elements. Finally, to help keep the data error-prone, the alignment between elements can be set using DCMI properties `dct:hasPart` and `dct:isPartOf`. Both this and the element ordering are introduced solely for convenience and do not provide additional information. For compatibility, this data can be added to existing datasets with a SPARQL UPDATE or a similar mechanism.

2.2 Converters

Now that we gave an overview of the model, we present a new set of converters, a part of a suite *ligttools*, a CLI tool with a Python API.[7] When introducing Ligt for the first time (Chiarcos and Ionov, 2019), we had already provided a prototype converter for some of these formats. While functional, these converters relied on intermediate shallow representations and were not packaged in a user-friendly way. Since the vocabulary developed over time, we decided to create new converters from scratch and make them as easy to use as possible. As a result, the list of supported formats changed according to their usage over time. Most importantly, instead of Xigt, which is no longer under active development, we added CLDF, a format used in the majority of typological databases created in the past decade.

2.2.1 CLDF

CLDF, Cross-Linguistic Data Formats is a set of guidelines and tools aimed at distributing linguistic data in a sustainable and standardised way based on tabular data format (Forkel et al., 2018). In the past years the user base of this standard has grown significantly, and more and more resources: dictionaries, wordlists, typological databases and more has become available for use and download. Thanks to its flexibility and simplicity, and the level of tooling and infrastructure it became a *de facto* standard for releasing linguistic data.

Given its prominence, it is extremely important to have a reliable converter from CLDF data to Ligt and back. Thankfully, due to the tabular nature of the data and the underlying Web-friendly technology, CSV on the Web,[8] accessing and analysing the data is a straightforward process. However, CLDF is designed to be flexible and even though there is a default machine-readable description of an example table,[9] data providers might change the structure of the data, omitting some of the fields. Our converter takes a CLDF metadata file as input and checks if all the necessary data is referenced and can be found.

Another issue with the conversion from CLDF is the fact that text and gloss layers are represented as single strings without any strict internal representation which may lead to data inconsistencies and parsing errors. In cases like this, our converter skips the sentence altogether.

These two issues do not arise when converting back from Ligt to CLDF, however, CLDF limits the number of layers to 4 columns. This is enough in most cases but can be insufficient when converting data with two different orthographies, for example. In these cases we still preserve the data in a separate column, but this column is ignored by most CLDF tools.

[6] `https://github.com/ld4lt/linguistic-annotation`.
[7] `https://github.com/ligt-dev/ligttools`.

[8] `https://github.com/w3c/csvw`.
[9] `https://github.com/cldf/cldf/blob/master/components/examples/ExampleTable-metadata.json`

2.2.2 Toolbox

SIL Toolbox is one of the most widely known applications developed specifically for creating IGT data.[10] Despite co-existing with FLEx, its successor (see below) for many years, it is still widely used by researchers. Toolbox internal file format is Standard Format Marker (SFM), which consists of lines of annotation prepended with one of the markers, e.g. `\tx haste nich gesehen`

One of the reasons why Toolbox is still actively used is that it supports user-defined markers (layers), while its successor does not.[11]

The conversion to and from Toolbox SFM format is straightforward as long as there is a mapping between the Ligt tiers and Toolbox markers. In our previous research we implemented conversion via an intermediate format, however it proved to be cumbersome, so now the conversion is implemented directly.

2.2.3 FLEx

SIL Fieldworks or FLEx is a spiritual successor to Toolbox.[12] It is probably the most widely used software for language documentation. Internally, FLEx stores all the project data as a series of XML files with a list of *records* interconnected via their GUID. It is a tricky format requiring quite a lot of overhead to read and even more to write the data. Another way of accessing FLEx data is via *flextext* files which contain exported texts. Unlike the database-like structure of the main XML format, the format for exporting is hierarchical, and its semantics is more clear. This is a much more common way to extract information from FLEx projects, even though it requires more work from users.

Currently, our converter works with the FLEx exports, which limits it to a one-directional conversion — from FLEx to Ligt, not vice versa.

3 Infrastructure

3.1 Traditional RDF infrastructure

Having converters from various commonly used formats in an easy to use package is a necessary step towards using the vocabulary. However, the next part could also be challenging.

One of the main obstacles for the adoption of RDF-based technologies is an amount of resources and technical skills required to set up the infrastructure (Chiarcos, 2021; Gromann et al., in press, p. 27). The conventional pipeline for using RDF data from the infrastructural point of view is the following:

1. Converting the data
2. Setting up a SPARQL endpoint
3. Uploading the data and keeping it up-to-date
4. Querying the data

These steps put a lot of technical and computational burden on someone who might just want to extract all instances of some grammatical phenomenon from several datasets. Many proposals on how to solve this argue for large infrastructure projects, e.g. Databus[13] or TriplyDB.[14] While this might help in some cases, this definitively is not a panacea, since it creates its own problems: data security, trust, reliance on external services that might unexpectedly cease to exist, among others.

In this section, we look at two steps towards simplifying this process and try to evaluate the trade-offs involved.

3.2 Client-side computation

Possibly the biggest obstacle of the traditional RDF infrastructure is the need to set up and maintain a SPARQL endpoint. Not only it requires technical skills, but also access to a decently powerful server. SPARQL endpoints like Apache Jena Fuseki[15] and Virtuoso[16] use quite a lot of resources in order to execute queries and maintain availability. Empiriacal studies suggest that they correctly execute only a fraction of all queries to large datasets (Saleem et al., 2015). In addition, this setup requires from a user to keep track of the data in an additional place, making sure the version uploaded to the endpoint is up-to-date with the local version.

An alternative to this is to use data dumps (Turtle or any other RDF serialization) and a SPARQL engine, like Apache Jena ARQ[17] or Comunica.[18] This eliminates the need for maintaining a server and managing the data, but increases the overhead

[10] `http://www-01.sil.org/computing/catalog/show_software.asp?id=79`.

[11] Newer versions of FLEx support this to a certain extent.

[12] `http://fieldworks.sil.org/flex`.

[13] `https://databus.dbpedia.org/`

[14] `https://triply.cc/`

[15] `https://jena.apache.org/documentation/fuseki2/`.

[16] `https://vos.openlinksw.com/owiki/wiki/VOS`.

[17] `https://jena.apache.org/documentation/query/index.html`.

[18] `https://comunica.dev/`.

required to load the data in memory and execute queries without indices.

To compare these two approaches, we prepared 3 linguistically motivated queries of different complexity:

Q1: Search for all surface forms with a gloss "woman".

Q2: Search for all examples with a causative morpheme and a past tense marker.

Q3: List all case markers used in a language.

Each query was tested on a low-to-medium-sized dataset (15k sentences, 1M triples) in 2 different scenarios:

- on an Apache Jena Fuseki SPARQL endpoint, and
- locally, using Comunica SPARQL engine.

The execution times queries are given in Table 1.[19]

	Endpoint	Locally
Q1	1.1	3.4
Q2	1.2	5.4
Q3	2.7	4.6

Table 1: Execution time (sec) in Fuseki vs. Comunica.

While generally Comunica executes the queries much slower than the remote endpoint, it performs very differently from Fuseki in Q3, which involves linking a small external dataset. While this takes Fuseki more that twice the time it need to process Q2, Comunica actually executes Q3 faster than Q2. This probably stems from the fact that Comunica is optimised for federated queries and combining data sources. On the other hand, when the query requires to go through a single dataset (or several unconnected ones) and filter it, Fuseki works better since it does not need to load the data in memory for every query and it can benefit from pre-constructed indices.

Based on these results, it seems that local execution without an endpoint makes sense when data is distributed across many small independent sources.

[19]The details about the environments, queries and a more thorough evaluation can be found at `https://github.com/ligt-dev/ldk-2025/blob/main/experiments.md`

3.3 On-the-fly conversion

An even more radical step to reduce the entry cost is to convert the data on demand. The advantage of this approach is that it completely eliminates the danger of data desynchronisation: the source data is the single source of truth. Additionally, this is helpful for when the data source is dynamic or needs to be scraped. Finally, this can be useful when the user does not have rights to save and modify the data — in this case they still can process the data on-the-fly and use it alongside static data.

The comparison in execution time for using converted datasets with Comunica vs. converting data on-the-fly and feeding it data streams is given in Table 2.

	Premade	On-the-fly
Q1	3.4	23.9
Q2	5.4	25.8
Q3	4.6	25.3

Table 2: Execution time (sec) for premade vs. converted on-the-fly.

The execution on-the-fly is predictably much slower than on the pre-made dataset and should not be used often. However, this can be useful in some situations, especially when dealing with trivially small datasets that change often or when the query only needs to be executed once.

4 Summary and Outlook

In this paper, we looked at the current state of Ligt, an RDF vocabulary for representing interlinear glossed text. Looking through the lens of publication of LD resources, we reflected on its current position and the current state of an ecosystem around it.

We presented *ligttools*, a suite of tools for Ligt, including a set of converters for common IGT formats: CLDF, Toolbox and FLEx.

Additionally, we explored two ways to simplify the usual infrastructure required to work with RDF resources: first by removing a SPARQL endpoint and moving the computation to the client, and second by removing static RDF data altogether, replacing it with an on-the-fly conversion. We conclude that while client-side computation provide competitive results while removing most challenging requirements, on-the-fly conversion is warranted only with very small datasets and infrequent queries. Still, there might be a case for it, especially when the source data is subject to change or cannot be easily saved as a whole.

The next step towards the Ligt ecosystem — developing *Ligt Workbench*, a prospective standalone tool for management and searching in local and remote IGT collections. Developing it testing on real-world linguistic problems is the direction of future research.

References

Christian Chiarcos. 2021. Get! Mimetypes! Right! In *3rd Conference on Language, Data and Knowledge (LDK 2021)*, volume 93 of *Open Access Series in Informatics (OASIcs)*, pages 5:1–5:4, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Christian Chiarcos and Maxim Ionov. 2019. Ligt: An LLOD-Native Vocabulary for Representing Interlinear Glossed Text as RDF. In *2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 70 of *Open Access Series in Informatics (OASIcs)*, pages 3:1–3:15, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

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

Robert Forkel, Johann-Mattis List, Simon J. Greenhill, Christoph Rzymski, Sebastian Bank, Michael Cysouw, Harald Hammarström, Martin Haspelmath, Gereon A. Kaiping, and Russell David Gray. 2018. Cross-linguistic data formats, advancing data sharing and re-use in comparative linguistics. *Scientific Data*, 5.

Dagmar Gromann, Elena-Simona Apostol, Christian Chiarcos, Marco Cremaschi, Jorge Gracia, Katerina Gkirtzou, Chaya Liebeskind, Verginica Mititelu, Liudmila Mockiene, Michael Rosner, and 1 others. in press. Multilinguality and LLOD: A Survey Across Linguistic Description Levels. *Semantic Web Journal*.

Sebastian Hellmann, Jens Lehmann, Sören Auer, and Martin Brümmer. 2013. Integrating NLP using linked data. In *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, volume 8219 of *Lecture Notes in Computer Science*, pages 98–113. Springer.

Maxim Ionov. 2021. APiCS-Ligt: Towards Semantic Enrichment of Interlinear Glossed Text. In *3rd Conference on Language, Data and Knowledge (LDK 2021)*, volume 93 of *Open Access Series in Informatics (OASIcs)*, pages 27:1–27:8, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Sebastian Nordhoff. 2020. Modelling and annotating interlinear glossed text from 280 different endangered languages as linked data with LIGT. In *Proceedings of the 14th Linguistic Annotation Workshop*, pages 93–104, Barcelona, Spain. Association for Computational Linguistics.

Sebastian Nordhoff and Thomas Krämer. 2022. IMT-Vault: Extracting and enriching low-resource language interlinear glossed text from grammatical descriptions and typological survey articles. In *Proceedings of the 8th Workshop on Linked Data in Linguistics within the 13th Language Resources and Evaluation Conference*, pages 17–25, Marseille, France. European Language Resources Association.

Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. 2015. Lsq: The linked sparql queries dataset. In *The Semantic Web - ISWC 2015*, pages 261–269, Cham. Springer International Publishing.

Daniel Vila-Suero, Asunción Gómez-Pérez, Elena Montiel-Ponsoda, Jorge Gracia, and Guadalupe Aguado-de Cea. 2014. Publishing linked data on the web: The multilingual dimension. In *Towards the Multilingual Semantic Web: Principles, Methods and Applications*, pages 101–117. Springer.