Flibl: A tool to ease text transfer between ELAN and FLEx

Abstract

Two of the most common software tools in language documentation are ELAN, for transcription, and FieldWorks Language Explorer (FLEx), for interlinearization. Many language documentarians use these tools together, and the transcribed output of ELAN is natural input to FLEx. Despite this, out of the box the two programs are not effectively interoperable. FLEx also does not allow users to display many data structures which are visible in ELAN and necessary for research purposes, such as speaker attributions. Therefore, we created Flibl [flɪbl̩], a software tool that automatically converts between the data formats used by ELAN and FLEx while keeping all ELAN information visible. We describe our research motivations for creating Flibl, how researchers can use it and for what topics, and how the software works on the backend. Readers interested in using Flibl can download it from our stable repository at https://osf.io/kaqr3/?view_only=609a8c7c034d4f5cbf61ad3f0e559cfb.

1. Introduction

Extended samples of connected language, i.e. "texts", are central to language documentation (Foley 2003; Austin 2006; Musgrave & Thieberger 2021). Many documentation projects which collect texts share a common workflow (Bowern 2015; Meakins, Green & Turpin 2018). Documentation workers record texts, then transcribe and translate (or otherwise annotate) them, then add morpheme-level glosses. Later in the project lifecycle, some people will create secondary analyses or language revitalization materials using the glossed texts, or will archive them. This workflow is as old as the field of language documentation, dating at least from the 1910s (Epps, Webster & Woodbury 2017: 49).

But while the text workflow itself is long-lived, *tools* for transcription, translation, and glossing have changed enormously over the last three decades with the advent of digital documentation methods. Researchers of the past transcribed or took dictation on paper; today, they rely on media annotation software, such as Praat (Boersma & Weenink 2022) and ELAN (MPI for Psycholinguistics 2023a; Wittenburg et al. 2006), to create time-aligned transcriptions and other annotations. Likewise, glossing was once done entirely by hand, but today, many documentarians instead use database software, typically FieldWorks Language Explorer or FLEx (Summer Institute of Linguistics 2022), to interlinearize texts. Given the complexity of these tools, interoperability and ability to seamlessly convert data between formats is vital. Nevertheless, interoperability continues to be identified as a sticking point in data workflows (cf. Glenn 2009; Han 2022).

This article, like the software tool which it presents, focuses on the technical interoperability needs of the transcription and glossing steps of the text workflow. Specifically, we describe Flibl, a software package developed to improve data transfer between ELAN – a widely used

digital tool for transcription and translation/annotation – and FLEx, the most widely used interlinearization tool. As well as facilitating transfer between these packages, Flibl also allows researchers to create data structures in FLEx that are not possible in the out-of-the-box software. These structures were designed for research on children's acquisition of Indigenous and understudied languages, but can be adapted for work on many other topics.

Below, we explain why Flibl is necessary for data transfer between ELAN and FLEx (§2), how it improves on the software's built-in functions (§3), how it works on the backend (§4), and how users can run it (§5). The Flibl scripts and full user manual are not included here. They can be downloaded from https://osf.io/kaqr3/?view_only=609a8c7c034d4f5cbf61ad3f0e559cfb.

2. Why Flibl is necessary

The goals of Flibl (named by deleting the <exe> from *flexible*) are (1) to facilitate transfer of annotated texts from ELAN to FLEx and vice versa, and (2) to improve the usability of FLEx for child language research. To show why we set these goals, and why a separate tool is necessary, we first describe the data structures needed for observational (i.e. non-experimental) research about child language. These structures are also needed for many other types of text-oriented research. We then discuss how well ELAN and FLEx support these data structures out of the box. This is not an exhaustive discussion of the role of ELAN or FLEx in language documentation. For a more in-depth treatment of the constraints that FLEx places on language documentation projects, see Authors (under review).

2.1. Data structures for acquisition research

Our work on Indigenous child language is based on observational data. That is, we record children interacting with their caregivers and other adults at home – sometimes while playing with objects that we provide, sometimes without any specific task or directions.

At-home observational recordings are a standard method for studying child language acquisition around the world (Cristia et al. 2023). Further, because of issues surrounding the design and implementation of experiments, often observational recordings are the only feasible method for analyzing Indigenous language acquisition (Stoll 2015; Pye 2021; Paradis 2022).

While much language documentation research focuses on monologic texts, observational recordings of children are different from this type of text in two ways.

First, these recordings always have at least two participants – the enrolled (also called "focal" or "target") child participant and the caregiver – and most recordings have more than two participants, for example because they include the enrolled child's siblings. This means that we need to tag utterances for speaker (because children's vs. adults' speech is different) and addressee (because child-directed vs. adult-directed speech is different). This is also a common need in other types of research, such as documentation of conversation (Hoey & Raymond 2022) and [Author 1] uses flibl to analyze conversational data as well as acquisition data (§5).

Second, recordings of young children always include utterances that are ungrammatical relative to the adult language. Acquisition researchers need to identify which of the children's utterances

are ungrammatical, and annotate these with their grammatical, adult-like ("target") equivalents. Later, at the stage of morphological analysis, we need to gloss both the children's actual speech and the target forms; this allows us to evaluate how often the children produce a given structure correctly. These characteristics also make child language data distinct from adult conversational data.

Besides these intrinsic differences in participation and text structure, observational acquisition research also relies on a large total volume of data. [Author 3]'s child language dataset, documenting the acquisition of Ayöök (also known as Totontepec Mixe, Mixe-Zoquean, Mexico), has 44 hour-long recordings with a total of 65,054 transcribed turns at talk. [Author 1]'s dataset, documenting the acquisition of Ticuna (isolate, Peru), has 73 recordings of varying length with a total of ~25,000 transcribed turns. Other Indigenous-language acquisition corpora are similar in size (e.g., Pye 2021; Rose & Brittain 2022). Some approaches have used large language models (LLMs; see Vong et al. 2024) to deal with the amount of data, but the languages studied here are so underserved by such models that this is not an option for this research.

Like many Indigenous languages, Ayöök and Ticuna are not very well documented and do not have extensive corpus resources. As a result, the child language data is a large proportion of the total data available about the language.[1] This means we need to be sure the child data can feed back into general (i.e., not acquisition-specific) documentation products.

Further, this volume of data means that we cannot reasonably add morphological glossing by hand; we need to use a trainable parser. The number of files involved also means that we need a streamlined process for preparing each file for analysis with the parser. For example, in a dataset with 72 files, every 5-minute increase in the preparation/export time of each file will add 6 hours to the time required to import the dataset. At the other end of the workflow, our ultimate analyses will be quantitative, as in most acquisition research. We therefore need to export the final interlinearized texts in a format that is tractable for statistical analyses, such as a CSV.

2.2. ELAN

ELAN is free media annotation software developed by the Max Planck Institute for Psycholinguistics and available at http://tla.mpi.nl/tools/tla-tools/elan/. It allows users to play back audio or video media and add transcriptions and other annotations.[2] The package places very few constraints on the user. For example, users can define the type of input on each annotation tier, and files can have an unlimited number of tiers and types.

Because of these minimal constraints, ELAN allows almost all of the data structures needed for acquisition research. These structures are shown in Figure 1, a screenshot of an ELAN file from

[1] For example, almost all documentation of North East Cree, including adults' speech, comes from the Chisasibi Child Language Acquisition Study (Rose & Brittain 2022).
[2] For further information about ELAN, see Wittenburg et al. (2006), Dingemanse et al. (2012), or the official manual (MPI for Psycholinguistics 2023b). Users have also created many informal guides to the software (e.g. Neely 2019).

[Author 3]'s Ayöök corpus. We can code utterances for speaker by creating a separate time-aligned tier for each speaker and naming it with that speaker's participant code (e.g. "YDN", "YDNM"), as shown at the left of the figure. We can also code for addressee by creating an addressee tier for each participant, aligned with the transcription tier. This addressee code tier ("YDNM-xds") appears in the second row from the bottom of Figure 1. There, the code "T" denotes an utterance addressed to the **t**arget child. We can use a similar tier structure to annotate ungrammatical utterances with their grammatical counterparts – creating a target tier for each child participant, and inputting target forms there for each utterance. The target tier in [Author 3]'s file ("YDN_Target-txt-mto") is shown in the third row from the top of Figure 1. This annotation scheme is based partly on a template developed by Casillas et al (2017).

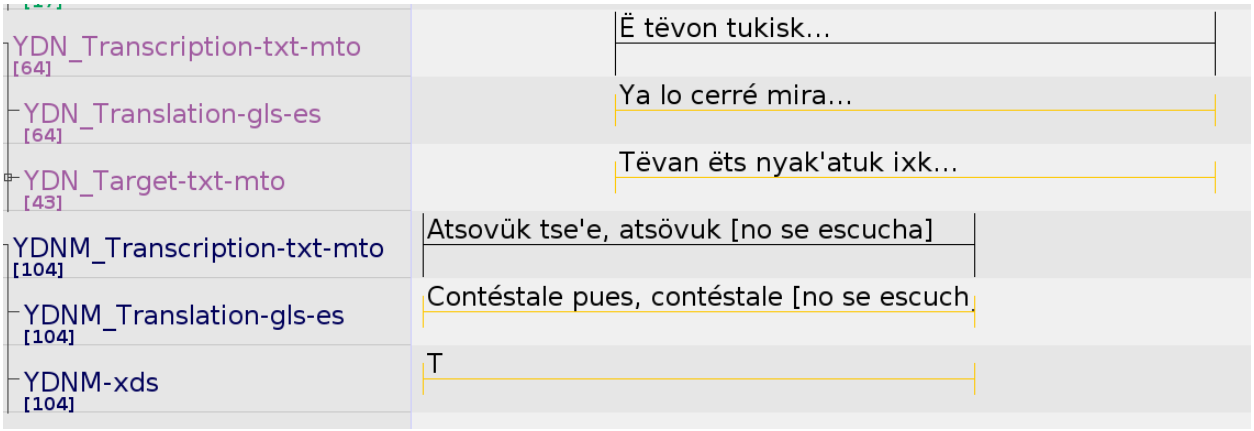

Figure 1. ELAN transcript of child-caregiver interaction

ELAN also has a number of features which speed up repetitive tasks. When starting annotation, we can use templates to rapidly set up files for transcription, and when annotation is complete, we can export transcripts to CSV format using a built-in function with batch processing capacity.

The area where ELAN does not meet the needs of acquisition research is morphological analysis. While ELAN offers an interlinearization mode, we had already used FLEx for interlinearization of texts from outside the child language project, meaning that we had already built lexicons and trained the morphological parser for interlinearization in FLEx. Combined with the other reasons described in §2.3.3, this led us to choose FLEx as our glossing tool.

2.3. FLEx

FLEx (Summer Institute of Linguistics 2022) is database software developed by the Summer Institute of Linguistics, a Christian missionary organization (Dobrin & Good 2009). The software combines a lexical database, a text database linked to the lexicon, and a powerful, trainable morphological parser which allows the user to produce glossed texts semi-automatically. As we argue elsewhere, there is no realistic competitor to FLEx for this glossing function [citation redacted for anonymous review].

The text module of FLEx requires users to input written text (not media), and the parsing function requires users to either begin from a lexicon or incrementally add words to one as they parse. Besides these requirements, FLEx also makes many other assumptions about the user, text structure, and language structure, which we have analyzed in more detail elsewhere [citation redacted for anonymous review]. Here, we discuss only the constraints that led us to create Flibl.

2.3.1. Import and Export Processes

FLEx allows users to import and export ELAN transcripts, but the built-in process is cumbersome and designed for one-off imports/exports of simple monologic transcripts (Gaved & Salffner 2014; Bodt 2022). The import requires many otherwise unnecessary changes to each ELAN file. The export does not allow CSV output, and other output types suffer from many bugs. Neither the import nor the export has automation or batch processing options. These issues make using the built-in import/export functions time- and cost-prohibitive for us.

2.3.2. Data Structures

When analyzing child language transcripts, we need to view a time-aligned transcription, translation, and speaker and addressee codes for every turn. The text module of FLEx displays transcriptions and translations, but not timestamps or speaker attributions.[3] FLEx for Linux also cannot display addressee coding in texts imported from ELAN, although FLEx for Windows can be configured to display addressee codes as a custom field.

Suppressing time and participant information makes child language transcripts – and most other interactional data – extremely difficult to interpret. For example, Figure 2 shows how the two turns shown in the ELAN file in Figure 1 look in FLEx when imported using the built-in EAF import. As it demonstrates, FLEx displays the surface forms of these lines (the "Word" line, in black text) and morpheme-level glossing information (the "Morphemes", "Lex. Entries", "Lex. Gloss", and "Lex. Gram. Info." lines, in purple text). It also displays word-level glossing information (the "Word Gloss" and "Word Cat." lines, in blue text). Below the gloss lines, the text display also includes space on each line for a free translation of the line ("Free") and notes on the line ("Note"). As shown on line 7 in Figure 2, a single text line can have multiple notes.

[3] As discussed in §4.2, the backend XML does include timestamps and speaker attributions for texts imported from ELAN. Our point here is that the user interface does not display this information.

Figure 2. FLEx view of the sequence in Figure 1 when imported with the built-in ELAN > FLEx process

Besides the information shown in the FLEx text display in Figure 2, the EAF in Figure 1 showed that these two turns came from different speakers, that they partially overlapped in time, that the adult's turn (first line) was addressed to the target child, and that the child's turn (second line) is not what the child actually produced, but instead the target (adult-like or "grammatical") form of their utterance. None of this information is visible in Figure 2.

Beyond the issue of speaker and addressee attributions, the text module of FLEx also does not allow us to associate actual (ungrammatical) and target (grammatical) representations of a line. This is because the software allows each line of text in the study ("object") language only one representation at each hierarchical level. For example, users can create both a surface representation and an underlying representation of a line, but they cannot create multiple analyzable representations of the same line.

As a result, we *can* import tiers for both children's actual utterances and their target utterances from ELAN files, but FLEx will display each tier as a separate line of text with no link to its (un)grammatical counterpart. This display is shown in Figure 3, where line 300 represents a child's actual utterance from the Ayöök materials, and line 995 represents the target form of that utterance. Because the target and actual utterances are not linked in any way, we cannot make any systematic comparisons between actual and target forms.

300	**Word**	Na	jayetmooy					
	Morphemes	na	jayu	ëts	mooy			
	Lex. Entries	na	jayu	ëts	mooy			
	Lex. Gloss	DESP	person	PRO1.SG	give			
	Lex. Gram. Info.	det	n	pro	v			
	Word Gloss	DESP	***					
	Word Cat.	det	***					
Free	Eng							
	Spa	Me lo dio una persona						

995	**Word**	Na	jayu	ëts	yë	xmooy		
	Morphemes	na	jayu	ëts	yë'ë	x=	mooy	-W
	Lex. Entries	na	jayu	ëts	yë'ë$_2$	x=$_2$	mooy	***
	Lex. Gloss	DESP	person	PRO1.SG	PRO3.SG	+6_1O.IND	give	***
	Lex. Gram. Info.	det	n	pro	pro	<Not Sure>	v	***
	Word Gloss	DESP	person	PRO1.SG	PRO3.SG	3SG gave it to 1SG		
	Word Cat.	det	n	pro	pro	v		
Free	Eng							
	Spa	Me lo dio una persona						

Figure 3. FLEx view of the actual and target forms of a child utterance when imported with the built-in ELAN > FLEx process

In sum, ELAN allows us to rapidly create transcripts, annotate them with the data structures needed for acquisition research, and export them to CSV formats, but it does not allow us to gloss texts. FLEx fulfills this need, providing a powerful parser for glossing. However, FLEx does not support the data structures that we need for acquisition research, and its ELAN import/export process is not functional.

2.3.3. Why Stay in FLEx?

These interoperability and data-structure issues relate almost entirely to FLEx, rather than ELAN. In theory, they could be resolved by moving analysis to a package other than FLEx. However, we decided against this option for several reasons.

First, as previously mentioned, there are currently no feasible competitors to FLEx that can (partially) automate the task of morphological analysis/glossing. Analyzing data outside FLEx would likely mean segmenting and glossing by hand.[4]

Second, like many documentation researchers, we already use FLEx for language work with adults. To analyze child-language materials with a different package, we would need to export data which has already been created in FLEx and is needed for glossing (e.g. the FLEx lexicon) into the new package. Due to FLEx's many interoperability issues, this task is non-trivial and might require manual re-entry of the data.

Third, analyzing child and adult-language materials in the same database is desirable in itself, because it allows data collected in the acquisition research to feed back into general language documentation. For instance, when new lexical items are found in adults' child-directed speech, they can simply be added to the main FLEx lexicon. The new words can then be fed directly into community-facing materials like dictionaries.

3. What Flibl Does

To solve these problems, we created a custom ELAN-FLEx-ELAN interchange tool, Flibl. Flibl has two purposes: (1) to increase the dimensionality of data usable within FLEx, so that we can maintain the data structures necessary for acquisition research, and (2) to improve the import/export process from ELAN to FLEx and back to ELAN.

3.1. Flibl Increases the Dimensionality of FLEx

While FLEx places many constraints on the content of text *lines*, it does not constrain the number or content of *notes* on a line. Flibl takes advantage of this freedom to increase the dimensionality of each line in FLEx. Specifically, when an ELAN file is imported to FLEx using Flibl, the script imports speaker and addressee information as notes.

---

[4] There are annotation packages in Python that allow parsing, but they are difficult to integrate into a documentation workflow (see also Pustejovsky & Stubbs 2013; Bird, Klein & Loper 2009).

Figure 4, a screenshot of a line of text imported from ELAN to FLEx with Flibl, illustrates this. In the built-in import process, the speaker attribution on this line was suppressed in FLEx (Figure 3). With the Flibl process, it instead appears a note (green box in Figure 4) below the gloss lines.

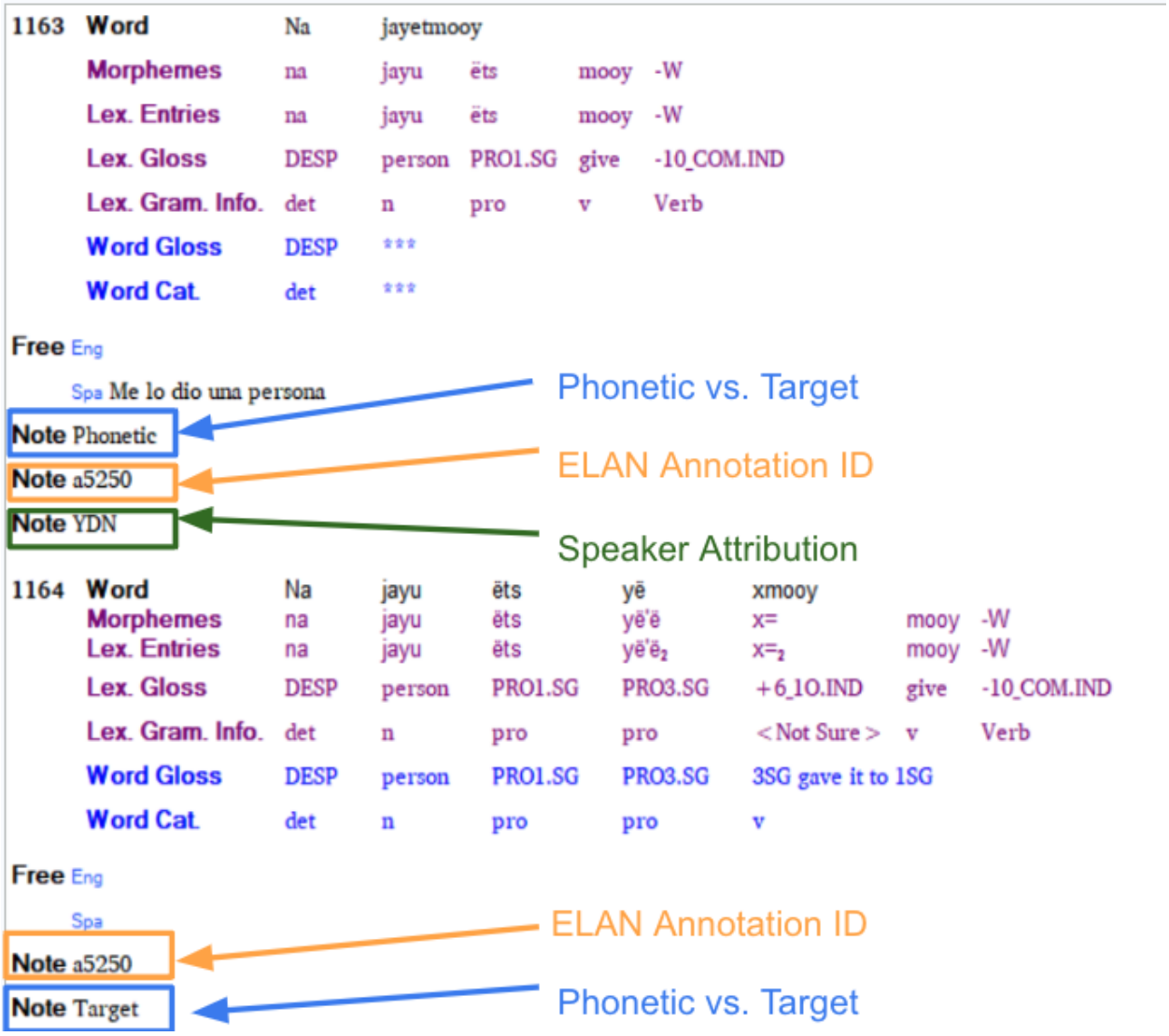

Figure 4. FLEx view of a text imported with Flibl: actual and target forms of the child's utterance are linked via a note with the original ELAN annotation number of the annotation on the time-aligned (root) tier ("a5250"). Actual vs. target forms are identified

In contrast to the speaker and addressee information about participants, we need to avoid importing target utterances as notes, because notes cannot be analyzed as lines of the text using the automatic parsing function of FLEx. Therefore, Flibl imports target utterances as independent lines of text, immediately following the actual utterances that they correspond to. The script then links the target utterance and actual utterance via a note on each line giving the ELAN annotation number of the actual utterance. (Annotation numbers are a property of ELAN annotations that is visible on the XML backend; see §4.1.) In Figure 4, the annotation number notes are shown in the orange boxes.

Below the annotation number note, an additional note marks whether the line represents the actual ("Phonetic") vs. the target ("Target") form of the utterance. In Figure 4, the actual vs. target notes appear in the blue boxes. Observe that in the line for the target utterance in Figure 4, line 1165, there is no speaker attribution. This is because Flibl never assigns speaker attributions to target utterances. There are various reasons for this. First, target utterances cannot truly be attributed to children because children do not actually produce them. Rather, they reflect an adult transcriber's hypotheses about what the child could or should have said. Suppressing speaker attributions on target forms reminds analysts of this. Second, the "speaker" in a target line is always the same as in the preceding line, so this information can be inferred. Third, on the technical side, keeping the speaker note on only the parent utterance was helpful for the logic of the program, so that Flibl could seek lines with a speaker note based on the configuration file information and treat them as the actual utterances when exporting from FLEx to ELAN.

Although the data structures in ELAN are different from those in FLEx, the speaker attribution, annotation number, and actual vs. target notes mean that the ELAN-specific information is preserved across import and export (cf. Han 2022).

3.2. Improve Import/Export Process

Flibl is a package composed of two Python scripts that users interact with, along with a Python module used in both scripts.

One script converts files annotated in ELAN in the .EAF format into the .FLExText format for import to FLEx. It takes two inputs: an ELAN file and a configuration file. Technical specifications for these files are described in detail in the documentation at https://osf.io/kaqr3/?view_only=609a8c7c034d4f5cbf61ad3f0e559cfb.

ELAN to FLEx configuration files can be reused for any ELAN file with the same tier structure. They include the following information: a list of all languages used in the ELAN file (named using the same codes as in the FLEx database setup); a list of all word-forming characters in the orthography of each language; a list of participants; a classification of each participant in terms of language use (e.g. child/learner vs. adult/fluent speaker); and a list of the names of the transcription, translation, notes, and addressee coding tiers in the ELAN file. Teams that use a consistent orthography and tier structure across all ELAN files should be able to use similar configuration files for most conversions, editing only the participant and tier names.

After creating the configuration file, users can perform the ELAN > FLEx conversion with a single command. In contrast, the built-in export-import process between ELAN and FLEx involves opening the annotated text in ELAN, navigating to the export feature in the program, adding extra information (e.g. a time-aligned tier with a title for the text), tokenizing the transcriptions into individual words, and renaming all tiers to follow FLEx conventions. Via the configuration file, Flibl automatically tokenizes and renames ELAN tiers appropriately, so that the user can avoid these steps. Even if the user is importing files with many different tier structures (which will require serially editing the configuration file), this alone makes the Flibl import process significantly faster than the built-in option.

The other script in the Flibl package converts glossed FLExTexts to ELAN files. It takes three inputs: an original ELAN file (i.e., the file used to create the FLExText that has now been annotated in FLEx), a FLExText export of the same file as glossed in FLEx, and a configuration file. Users again need to edit the configuration file with tier information, but can then run the converter with one line of code and a few interactive responses in the command line, as detailed in §5.2. The script then outputs an ELAN file with the FLEx glossing. Users can also use script options to output the glossed texts as JSON files, which can be useful in quantitative data analysis.

The FLEx > ELAN export script is much faster to use than the built-in export feature of FLEx and fixes a variety of bugs in the built-in export. For example, the built-in process produces ELAN files that remove speaker names, lack phrase-level tiers (i.e. treat each word as its own annotation), and as of FLEx 9.0, do not distinguish between different types of notes. In contrast, Flibl creates ELAN files that retain all of this information.

The export script currently does not produce a CSV version of the FLExText. We recommend that users export glossed EAFs to CSV using the built-in CSV export function of ELAN. If the endpoint of the workflow is statistical analyses of the glossed text, R users can read in the glossed EAFs to R as nested tibbles using the R scripts that we provide. Specifically, users should download the "eaf_to_table_control.R" and "eaf_to_table_functions.R" scripts from the repository, open "eaf_to_table_control.R", add the paths to their glossed EAFs and their FLEx language code in the indicated lines, and run the script to produce a nested tibble that can be analyzed in R (or saved as a CSV). An alternative is that users can select the option to output a JSON during EAF construction, then read in the JSON to their statistical package.

4. How Flibl Works

Both ELAN and FLEx store data in XML format, but the structure of ELAN vs. FLEx XML is radically different. Because they do not have 1:1 correspondences, the files are passed through a third format (JSON) which can create XML outputs of the desired structure. Due to these differences in structure, transferring files between ELAN and FLEx is much more complicated than one might expect. This section explains the XML structure used in each application and how Flibl translates between them. We assume that the reader is familiar with XML. Those new to XML may want to start with an introduction to the language (e.g. Ray 2003: chap. 1). Users do *not* need to comprehend this design information in order to use the tool.

4.1. ELAN XML Structure

ELAN creates XML files with the extension EAF and a flat XML structure.[5] "Flat" here means that every annotation tier, regardless of its linguistic type and stereotype in ELAN, appears at the same level in the XML structure. Each tier of EAF corresponds to a single XML node, with participant and linguistic type as attributes. All of the annotations in the tier are children of the XML tier node.

[5] EAF's XML schema is documented at https://www.mpi.nl/tools/elan/EAF_Annotation_Format.pdf.

In the ELAN user interface, dependent tiers – tiers where every annotation's timepoints align exactly with the timepoints of another annotation – are described in the ELAN user interface as being "children" of time-aligned tiers. (In our transcripts, the only time-aligned tiers are the transcription/actual utterance tiers.) Because these tiers depend on other tiers (for instance, every annotation on a dependent tier has exactly the same timestamps as the parent annotation on the time-aligned tier), they appear to be children in the user interface.

However, in the XML structure, dependent tiers are actually at the same level as independent tiers, rather than being nested children of the tiers they depend on. In order to establish the dependency relationship for ELAN, each dependent tier has an attribute in its definition which gives the ID of the tier which it depends on. In Figure 5, a snippet of ELAN XML, these attributes appear in the blue boxes. Additionally, within each annotation for a dependent tier, there is an attribute giving the ID of the parent tier annotation that it refers to. In Figure 5, these annotation ID attributes appear in the orange boxes. Because of this logic, only the independent tiers have timestamp information, and the placement of dependent tiers is inferred when rendering the file.

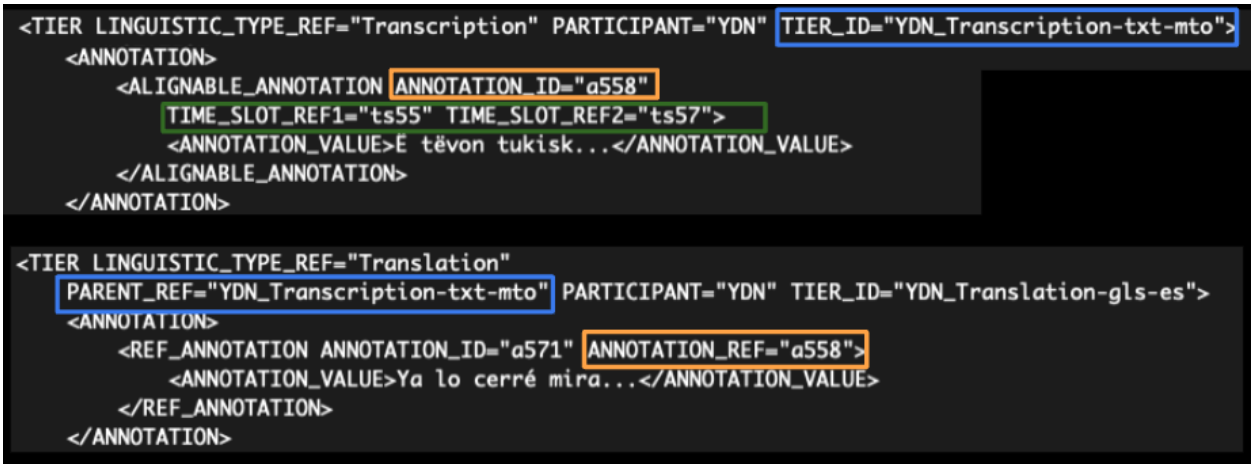

Figure 5. XML structure of a time-aligned tier (top) and a referring tier (bottom) in an ELAN file

Another key feature of ELAN XML that is not visible in the user interface is that, for annotations on time-aligned tiers, the begin and end timestamps are not stored as attributes of the annotation node. Instead, the XML contains a node, at a level above the tiers, called "Time Order." This node stores all timestamps in the file as pairs of an actual timecode and an arbitrary time slot number. Annotations are linked to their timestamps by an XML attribute that refers to the time slot number. This is the "time slot reference" ("TIME_SLOT_REF") attribute shown in the green box in Figure 5.

Last, an additional hidden feature of ELAN XML is that the ID numbers for time slots and annotations are not ordered based on the chronology of the media of the file. Instead, the ID numbers are based on the order in which the annotations were created. There are often gaps in the number sequence due to deleted annotations, and annotation IDs are sometimes not in numerical order due to the creation of annotations in an order that is not linear with the media.

4.2. FLEx XML Structure

FLEx also stores data in XML, but it uses a hierarchical, linguistically motivated XML structure. Texts in the .FLExText XML format are found within the files that make up a FLEx database; this format also appears as an option for text export. They are organized into paragraphs, then phrases, then words, then morphemes. Each level in this structure corresponds to an XML node that has the hierarchically lower XML nodes as its children.

For example, Figure 6 shows a snippet of the FLExText created by importing the ELAN file in Figure 1 using the built-in process. (The user's view of this phrase in the FLEx texts module appears in Figure 2.) In Figure 6, the top XML node is a "paragraph." It contains one "phrases" node, which in turn contains one "phrase" node. The "phrase" node has XML attributes, but not text. Instead, the complete unanalyzed text of the phrase is stored as an "item" child of the phrase node. It has the type attribute "txt" (object language text) and the language attribute "mto" (the ISO code for Ayöök). The Spanish gloss of the phrase is stored as a sister node to the original-language text, with the type "gls" (free translation) and the language "es". (This is why the configuration files for Flibl need to include a list of languages.) The "words" node of the phrase, where the individual words of the line are stored, is a sister to the "item" nodes containing the text and free translation.

```
<paragraph guid="0324d04e-32bf-4757-9b35-9d00463f2723">
    <phrases>
        <phrase begin-time-offset="13330"
            end-time-offset="15280"
            guid="e4544e6d-82dc-4dfd-90c8-0f4e25784076"
            media-file="eb26fb4d-6979-4b47-b16c-b106de4eee46" speaker="YDN">
            <item lang="mto" type="txt">Ë tëvon tukisk...</item>
            <item lang="es" type="gls">Ya lo cerré mira...</item>
            <item lang="mto" type="txt">Tëvan ëts nyak'atuk ixk...</item>
            <words>
                <word guid="8b477c6f-ce37-4c11-8ea6-f21b60514403">
                    <item lang="mto" type="txt">Ë</item>
                </word>
                <word guid="77fb993d-b367-44d3-a07b-1d31b44d6664">
                    <item lang="mto" type="txt">tëvon</item>
                </word>
                <word guid="2df2d34a-8a7e-4953-935f-ef26ba6b79df">
                    <item lang="mto" type="txt">tukisk...</item>
                </word>
            </words>
        </phrase>
    </phrases>
</paragraph>
```

Figure 6. XML structure of a phrase, its translation and its constituent words in a FLExText created by the built-in ELAN to FLEx import process.

Figure 6 also illustrates how FLEx treats participant and timestamp information. In texts imported from ELAN using the built-in process, speaker codes are stored as an attribute of the phrase node (blue box in Figure 6). Timestamps are also stored as attributes of the phrase node

(orange box in Figure 6). They are treated as numeric values (in milliseconds), not linked to a time slot number as in ELAN. Although this speaker and time information is stored on the phrase node in the FLExText, it is – strangely – not displayed in the user interface. Additionally, if the FLExText is exported to ELAN with the built-in process, speaker attributions are replaced by arbitrary letters [A, B,...].

Note that the line from a text shown in Figure 6 has not been assigned a morphological analysis. To illustrate how morphological information is treated in FLExText XML, Figure 7 shows a single word from the same text with analysis. The word node is the parent of a "morphemes" node, which is in turn the parent of a single "morph" node. The "morph" node is the parent of several "item" nodes, which contain the surface realization ("txt"), underlying form ("cf"), gloss ("gls"), and syntactic category ("msa") of the morpheme. If this word was multimorphemic, the "morph" node would have a sister node for each morpheme with the same structure. Finally, this text has been glossed and has part-of-speech tagging at the word level as well as the morpheme level; therefore, the word node also has immediate "item" node children which contain the word-level gloss and word-level part of speech.

```
</word>
<word guid="5059d1e3-55dd-4f02-afdd-e7f912aeef39">
  <item type="txt" lang="cps">tëvon</item>
  <morphemes>
    <morph type="stem" guid="d7f713e8-e8cf-11d3-9764-00c04f186933">
      <item type="txt" lang="cps">tëvan</item>
      <item type="cf" lang="cps">tëvan</item>
      <item type="gls" lang="en">already</item>
      <item type="msa" lang="en">adv</item>
    </morph>
  </morphemes>
  <item type="gls" lang="en">already</item>
  <item type="pos" lang="en">adv</item>
```

Figure 7. XML structure of a word, its word-level gloss, its constituent morphemes, and their glosses in a glossed FLExText.

While glossing texts in FLEx requires linking to a FLEx project lexicon, FLExText XML does not contain the lexicon - rather than linking tokens of the same morpheme to a dictionary or lexicon, all of the morphological information is repeated for every token.

4.3. Interchange Format

In order to translate between these formats, Flibl uses a JSON-like object which organizes the text hierarchically. While XML emerged as a markup language to specify instructions for another program to render information, JSON is a format designed for data organization. JSON files can be made hierarchical by nesting items. This structure is very useful for making interchange documents.

In converting EAFs to FLExText format, we parse the EAF XML and convert it to the JSON interchange format. We then construct a FLExText from the JSON, restructuring the speaker,

addressee, and annotation ID attributes of each time-aligned phrase as children of the phrase node with the type "note." We also generate a variety of additional information that is not based on the EAF, but is required for well-formed FLExText XML. This includes, for example, generating arbitrary GUIDs (Global/Universal Unique Identifiers, a 32 character long hexadecimal string) for each paragraph, phrase, and word (see the "guid" attributes of these nodes in Figure 6).

To convert glossed FLExTexts to EAF format, we parse the XML of both the FLExText and the original EAF used to create it. We use the original EAF to create a new well-formed EAF with tiers for the additional annotation types in the FLExText, such as the morpheme forms, glosses and syntactic categories. We convert the glossed FLExText to the JSON interchange format, and we match phrases in the JSON to transcription tier annotations in the original EAF using their annotation IDs. We then use the word and morpheme information in the JSON to construct annotations on the word and morpheme tiers in the EAF. When running the script, users can optionally output this JSON file as well as the EAF.

5. How to use Flibl

This section describes how to download and use Flibl. Full documentation appears in the Flibl repository, https://osf.io/kaqr3/?view_only=609a8c7c034d4f5cbf61ad3f0e559cfb.

For both ELAN to FLEx and FLEx to ELAN conversions, users must download the files "flexible.py", "flextext_construction.py," "to_flextext_config.json ," "eaf_construction.py," and "to_eaf_config.json" from the repository. Users also need to have Python 3.10 or higher installed. Python is included on many operating systems and easily installed using Anaconda.

If a user's ELAN file has more than ~600 turns at talk, and they plan to work with the final glossed text in ELAN rather than exporting to another format, we recommend breaking it into smaller files before the first import step. For example, when we gloss conversation recordings, we divide each hour-long recording (typically 1000-1800 turns) into sections of 20 minutes each (500-600 turns). This is necessary because adding morphological information to an EAF balloons the file size. As a result, fully glossed EAFs with more than ~600 turns are too large to view in ELAN - they may not open at all, or may make the software unresponsive. Users can divide EAFs into sections by selecting each section's timepoints in Annotation or Segmentation Mode, then using the "File > Save Selection as EAF" command.

5.1. ELAN to FLEx Conversion

To transfer a file from ELAN to FLEx, users should take the following steps.

First, the user will need to create an ELAN to FLEx configuration file, "to_flextext_config.json," to reflect their project conventions. The format and information needed for the configuration file are fully specified in the documentation ("Readme.md"), as well as listed above in §3.2. The repository includes a sample ELAN to FLEx configuration file from the Ayöök project as an example of a well-formed input. Additionally, the next release of Flibl will include HTML forms which people can use to interactively generate configuration files.

After generating or editing the EAF to FLEx configuration file, users should check that "flexible.py," "flextext_construction.py," the configuration file, and the input EAF(s) are all in the same directory. Users can then open the command line, navigate to the relevant directory, and run the script using the command "python flextext_construction.py" (or "python3 flextext_construction.py" if "python" runs Python 2 on your computer; this is the case for many Macs). If the script runs correctly, this will produce one FLExText for each input EAF. The user can then open FLEx and import the FLExText file(s) interactively by selecting "File > Import > FLExText Interlinear."

If the script does not run correctly, the user will see an error print to the command line, and no FLExText will be generated in the directory. Most of the errors that we have encountered so far have come from not formatting the EAF to Flibl's specifications - for example, using different participant codes for tier prefixes vs. for the participant attribute of the tier. The documentation provides full specifications for the input EAF and configuration file, as well as a troubleshooting guide.

As users are glossing and editing the imported texts in FLEx, they need to avoid making changes to the notes fields generated by Flibl. These are the note with the participant name, the note with the phonetic/target code, and the note with the annotation number. Our scripts rely on these fields to convert between EAF and FLExText formats. If they are edited, the user will not be able to re-export the text to ELAN using the FLEx to EAF converter. Users also need to avoid changing line breaks in FLEx; this will break the converter too. All other changes to the baseline, translation, and notes (e.g. correcting transcriptions or adding observations in a new note) are acceptable.

5.2. FLEx to EAF Conversion

To transfer a file from FLEx to ELAN using Flibl, the first step is to export it from FLEx in FLExText format by selecting "File > Export > FLExText". Users should then save the FLExText in the same directory as "flexible.py," "eaf_construction.py," the configuration file, and the original input EAF. Users will then need to edit the configuration file "to_eaf_config.json" to reflect their input file names and project conventions. As with the EAF to FLEx process, users can refer to the sample version of "to_eaf_config.json" on the repository and to the extensive instructions for creating config files provided in the documentation.

Once the FLEx to ELAN configuration file is ready, users should check that all of the needed files are in the same directory. When they are, the user can then open the command line, navigate to the directory, and run the command "python eaf_construction.py" (or "python3" on Macs). The script will run, printing a progress report to the command line. When complete, it will generate an EAF in the same directory. If the export script does not run correctly, the user will see an error in the command line and no FLExText in the folder. Errors at this point tend to come from formatting issues in the configuration file or FLExText, such as lines missing "Phonetic" vs. "Target" tags. Users should check that the configuration file, FLExText, and EAF conform to the specifications.

Once the user has an EAF, they can open it and view it in ELAN. In ELAN's Annotation Mode, the glossed text will look approximately like Figure 8.

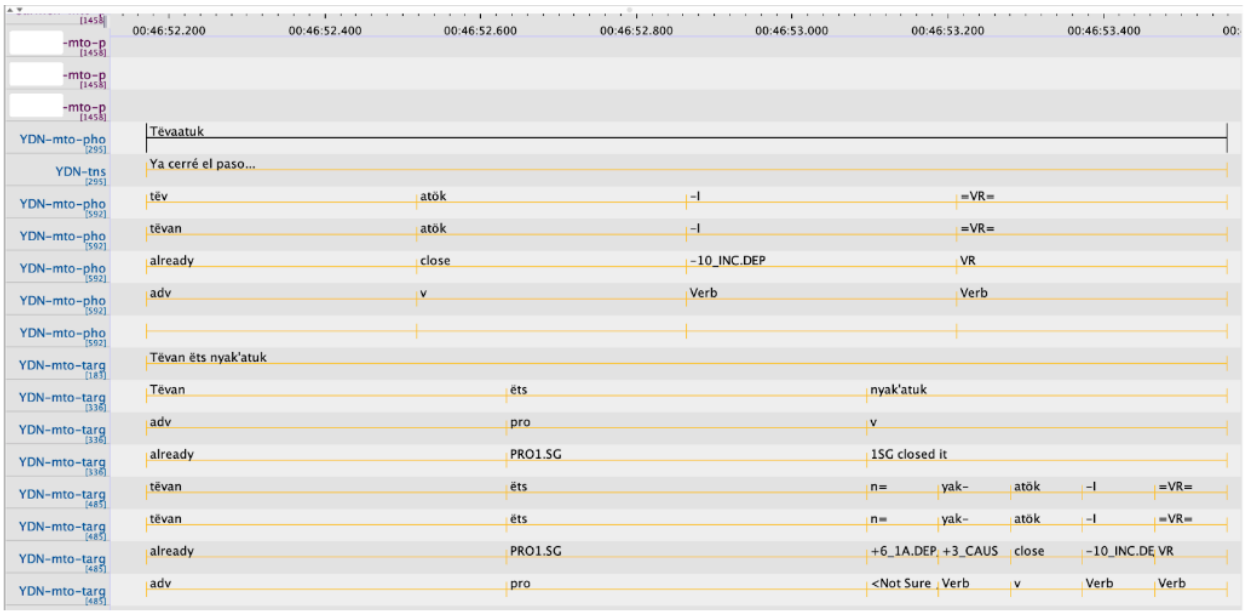

Figure 8. ELAN file exported from a glossed FLExText using Flibl.

Output EAFs will not include any tiers that were excluded during the FLEx import process. For example, if the user excluded time-aligned tiers with gesture annotations from the import, those tiers will not be present in the EAF export on this end. However, users can add excluded tiers again by merging the input and output EAFs with the "File > Merge Transcriptions" command in ELAN.

If users want to work with the glossed text directly in ELAN, at this point the import process is done. If they prefer to analyze the data in spreadsheet form (using Excel, for example), they can export it to CSV using the built-in CSV export function of ELAN ("File > Export as > Tab-delimited text"). Alternatively, if the endpoint of the workflow is statistical analyses of the glossed text, users can also (1) read the EAFs directly into their statistical package or (2) use the flag "-j" (i.e., "python eaf_construction.py -j") to output a JSON during EAF construction, then read in the JSON instead of the EAF. For instance, R users can read in the EAF directly using the R scripts that we provide ("eaf_to_table_control.R" and "eaf_to_table_functions.R"), or can read in the JSON with "jsonlite" (Ooms, Temple Lang & Hilaiel 2023).

6. Conclusion

In this paper, we have described the need for Flibl: a series of scripts which facilitate transfer of text materials from ELAN to FLEx and vice versa, specifically (though not exclusively) to facilitate research in child language acquisition.

Since child language researchers require data structures that are not necessary for adult-centered work (particularly on monologic texts), they have tended to use ELAN for transcription. Working with such ELAN files with FLEx, however, necessitates a conversion process which is not currently possible using FLEx's native import formats. Flibl solves this problem by creating

an interchange format for ELAN and FLEx XML files. We use JSON to create a hierarchical text object which can then be ported between ELAN and FLEx, as described in §4.3.

While this tool has focused on the needs for child language acquisition research, the interchange of material between ELAN and FLEx is a more general issue. Researchers working with the linguistic analysis of audio and visual materials often need to parse such materials, and ELAN does not yet do that. FLEx provides a parser, but does not allow the integration of non-text language material within FLEx itself. Future work could extend Flibl for other use cases.

7. Bibliography

Austin, Peter K. 2006. Data and language documentation. In Jost Gippert, Nikolaus Himmelmann & Ulrike Mosel (eds.), *Essentials of language documentation*, 87–112. Berlin: Mouton de Gruyter.

Bird, Steven, Ewan Klein & Edward Loper. 2009. *Natural language processing with Python*. 1st ed. Beijing ; Cambridge [Mass.]: O'Reilly.

Bodt, Timotheus. 2022. An integrated FLEx–ELAN workflow for linguistic analysis with multiple transcriptions and translations and multiple participants. *Language Documentation & Conservation* 16. 417–452. http://hdl.handle.net/10125/74686.

Boersma, Paul & David Weenink. 2022. Praat: doing phonetics by computer. http://www.praat.org.

Bowern, Claire. 2015. *Linguistic fieldwork: practical guide*. Second Edition. New York: Palgave Macmillian.

Casillas, Marisa, Elika Bergelson, Anne S Warlaumont, Alejandrina Cristia, Melanie Soderstrom, Mark VanDam & Han Sloetjes. 2017. A New Workflow for Semi-automatized Annotations: Tests with Long-Form Naturalistic Recordings of Childrens Language Environments. In *Proceedings of Interspeech 2017*, 2098–2102.

Cristia, Alejandrina, Ruthe Foushee, Paulina Aravena-Bravo, Margaret Cychosz, Camila Scaff & Marisa Casillas. 2023. Combining observational and experimental approaches to the development of language and communication in rural samples: Opportunities and challenges. *Journal of Child Language* 50(3). 495–517. https://doi.org/10.1017/S0305000922000617.

Dingemanse, Mark, Jeremy Hammond, Herman Stehouwer, Aarthy Somasundaram & Sebastian Drude. 2012. A high speed transcription interface for annotating primary linguistic data. In *Proceedings of the 6th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 7–12. Association for Computational Linguistics.

Dobrin, Lise & Jeff Good. 2009. Practical Language Development: Whose Mission? *Language* 85(3). 619–629.

Epps, Patience L., Anthony K. Webster & Anthony C. Woodbury. 2017. A Holistic Humanities of Speaking: Franz Boas and the Continuing Centrality of Texts. *International Journal of American Linguistics* 83(1). 41–78. https://doi.org/10.1086/689547.

Foley, William A. 2003. Genre, register and language documentation in literate and preliterate communities. *Language Documentation and Description* 1. 85–98. http://www.elpublishing.org/PID/009.

Gaved, Tim & Sophie Salffner. 2014. Working with ELAN and FLEx together: an ELAN-FLEx-ELAN teaching set. SOAS University of London, ms. https://groups.google.com/group/flex-

list/attach/18bab21291a9a/Working%20with%20ELAN%20and%20FLEx%20together_2014-01-20.pdf?part=0.1.
Glenn, Akiemi. 2009. Five Dimensions of Collaboration: Toward a Critical Theory of Coordination and Interoperability in Language Documentation. *Language Documentation & Conservation* 3(2). 149–160.
Han, Na-Rae. 2022. Transforming Data. In Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller & Lauren B. Collister (eds.), *The Open Handbook of Linguistic Data Management*, 73–88. The MIT Press. https://doi.org/10.7551/mitpress/12200.003.0010.
Hoey, Elliott M. & Chase Wesley Raymond. 2022. Managing Conversation Analysis Data. In Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller & Lauren B. Collister (eds.), *The Open Handbook of Linguistic Data Management*, 257–266. The MIT Press. https://doi.org/10.7551/mitpress/12200.003.0025.
Meakins, Felicity, Jennifer Green & Myfany Turpin. 2018. *Understanding linguistic fieldwork*. London: Routledge.
MPI for Psycholinguistics. 2023a. ELAN (Version 6.7). Nijmegen: The Language Archive. https://archive.mpi.nl/tla/elan.
MPI for Psycholinguistics. 2023b. ELAN 6.7 Manual. https://www.mpi.nl/tools/elan/docs/manual/index.html.
Musgrave, Simon & Nick Thieberger. 2021. The language documentation quartet. In *Proceedings of the 4th Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, 6–12. Online: Association for Computational Linguistics. https://aclanthology.org/2021.computel-1.2.
Neely, Kelsey C. 2019. Some tips and tricks for using ELAN. http://kelseycneely.com/files/Neely-Some-tips-and-tricks-for-using-ELAN-Sept2019.pdf.
Ooms, Jeroen, Duncan Temple Lang & Lloyd Hilaiel. 2023. jsonlite (R package). https://cran.r-project.org/web/packages/jsonlite/index.html.
Paradis, Johanne. 2022. What can journals do to increase the publication of research on the acquisition of understudied languages? A commentary on Kidd and Garcia (2022). *First Language* 42(6). 794–798. https://doi.org/10.1177/01427237221089171.
Pustejovsky, James & Amber Stubbs. 2013. *Natural language annotation for machine learning: a guide to corpus-building for applications*. 1. ed. Beijing Köln: O'Reilly.
Pye, Clifton. 2021. Documenting the acquisition of indigenous languages. *Journal of Child Language* 48. 454–479.
Ray, Erik T. 2003. *Learning XML*. 2. ed. Beijing Köln: O'Reilly.
Rose, Yvan & Julie Brittain. 2022. Managing Phonological Development Data within PhonBank: The Chisasibi Child Language Acquisition Study. In Andrea L. Berez-Kroeker, Bradley McDonnell, Eve Koller & Lauren B. Collister (eds.), *The Open Handbook of Linguistic Data Management*, 391–400. The MIT Press. https://doi.org/10.7551/mitpress/12200.003.0037.
Stoll, Sabine. 2015. Studying language acquisition in different linguistic and cultural settings. In Nancy Bonvillain (ed.), *The Routledge Handbook of linguistic anthropology* (Routledge Handbooks in Linguistics), 140–158. New York, NY: Routledge.
Summer Institute of Linguistics. 2022. FieldWorks Language Explorer. https://software.sil.org/fieldworks/download/fw-91/fw-9118/.

Vong, Wai Keen, Wentao Wang, A. Emin Orhan & Brenden M. Lake. 2024. Grounded language acquisition through the eyes and ears of a single child. *Science* 383(6682). 504–511. https://doi.org/10.1126/science.adi1374.
Wittenburg, Peter, Hennie Brugman, Albert Russel, Alex Klassmann & Han Sloetjes. 2006. ELAN: a professional framework for multimodality research. In *5th International Conference on Language Resources and Evaluation (LREC 2006)*, 1556–1559.