LingView: A Web Interface for Viewing FLEx and ELAN Files

Kalinda Pride* (Brown University), Nicholas Tomlin* (Brown University), Scott AnderBois (Brown University)

Abstract

This article presents LingView (`https://github.com/BrownCLPS/LingView`), a web interface for viewing FLEx and ELAN files, optionally time-synced with corresponding audio or video files. While FLEx and ELAN are useful tools for many linguists, the resulting annotated files are often inaccessible to the general public. Here, we describe a data pipeline for combining FLEx and ELAN files into a single JSON format which can be displayed on the web. While this software was originally built as part of the A'ingae Language Documention Project to display a corpus of materials in A'ingae, the software was designed to be a flexible resource for a variety of different communities, researchers, and materials.

*Equal contribution between first two authors.

List of Figures

1 Introduction

In recent decades, language documentation has emerged as both an interdisciplinary field and as a form of community engaged scholarship wherein speakers of indigenous and other minority languages can be empowered to create various kinds of materials of direct use and interest to community members. Language documentation researchers have been aided by the creation of powerful computer-based tools such as EUDICO Linguistic Annotator (Max Planck Institute for Psycholinguistics (2018), Wittenburg, P., Brugman, H., Russel, A., Klassman, A. and Sloetjes, H. (2006)) and Fieldworks Language Explorer (SIL FieldWorks (2018)), here referred to as ELAN and FLEx respectively. Such tools have allowed for the creation of sizable bodies of language materials, along with richly structured annotations including transcriptions, translations, and morphological analyses.

While these tools are invaluable, they are designed for the production and analysis of language materials by trained users and therefore are not themselves suited for presenting these materials to a diverse audience in a user-friendly way. Existing means of developing such user-facing outputs (e.g. creating subtitled videos using ELAN) require one-time choices to be made about what orthography to use, what translation(s) to include, etc. That is to say, they produce outputs which don't pass on much of the rich structure that has been created.

In this paper, we present LingView, a set of tools that allow language documentation practitioners to present language materials from ELAN and FLEx to a diverse range of users with different goals and needs. To do this, we provide a pipeline for converting ELAN and FLEx files into a single JSON format together with a graphic user interface that allows users to choose which of these fields are displayed.

The rest of the paper is structured as follows: §2 motivates the system both in general as well as by describing its application within the documentation of A'ingae, an understudied language of Ecuador and Colombia also known as Kofán or Cofán (ISO 639-3 code: `con`). §3 describes how the system is structured at a general level (more specific technical documentation can be found at `https://github.com/BrownCLPS/LingView`). §4 describes the recommended workflow for practitioners to use LingView alongside ELAN and FLEx. Finally, §5 summarizes and presents some potential directions for future development.

2 Motivating the system

Language documentation is a field that typically involves collaboration between individuals with different goals and differing levels of technical linguistic expertise. Whether they are native speaker linguists or outsiders, linguists often have an interest in examining detailed linguistic analyses, such as morphological analyses with detailed glosses or phonetic transcriptions. On the other hand, these detailed technical analyses may be unintelligible to community members or simply may not be relevant to their goals. Furthermore, in the case of endangered languages, there often will also be community members with limited or no fluency in the language being documented, but who nonetheless have a strong interest in the materials being produced. In short, the audience for language documentation materials both within the language community and beyond is diverse and varies in many ways across languages.

Given this diverse audience, language documentation practitioners often face choices in producing materials of how to balance these diverse goals. In some cases, this forces potentially quite fraught decisions between the needs of different constituents within the larger audience. For example, a practitioner may choose a single orthography in a context where multiple orthographies are in use or a single translation language in a context where there are multiple languages of wider communication in which translations would be relevant to different groups within the community (e.g. an indigenous language spoken by communities in both Brazil and Colombia).

For some materials, their nature may make such choices inevitable. The audience of an academic article will naturally have a different set of readers with different goals than elementary school pedagogical materials will. For others, however, such as audio, video, and written texts, the needs of different audience members are relatively similar, consisting of the primary media itself possibly along with various annotations such as transcriptions, translations, or morphological analysis. Different users may have more or less need for annotations or require different kinds or amounts, but the needs of different users are to an extent aligned with one another.

While producing audio, video, and written texts to serve a diverse audience is a challenge, it is one which modern computational tools such as richly structured databases and web-based interfaces have the potential to greatly reduce. Creating such platforms may seem a large task, but the reality is that many language documentation practitioners already create richly structured databases of annotated media of the kind that is needed in the form of ELAN files and databases in FLEx. At present, however, creating language materials typically involves taking these richly structured files and selectively removing some of this richness.

By leveraging the complex, structured ELAN and FLEx files practitioners are already producing, LingView aims to pass along this rich structure to users. In particular, through a series of scripts, LingView creates a backend database which allows for ELAN and FLEx files to be stored in a common format retaining most of their rich structure. Second, the user interface allows for individual users to decide which of the fields in this database will be displayed, as seen in Figure 1.

<Figure 1>

Beyond this general motivation, we discuss more specific cases where the customizability this user interface provides is useful. We do this first by discussing in §2.1 the specific project for which LingView was created, the A'ingae Language Documentation Project (ALDP) and second by discussing in §2.2 the design features for LingView that emerged from this case as well as consideration of other language documentation situations.

2.1 Case study: A'ingae Language Documentation Project

In this section, we provide background on the A'ingae language, its speakers, and the ongoing community collaborative language documentation project, ALDP, within which LingView was created. Beyond laying out some of the specific motivations for features of the software in more detail, we believe that this case study illustrates a number of the complexities that are common in language documentation and therefore serves as a useful example of broader patterns potential users are likely to face.

A'ingae is an indigenous language spoken in the northeast of Ecuador and southeast of Colombia, primarily along the San Miguel and Aguarico rivers (Fig. 2).[1] The language has approximately

[1]Map is from `https://library.brown.edu/create/firstreading2014/learn-more/`

1,500 speakers and is endangered (especially so in Colombia), but is nonetheless still being learned by children in several communities. In some towns (e.g. Zábalo and Dureno), A'ingae remains the language of everyday life across most social settings, while in other towns Spanish is increasingly prevalent in certain contexts. As is clear from Figure 2, the towns where A'ingae is spoken are fairly far from one another. They also differ in other ways such as the extent to which community members rely on a traditional hunter-gatherer lifestyle, access to internet, interactions with settlers and other outside groups, and prior and ongoing impact of oil exploration and its secondary consequences (see Cepek (2018) for a detailed examination).

<Figure 2>

A few short word lists notwithstanding, outside scholarship on A'ingae began in 1955 with the arrival in Dureno of SIL missionaries Marlytte ("Bub") and Roberta ("Bobbie") Borman. The Bormans developed a practical orthography using Latin letters and based in part on Spanish orthography. They produced a few short scholarly papers on specific grammatical topics such as phonology (Borman, 1962) and pragmatics (Borman, 1977), as well as a few community-oriented publications focused on topics such as health and literacy.

In addition to a bible translation, the two most sizable works they produced are a $\approx$2,500 word A'ingae-Spanish bilingual dictionary with brief grammatical notes (Borman, 1976) and a trilingual (A'ingae-English-Spanish) collection of texts about cosmology from a single speaker, Enrique Criollo (Borman, 1990). While community members were aware of these works existing, even teachers and others with an interest in them reported not having access to them either in physical or digital form.

More recent academic work has included a comprehensive grammatical sketch (Fischer and Hengeveld, ta), papers covering particular topics in more detail (Fischer and van Lier (2011), Hengeveld and Fischer (msa), Hengeveld and Fischer (msb), Repetti-Ludlow et al. (ms)), and community-oriented stories (Blaser and Chica (2008), Comunidad Cofán de Zábalo and Comunidad Cofán Pakuya (2011)).

Aside from mere availability, the dictionary and existing text collections exhibit several properties which limit their utility to the present-day A'i who might use them and which are not uncommon for such materials. First, the orthography that they use is no longer actively used by most A'ingae-speaking communities. In Ecuador, a community-led process produced a new orthography in the last decade, which modifies the Borman orthography in several respects.[2] As far as we have seen, this orthography is currently in use by teachers and is the only one with which younger generations are familiar. In Colombia, materials found online suggest that a slightly different orthography from either of these has begun to be used, though the details of the process that produced it and its current usage remain unclear. One of the challenges that we faced then in producing written content including annotations to audio and video is that different community members have familiarity with only one of these orthographies.

Second, with respect to written texts, Borman (1990) illustrates the trade-offs that traditional print publications require. Will there be translations? What language(s) will they be in? Will there be morphemic analysis and glossing? As seen in Figure 3, in this case, the decision was made to

maps-of-ecuador/map-of-cofan-territories/

[2]Key changes include: (i) writing voiceless velar stops with "k" instead of the Spanish-like "c" and "qu"; (ii) writing the mid/high back vowel as "u" instead of "o" and the high central vowel as "û" instead of "u"; and (iii) writing aspiration with digraphs ending in "h" rather than doubling of the unaspirated stop grapheme (i.e. "th" rather than "tt").

include all of the conceivably relevant translations, analysis, and glosses. The result is a text which has all the information most users could want, but in a format which is fairly difficult to read for any user.

<Figure 3>

None of this is to question the decision that the authors made in this case. Leaving out any of the information they include would have deprived some group of potential users of necessary information. For example, if the stories had been published only in A'ingae, they would not be accessible to outsiders and would be less accessible to some community members too, since literacy in Spanish often exceeds literacy in A'ingae. Rather, we believe this example illustrates the impossibility of any one-time decision like this meeting the needs of all segments of the potential audience for such materials.

The other major collection of written materials, Blaser and Chica (2008), addresses this by publishing alternating A'ingae and Spanish language texts, Figure 4. While this of course creates texts that are more accessible to most community members, it doesn't make available any of the analysis. It also omits English translations, something which several teachers and other participants in community workshops identified as a desirable feature in a pilot version and which needless to say is of use to some researchers too.

<Figure 4>

In terms of audio and video, there is little prior material in the language available to community members via Youtube, including a dubbed film about the life of Jesus and a number of videos about the A'i people, their culture, land, and struggles to maintain these in the face of outside pressures such as oil companies and settlers. Most of these videos are aimed at outsiders and are limited in topic and genre. While some are subtitled in Spanish, none have transcriptions in A'ingae and so are less useful for pedagogical purposes.

As mentioned above, LingView is part of the A'ingae Language Documentation Project (ALDP) [Proyecto de documentación lingüística del A'ingae]. ALDP is a community-based language documentation research project started by the third author together with Wilson Silva (a faculty member at the University of Arizona) and Hugo Lucitante, an A'i community member and native speaker of A'ingae who is also an undergraduate student at Brown University. The project aims to create a multimedia record of the A'ingae language and its use across a variety of different social and communicative contexts as well as to use this record to explore aspects of the language in greater scientific detail and create materials that serve the needs of the A'i community as they work to maintain and revitalize their language and assert their right to cultural autonomy more generally. To date, the project has collected approximately 17 hours of high quality video from speakers in Zábalo covering a variety of topics and genres as chosen by participants. Of this corpus, the majority is transcribed and translated into Spanish and a smaller portion is analyzed morphologically and translated into English as well.

With this mix of academic and community goals in mind, LingView was developed as a set of tools to make audio, video, and written materials available to community members while allowing for the creation of a linguistically sophisticated database. This latter part can be achieved to a large extent by existing tools long used by language documentation practitioners such as ELAN and FLEx. However, ELAN and FLEx do not themselves provide user interfaces designed for non-linguists and so do not themselves serve community needs. While existing options such as creating subtitled videos in ELAN and IATH ELAN Text-Sync Tool (ETST, Dobrin and Ross (2017)) bridge this gap somewhat, they both still require one-time choices in which fields to display. They therefore do not fully meet the needs of the diverse community of users described above with their different interests, goals, language backgrounds, and literacy levels. These sorts of challenges are quite general ones in language documentation and LingView was therefore designed as a more general tool rather than tailored to ALDP specifically.

2.2 Design principles

In this section, we outline the four main design principles we had in mind in creating LingView: (i) multimedia compatibility, (ii) end-user customizability, (iii) linguist-friendly input, and (iv) practitioner-friendly pipeline.

(i) Multimedia compatibility

Language documentation practitioners typically collect a mix of audio and video language materials. While video materials are increasingly common as videography technology becomes more affordable and widely available, there may be any number of reasons why audio materials may also be collected. Additionally, legacy materials are quite likely to exist only as audio or possibly only in written form (e.g. the two prior story collections in A'ingae). Given the mix of previously existing written materials and new video materials our project was working with, compatibility with these different media types was an important feature.

When both formats are available, users may alternate between audio and video displays by clicking the "show video" button; this may be especially useful when weak internet connections preclude videos from loading. Alternatively, when audio and video resources are not available, we allow users to upload text-only materials by exporting from FLEx or by uploading an ELAN file without providing the associated audio or video.

(ii) End-user customizability

Language documentation brings together people with a diverse range of interests, backgrounds, and prior language experience. The consumers of language documentation materials are therefore similarly diverse and so different end-users will have different needs in terms of the kind and amount of annotations that accompany primary media. While it may be obvious that community members and researchers may differ in this way, we have tried to show that the complex reality of many language documentation situations means that similar kinds of diversity exist within language communities and so there is often not a finite set of options for practitioners to choose, but rather a multiple of different needs for different end-users.[3]

To address this challenge, we designed LingView to allow for end-users themselves to customize the choices of translations, analysis, orthography, etc. As described below, we aim to address this challenge through the use of a richly structured database along with a user interface allowing for the end-user to customize the display at any moment.

[3]Even this is somewhat of a simplification, as there will be cases in which an individual user might have different goals on different occasions and so might want correspondingly different settings. For example, a native speaker linguist might sometimes want to see morphological analysis, but other times only want monolingual subtitles. Or a linguistic anthropologist from outside the community might sometimes wish to see morphological analysis for linguistic purposes, but other times only need bilingual subtitles.

While we expect the current tools to be quite useful, many communities would benefit from additional features or community-specific alterations (see §5). To support these cases, we encourage technically-inclined users to improve and customize the software, and to freely share their improvements. The source code for all LingView tools is available through GitHub, a platform which both facilitates the creation and sharing of custom versions of the software, and allows changes to be reincorporated into the main version. LingView is copyrighted under the permissive MIT License in order to encourage such innovation.

(iii) Non-proprietary input

Language documentation practitioners already have a set of commonly used tools at their disposal such as ELAN and FLEx. As such, LingView was designed to be compatible with these programs, taking advantage of the features they already possess by using the files they produce as inputs rather than inventing a new proprietary data format specific to this purpose. As described in §3.3, creating such compatibility with both FLEx and ELAN was a challenge since the underlying XML data structures that they use are quite different.

Another place where principle (iii) is manifested is in the ways in which individual practitioners use ELAN and FLEx in the first place. This is most relevant for ELAN files, since ELAN allows users significant freedom to create different tier structures with customized names. Previous tools for displaying ELAN transcriptions in synchrony with media have limited the display to one or two chosen tiers at once. This holds true for ELAN's built-in feature for creating subtitled videos, as well as for separate tools such as Schroeter and Thieberger (2006)'s EOPAS and Dobrin and Ross (2017)'s ETST, which also requires specific tier names. Unlike these existing tools, LingView accepts any format of ELAN file and displays whatever tiers a user provides. Beyond merely being more flexible, this also helps contribute better "backward compatibility" in the sense that it allows practitioners to use LingView with existing ELAN files without having to convert them to a specific format or rename tiers to specific names. The need for such standardization may be a mere annoyance in some cases, but could be a more fundamental problem for some projects such as collections of texts featuring multiple languages either across or within files (e.g. cases of code-switching annotated on distinct tiers).

(iv) Practitioner-friendly pipeline

One of the challenges of language documentation is keeping track of the large body of data of different kinds of primary and secondary files generated in the documentation process. With this in mind, we have designed LingView to have some tools for managing this. First and foremost, this principle is seen in the inclusion of scripts which allow for the input of metadata about the files and from this, automatically generates an index of the available media.[4] Beyond this, we include scripts for batch editing, deleting, and uploading files to LingView sites (see §4). While many language documentation practitioners are quite technically adept, the scripts (and the workflow more generally) are intended to be usable without any particular technical expertise.

While these four design principles are motivated in part by the particular challenges we faced in the context of our particular project, we hope it is clear that the sorts of challenges we faced are

[4] We hasten to point out that despite these indexing and metadata features, LingView should in no way be considered to be an alternative to the use of digital archives such as ELAR, AILLA, PARADISEC, etc.

fairly typical of community-based language documentation. There is one additional design consideration which falls into this category as well and does not fit neatly within (i-iv): the lack of access to high speed internet. While some A'i have regular access to internet, many do not. Some towns, such as Zábalo, have no internet access of any kind, including cellular network coverage. While it is reasonable to think that internet access will only become more prevalent in coming years, the extent and speed with which this will happen and the extent to which this will be affordable (and desirable) to potential end-users is unclear. We therefore designed LingView to be fully usable offline with files locally stored on an end-user's computer or on a computer located in a school or community center. In cases where the computer has inadequate space to store many hours of video, our software is adaptable using the edit and delete scripts (4.4 and 4.5) to selectively remove files from a particular instance of the site.

3 How does the system work?

3.1 Overview of data pipeline

LingView allows users to view the disparate ELAN and FLEx filetypes in a single web interface. This is accomplished via a multi-step data pipeline, depicted in Figure 5 below, which allows us to merge ELAN and FLEx files into a proprietary JSON format. This JSON format is then converted into HTML by the front-end ReactJS code.

<Figure 5>

This project can be divided into two major components: the preprocessing work, which converts .eaf and .xml files into the proprietary JSON format, and the front-end work, which allows this JSON data to be displayed on the web. While ELAN and FLEx both use XML-like structures for data storage, the schemata are quite different. For example, ELAN internally orders data by the tier (e.g., transcription, gloss, translation), while FLEx orders data by the sentence. The first major component involves combining these two schemata into a new filetype and will be discussed in Section 3.3. The second major component, i.e., displaying the JSON format as a interactive website, will be discussed in Section 3.4. Before doing this, we briefly motivate this choice of data pipeline.

3.2 Motivating the pipeline

As mentioned, ELAN and FLEx use different data schemata. This is partly due to the fact that ELAN files (.eaf) are time-aligned, while FLEx files are not; it is also partly due to quirks in the software which cause data and metadata to be stored differently. To efficiently display these files on a website, the data must be rearranged and condensed into a single format.

However, combining these into a single file format is not an easy task. The data pipeline is constrained by (1) an avoidance of information loss, (2) the ability for the website to load quickly and correctly on various platforms, and (3) the ability for the website to run offline. Combined, these constraints present an issue: the data processing task is computationally expensive, and the user may not be connected to a server which can perform these computations. To resolve this, we must re-consider the prototypical client-server model.

Generally, websites are broken into a back-end and a front-end. Back-end code runs on a server and consequently has a large amount of processing power; front-end code, however, runs in the browser and has significantly less processing power. Since this website is meant to run offline (i.e., without a server) on a variety of devices, the code that runs when a user accesses the website must be relatively lightweight, front-end code. However, the code to convert ELAN and FLEx files into a single JSON format is not lightweight, and may require looping through files multiple times to re-organize the inherent tier structure. If this code were to run in a browser, the website would be very slow, bordering on unusable. It is for this reason that the conversion to JSON occurs offline, in a step called "preprocessing." This step involves running a suite of Node.js scripts locally to create JSON files which may then be uploaded to the website. Since this happens offline, it is not constrained by the speed requirements detailed above.

3.3 Step 1: preprocessing

As mentioned, the preprocessing code is a suite of Node.js scripts that allow the user to convert FLEx and ELAN files into a single JSON format. Details of how to use these scripts will be discussed in Section 4. This section will simply describe how the scripts work and motivate the data schema. The full details of the preprocessing scripts are too long to explain here, so we will present only a sampling of the work done in this section.[5] We'll explain how preprocessing works by discussing the JSON data schema, a portion of which is shown in Figure 6.

<Figure 6>

3.3.1 Metadata

The first portion of this schema is the metadata. Inside this section, we have the `timed` value. This is set to true for ELAN files (which have timestamps), and false for FLEx files (which do not). Next, we have media `audio` and media `video`. These will be empty strings for FLEx files, but for ELAN files they will contain the names of files contained in the `media_files` directory. When the preprocessing scripts are run, they will search `media_files` for matching audio and video files, which will be linked here.

Then, we have other metadata like the author, title, date, and so on. This will be displayed in the sidebar that appears alongside each story. Some of this metadata appears in ELAN and FLEx files, and the scripts gather this data. However, additional data can be entered by running the `edit.js` script, which allows the user to edit a variety of metadata fields. Once again, usage will be discussed in Section 4.

Finally, we have the lists of speaker and tier IDs. Speaker IDs give a short abbreviation like "S1" and "S2," as shown in Figure 7. Meanwhile, the tier ID list is used to generate tier checkboxes which allow the end-user to show or hide certain tiers. For example, an end-user may opt not to view the translation tier, or the gloss tier. These tier names are generated based on the tier names in corresponding FLEx/ELAN files, but they may be changed.[6] Finally, these tiers may either be

[5]Full details can be found in the in-line documentations of the `preprocess_eaf.js` and `preprocess_flex.js` files, both of which can be found in the `preprocess` directory.

[6]The simplest way to do this would be to change tier names in the original ELAN and FLEx files. However, tier names may also be edited during the preprocessing stage, by modifying `preprocessing/preprocess_eaf.js` or `preprocessing/preprocess_flex.js`. Finally, they may be edited on the client side by modifying the

subdivided or not subdivided.

<Figure 7>

3.3.2 Aside: tier subdivision

Understanding tier subdivision is crucial to knowing how this software works. Each sentence is broken into a number of tiers, such as the transcription, gloss, and the translation. The transcription tier is normally not subdivided, i.e., a single string represents the entire sentence. However, a gloss tier is normally subdivided. That is, the gloss tier contains a list of separated strings corresponding to each word in the text. We use data from FLEx and ELAN to determine if a given tier is subdivided or not.

Based on the subdivided tiers, each sentence is broken into some integer number of slots. For example, say a sentence is broken into 3 slots. Then a non-subdivided tier would span all 3 slots; however, a subdivided tier would contain a list of values, perhaps one from slot 0-1, another from slot 1-2, and another from slot 2-3. This allows us to represent each gloss as an HTML table in Step 2 (Section 3.4).

In ELAN, some subdivided tiers come with specific timing information, while some ELAN tiers and all FLEx tiers do not. To simplify the user interface, LingView displays timestamps only for the beginning of each sentence.

3.3.3 Sentences

We'll now move onto the `sentences` from Figure 6. This is a list, generated either from independent ELAN annotations or from FLEx sentences. Each sentence has some associated data, such as the speaker, tier, and start/end time. As discussed above, each sentence also has a parameter `num_slots`, as well as the high-level text (usually a transcription).

Then, each sentence also has a number of dependent tiers, which as discussed, may or may not be further subdivided. If they are subdivided, the `start_slot` and `end_slot` parameters will be used to store slot values. Either way, they will contain some list of `values`, where the actual text is stored in `value`. A single sentence from this `sentences` list will be rendered into a gloss resembling that in Figure 7.

Within each sentence, tiers are displayed in the same order as in FLEx or ELAN. In FLEx, tiers are ordered based on their function (transcription first, then morphemes and their glosses, then free glosses). ELAN, however, allows its users to hide or reorder tiers at will, and it stores the current configuration in a separate .pfsx preferences file. Practitioners may omit this file to use a default order, or they may edit the tier configuration in ELAN, which automatically creates or updates the preferences file. The default order is reasonable on many ELAN files, but not for ELAN files which were exported from FLEx. Exporting from FLEx to ELAN does not lose information, but it produces additional unwanted tiers, some of which are parents of wanted tiers. Thus, for audio/video files annotated using FLEx's computer-assisted glossing or multiple-orthography tools, the ability to hide tiers is particularly important.

`TierCheckboxList.jsx` file.

3.3.4 The `database.json` file

All texts, represented in the schema described in Figure 6 above, will be stored in the `database.json` file. This file simply contains a list of stories, represented as above, along with an index that is used to generate the index of stories. This file is used to generate the site. However, the individual JSON representations may be viewed in `data/json_files`.

3.4 Step 2: the web display

The website frontend is built with ReactJS, which is a component-based Javascript library for building user interfaces. Similar to a templating language, ReactJS excels at converting documents (in our case, the JSON files) into HTML representations. Being component-based means that we can write code for separate components, such as the metadata sidebar, the tier checkboxes, and a glossed sentence. Given the abstract code for each of these components, we can read the JSON documents and create the specifically required components as desired. Some of these components are shown in Figure 8, which demonstrates how a sentence is broken down into a `LabeledTimeBlock`, `TimeBlock`, etc.

<Figure 8>

With a few exceptions, each file inside the `jsx` directory corresponds to one such component. A component may contain either another component, a list of components, or no components at all. For example, the `TimedTextDisplay` component, which is used when displaying ELAN files, is broken down into numerous `LabeledTimeBlock`s. Each of these contains a timestamp and some number of `LabeledSentences`. The highest level component, `AppContainer`, handles retrieving the JSON data from the `database.json` file. This data is passed from component to subcomponent until it is displayed on the site. Finally, these JSX files are converted into Javascript with Webpack, a bundling tool that converts the dozens of JSX files into a single `bundle.js` file.

3.4.1 Time-alignment for ELAN files

The web display also features time syncing between the displayed text and any associated audio/video. Since ELAN provides timestamps for each annotation, we highlight the corresponding `LabeledTimeBlock` whenever it matches the current media timeslot. This is done in `js/txt_sync.js`. Additionally, whenever a user clicks on a timestamp in the text, the media scrubber jumps to that time value.

4 Workflow for using the system

This section describes the workflow for building and maintaining a collection of texts in LingView. Additional guides and troubleshooting may be found at `https://github.com/BrownCLPS/LingView/wiki`.

4.1 Installing the Software

4.1.1 Download LingView

To use the software, begin by downloading the repository stored at `https://github.com/BrownCLPS/LingView.git` and extracting the zip file. You may place the extracted file anywhere on your computer, but to follow along with these instructions, we recommend extracting to your Desktop and renaming the extracted directory from "LingView-master" to simply "LingView". Open the extracted directory in File Explorer (on Windows) or Finder (on Mac). At this point, you will be able to view sample files by opening the `index.html` file with your browser.[7] The LingView download comes with two sample files, named "Intro" and "Singo a'i." The former was imported from ELAN and has associated audio/video, while the latter was exported from FLEx and therefore does not have any attached media.

4.1.2 Download Node.js and NPM packages

If you do not have Node.js installed, you will need it to run the preprocessing scripts. Instructions for downloading Node can be found at `http://blog.npmjs.org/post/85484771375/how-to-install-npm`. Once you have installed Node.js and NPM (both of which are included in the previously linked download), you will need to install some packages. These are listed in the `package.json` file. Simply navigate to the root LingView directory, as described in §4.6, and run

```
npm install
```

This will read the required packages from the `package.json` file and install them for you. It is preferable to use NPM version 4.6.x for this process.

4.2 Adding files

This section describes the steps for displaying your own ELAN and FLEx files using the LingView software.

4.2.1 Export and place files

For ELAN, the program automatically generates .eaf files. Simply copy these files to the `data/elan_files` directory. For FLEx, you must export the files into the correct XML format. To do so, open the file in FLEx, go to the "Analyze" tab, and select "File > Export Interlinear > Verifiable generic XML." Save the resulting .xml file in the `data/flex_files` directory.[8]

Place any corresponding media files (WAV, MP3, or MP4) into the `data/media_files` directory. Ensure that media files use web-friendly encoding, e.g., the H.264 codec for MP4 files.

[7] As of this writing, Google Chrome's same-origin policy will prevent the website from loading without a server. To get around this, you may choose to run a Python SimpleHTTPServer (or equivalent) from the root LingView directory. Alternatively, the website will run offline without a server in Firefox or Safari.

[8] An alternative approach would involve running preprocessing scripts directly on the original FLEx database, rather than exporting FLEx files as described above. However, our chosen method offers users the flexibility to choose which files they wish to display and simplifies the preprocessing code.

Note that, for large collections, copying all media files into the web directory may be inefficient. Instead, we recommend creating symbolic links to these files and ensuring they are placed in a web-accessible directory. Symbolic links may also be used for FLEx and ELAN files but web-accessibility of the original files is not required.

To avoid unintended behavior, confirm that all filenames are unique.

4.2.2 The rebuild script

Once the ELAN, FLEx, and media files are in place, run the following command as described in §4.6:

```
node preprocessing/rebuild.js
```

This will run both ELAN and FLEx preprocessing scripts on every available file. For ELAN files, the script will attempt to find matching audio or video files. Importantly, this script is also responsible for rebuilding the site index. Therefore, we recommend running the rebuild script after deleting or editing any files.

After this process, when you open the `index.html` file with your browser, you will see your ELAN and FLEx files included in the index. You may repeat this process whenever you wish to add additional ELAN, FLEx, or media files.

4.3 Editing files

To update files, make the desired changes using ELAN or FLEx, save the result to the corresponding `data/elan_files` or `data/flex_files` directory, and then run the rebuild script as described in §4.2.

4.4 Editing metadata

The website can show metadata for each story, including title, author, description, and other fields. This data, which is displayed on the story index and on the individual story pages, can be added or changed using the `edit.js` script. To use this script, first open `index.html` in your browser, navigate to the story you want to edit, and locate the unique ID associated with that story. This is a 36-character string found at the end of the story URL. For example:

https://brownclps.github.io/LingView/#/story/$\boxed{\text{97b8ab3b-d2a5-428a-aa68-0aa304ba1c44}}$

With this unique ID copied to your clipboard, return to your terminal and type the following command:

```
node preprocessing/edit.js unique_id
```

where `unique_id` is replaced with the 36-character string described above. This script will offer a number of prompts, allowing you to edit the story title, description, etc. Repeat this process with any other stories whose metadata you wish to edit. After making these changes, run the rebuild script again so that they will appear on the LingView site.

4.5 Removing files

To remove an ELAN or FLEx file, first locate its unique ID as described in §4.4. With this unique ID copied to your clipboard, return to your terminal and type the following command from the root project directory:

```
node preprocessing/delete.js unique_id
```

where `unique_id` is replaced with the 36-character ID. If the file has associated media, you will be prompted to delete or retain the associated media files. Repeat this process for any other stories you wish to delete. Finally, run the rebuild script described in Section 4.2.2 so that your changes will appear in the index.

To remove a media file from the LingView site while leaving its ELAN file in place, delete the media file from the `data/media_files` directory and then run the the rebuild script for the changes to appear on the site.

4.6 The terminal

The text commands in this section must be run from a *terminal*, a text-based interface, which comes preinstalled on all Mac, Windows, and Linux computers. On Mac, the Terminal app can be found using Spotlight search. On Windows, the relevant app is called Windows PowerShell,[9] and it can be found by tapping the Windows key to bring up the start menu, and then searching for the app's name. After opening the terminal, navigate to the root LingView directory using the "cd" command. For example, if you saved the LingView directory on your Desktop with the name "LingView":

```
cd ~/Desktop/LingView
```

At the end of each command you type in the terminal, tap the Enter or Return key to execute the command.

4.7 ReactJS and Webpack

Most of the JavaScript code that runs on the site is stored in `js/bundle.js`. This code was automatically generated by Webpack, a module bundler which converts the ReactJS files (in .jsx format) into a single JavaScript file. To make edits to the front-end/UI code, we recommend editing the ReactJS files which are contained in the `jsx/` directory. However, since the website reads from the `bundle.js` file, this will not cause immediate changes on the site. Rather, you will need to re-bundle with the following command:

```
npm run webpack
```

This will update the `bundle.js` file and modify the site accordingly.

[9]Not Windows PowerShell ISE.

5 Future directions

In this paper, we have presented LingView, a web interface for viewing annotated linguistic materials from ELAN and FLEx, along with a set of scripts for managing an installation of LingView. In developing LingView, we aimed to follow four design principles, discussed in §2.2: (i) multimedia compatibility, (ii) end-user customizability, (iii) linguist-friendly input, and (iv) practitioner-friendly pipeline. The resulting system relies on non-proprietary linguist-friendly inputs from FLEx and ELAN, but is flexible in terms of the specific ways in which those inputs are structured (e.g. no specific tier names or structures are required in ELAN).

While this flexibility is necessary, it does have some undesirable consequences, at least as presently constituted. Since LingView imposes no constraints on the tier names or their structure, it also does not "understand" tier names or relationships between them which may be obvious to a human user. For example, if an ELAN file has two tiers named "S1_SpanishTranslation" and "S2_SpanishTranslation" respectively, the software will not recognize their commonality in any way. So, while an end-user would presumably never wish to see the Spanish translation of one speaker but not the other, there will still be separate checkboxes for the two tiers. This can create a cluttered appearance in the UI for texts with many speakers. An individual installing a LingView site could add additional code to address this for their particular tier-naming scheme, but no general solution exists at present and so better support for media with multiple speakers is an area ripe for future development.

While such data could potentially be extracted from the tier type fields, we found this information was frequently missing or incorrect in practitioners' existing ELAN files. Automatically simplifying the UI based on inconsistent or incorrect tier information would lead to confusing results and, on some ELAN files, would remove important flexibility from the user interface. For example, if the Speaker 1 Spanish gloss is accidentally given the same tier type as the Speaker 2 English gloss, then the UI would make it impossible to show either of them while hiding the other. To avoid such problems, the current version of LingView ignores tier type fields and treats each tier individually.

This example of unused metadata is representative of a whole suite of possible extensions which leverage additional metadata information. Synthesizing information from metadata formats such as IMDI (Broeder and Wittenburg, 2006) and CMDI (Broeder et al., 2012) would be one such promising approach but would require interface modifications to display the new metadata fields.

Two other potential future directions are worth mentioning, but would require significant additional effort to achieve. First, for many projects, it will be important to incorporate a system of permissions/access restrictions for materials to which access is to be restricted, similar to archives such as ELAR. Second, for many projects, concordance and other search features are ones which would be of great use to linguists and community members alike. While the single unified JSON format lays groundwork for this in some sense, there also is a fundamental tension between the need for uniformity which advanced search features require and the flexibility to differently structured files which we have considered a design feature in the creation of LingView.

Finally, one common question we have received is the possibility for use on mobile devices such as tablets and phones. Since the site is run in a browser window, it should in principle work on any device with a suitable browser, especially since navigating LingView sites is done primarily through links rather than via typing. For phones, however, the smaller screen size means that the site's functionality is significantly diminished. While development of a dedicated mobile version (or an app) would be a worthy future endeavor, such development appears at present to be separate enough from the present incarnation of LingView that it might be better regarded as a separate project than a future direction for LingView per se.

References

Blaser, M. and E. Chica (2008). *Mitos del pueblo cofán: A'indeccu canqque'sune condase'cho*. Centro Cultural de Investigaciones Indígenas Ecuador, Padre Ramón López.

Borman, M. (1962). Cofán phonemes. In *Studies in Ecuadorian Indian Languages I*, pp. 45–59. Summer Institute of Linguistics of the University of Oklahoma.

Borman, M. (1976). *Vocabulario cofán: Cofán-castellano, castellano-cofán*. Instituto Lingüístico de Verano (Summer Institute of Linguistics).

Borman, M. (1977). Cofán paragraph structure and function. In R. Longacre (Ed.), *Discourse grammar: studies in indigenous languages of Colombia, Panama, and Ecuador*, pp. 289–338. Summer Institute of Linguistics.

Borman, M. (1990). *La cosmología y la percepción histórica de los cofanes de acuerdo a sus leyendas*. Instituto Lingüístico de Verano (Summer Institute of Linguistics).

Broeder, D., M. Windhouwer, D. Van Uytvanck, T. Goosen, and T. Trippel (2012). CMDI: a component metadata infrastructure. In *Describing LRs with metadata: towards flexibility and interoperability in the documentation of LR workshop programme*, Volume 1.

Broeder, D. and P. Wittenburg (2006). The IMDI metadata framework, its current application and future direction. *International Journal of Metadata, Semantics and Ontologies 1*(2), 119–132.

Cepek, M. (2018). *Life in Oil: Cofán Survival in the Petroleum Fields of Amazonia*. University of Texas Press.

Comunidad Cofán de Zábalo and Comunidad Cofán Pakuya (2011). *La gente del río – Naensu ai*. Imprenta Rimana.

Dobrin, L. M. and D. Ross (2017). The IATH ELAN Text-Sync Tool: A Simple System for Mobilizing ELAN Transcripts On-or Off-Line. *Language Documentation & Conservation 11*, 94–102.

Fischer, R. and K. Hengeveld (t.a.). Cofán (A'ingae). In Patience Epps & Lev Michael (eds), Amazonian Languages, An International Handbook. Berlin: de Gruyter Mouton.

Fischer, R. and E. van Lier (2011). Cofán subordinate clauses in a typology of subordination. In *Subordination in Native South American Languages*, pp. 221–249. John Benjamins.

Hengeveld, K. and R. Fischer (msa). A'ingae (Cofán/Kofán) operators. Intended for: Kees Hengeveld & Hella Olbertz (eds), Systems of tense, aspect, modality, evidentiality and polarity in Functional Discourse Grammar. Special issue of Open Linguistics.

Hengeveld, K. and R. Fischer (msb). A'ingae (Cofán/Kofán) as a transparent language. Intended for: Kees Hengeveld (ed.), Studies in Transparency. Special issue of Linguistics in Amsterdam.

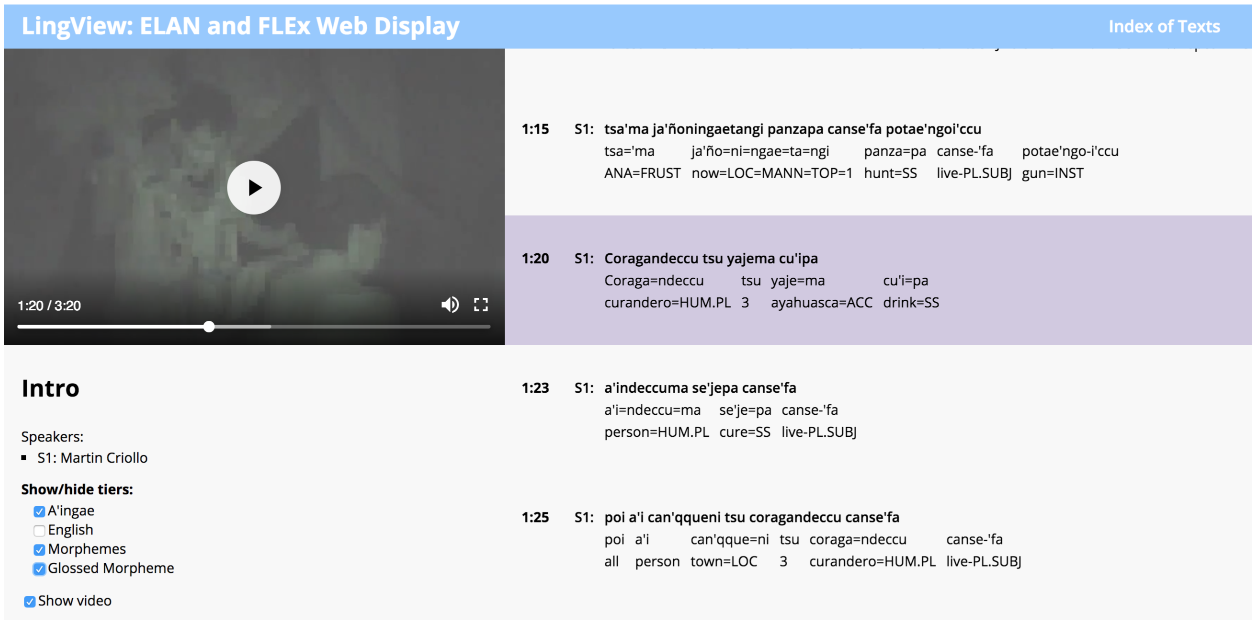
Max Planck Institute for Psycholinguistics (2018). *ELAN*. The Language Archive, Nijmegen, The Netherlands. Version 4.1.2. URL: http://tla.mpi.nl/tools/tla-tools/elan/.

Repetti-Ludlow, C., H. Lucitante, H. Zhang, S. AnderBois, and C. Sanker (ms). A'ingae (Kofán). Submitted to JIPA's illustrations of the IPA.

Schroeter, R. and N. Thieberger (2006). EOPAS, the EthnoER online representation of interlinear. *ResearchGate*, 1–19.

SIL FieldWorks (2018). FieldWorks Language Explorer. `https://software.sil.org/fieldworks/`.

Wittenburg, P., Brugman, H., Russel, A., Klassman, A. and Sloetjes, H. (2006). ELAN: a Professional Framework for Multimodality Research.



LingView: ELAN and FLEx Web Display

Index of Texts

1:20 / 3:20

Intro

Speakers:

- S1: Martin Criollo

Show/hide tiers:

- ☑ A'ingae
- ☐ English
- ☑ Morphemes
- ☑ Glossed Morpheme

☑ Show video

1:15 **S1:** **tsa'ma ja'ñoningaetangi panzapa canse'fa potae'ngoi'ccu**

tsa='ma	ja'ño=ni=ngae=ta=ngi	panza=pa	canse-'fa	potae'ngo-i'ccu
ANA=FRUST	now=LOC=MANN=TOP=1	hunt=SS	live-PL.SUBJ	gun=INST

1:20 **S1:** **Coragandeccu tsu yajema cu'ipa**

Coraga=ndeccu	tsu	yaje=ma	cu'i=pa
curandero=HUM.PL	3	ayahuasca=ACC	drink=SS

1:23 **S1:** **a'indeccuma se'jepa canse'fa**

a'i=ndeccu=ma	se'je=pa	canse-'fa
person=HUM.PL	cure=SS	live-PL.SUBJ

1:25 **S1:** **poi a'i can'qqueni tsu coragandeccu canse'fa**

poi	a'i	can'qque=ni	tsu	coraga=ndeccu	canse-'fa
all	person	town=LOC	3	curandero=HUM.PL	live-PL.SUBJ

Figure 1: LingView UI with time-synced video

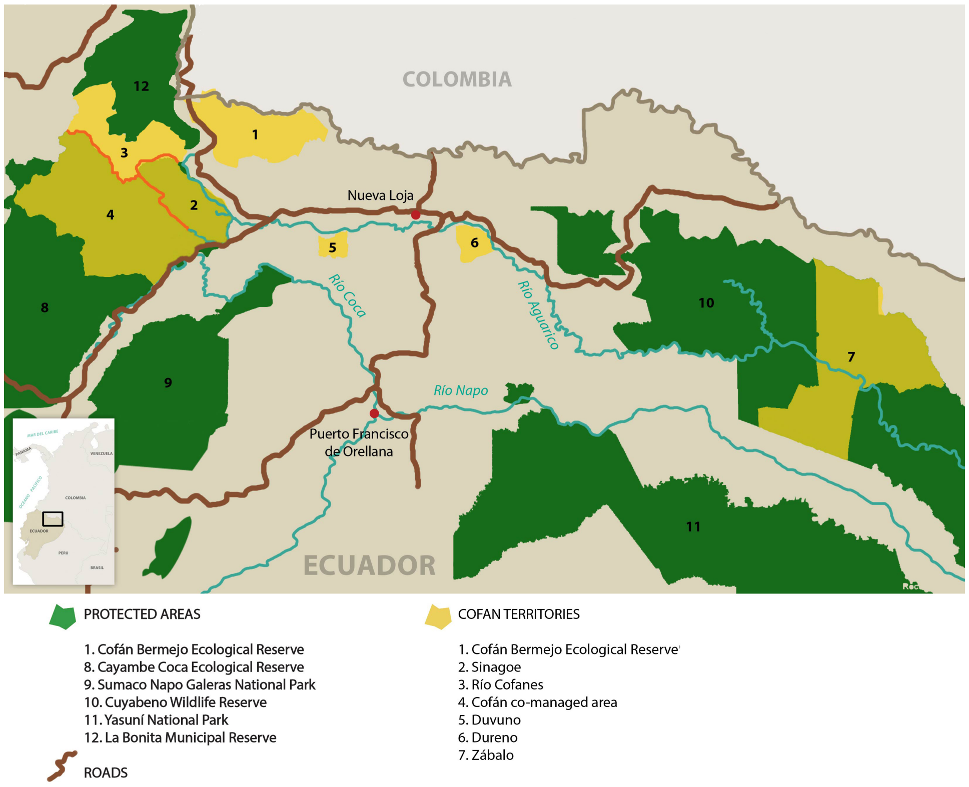
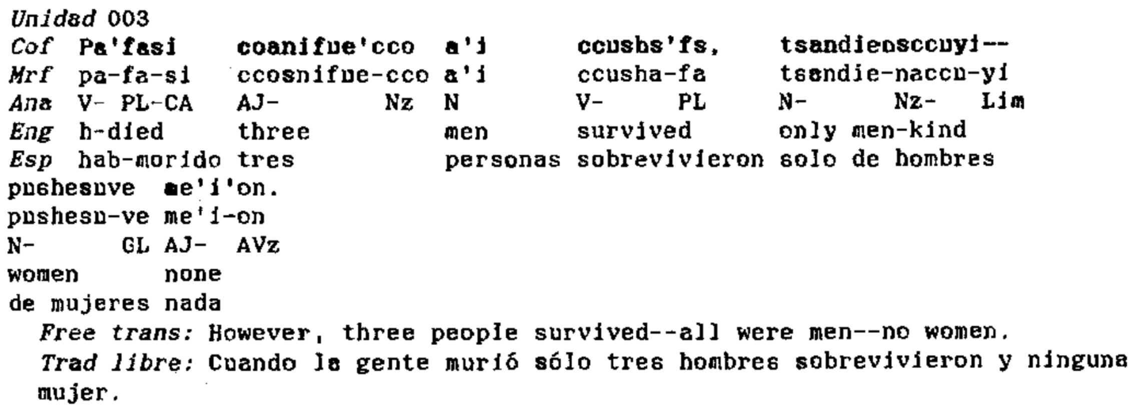
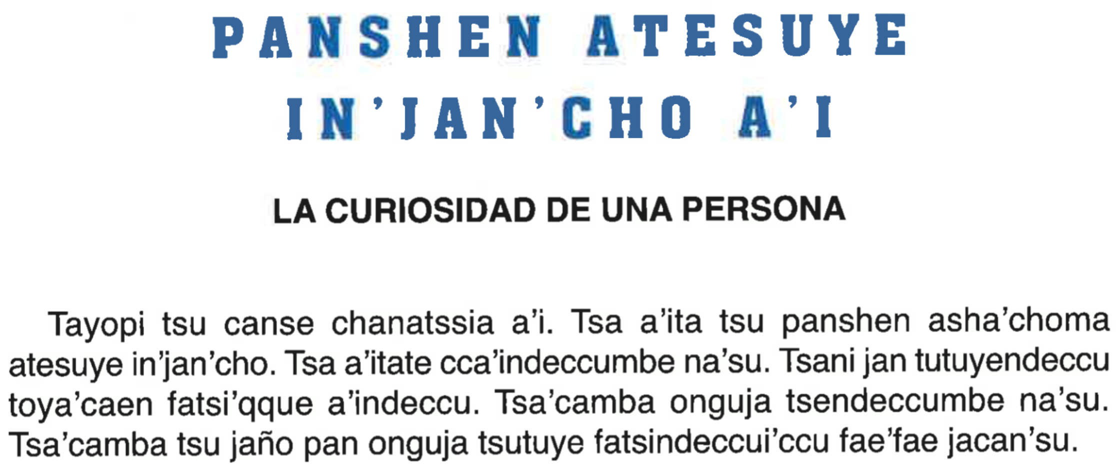

COLOMBIA
12
3
1
4
2
Nueva Loja
5
6
Río Coca
Río Aguarico
8
10
7
9
Río Napo
Puerto Francisco de Orellana
11
ECUADOR
PROTECTED AREAS
1. Cofán Bermejo Ecological Reserve
8. Cayambe Coca Ecological Reserve
9. Sumaco Napo Galeras National Park
10. Cuyabeno Wildlife Reserve
11. Yasuní National Park
12. La Bonita Municipal Reserve
ROADS
COFAN TERRITORIES
1. Cofán Bermejo Ecological Reserve
2. Sinagoe
3. Río Cofanes
4. Cofán co-managed area
5. Duvuno
6. Dureno
7. Zábalo

Figure 2: Map of A'i territories in Ecuador



```
Unidad 003
Cof  Pa'fasi     coanifue'cco  a'i      ccusbs'fs,     tsandieosccuyi--
Mrf  pa-fa-si    ccosnifue-cco a'i      ccusha-fa      tsendie-naccu-yi
Ana  V- PL-CA    AJ-        Nz N        V-      PL     N-      Nz-   Lim
Eng  h-died      three         men      survived       only men-kind
Esp  hab-morido  tres          personas sobrevivieron  solo de hombres
pushesuve  me'i'on.
pushesu-ve me'i-on
N-      GL AJ-  AVz
women      none
de mujeres nada
  Free trans: However, three people survived--all were men--no women.
  Trad libre: Cuando la gente murió sólo tres hombres sobrevivieron y ninguna
  mujer.
```

Figure 3: A one-line excerpt from Borman (1990)



PANSHEN ATESUYE IN'JAN'CHO A'I

LA CURIOSIDAD DE UNA PERSONA

Tayopi tsu canse chanatssia a'i. Tsa a'ita tsu panshen asha'choma atesuye in'jan'cho. Tsa a'itate cca'indeccumbe na'su. Tsani jan tutuyendeccu toya'caen fatsi'qque a'indeccu. Tsa'camba onguja tsendeccumbe na'su. Tsa'camba tsu jaño pan onguja tsutuye fatsindeccui'ccu fae'fae jacan'su.

Figure 4: A brief excerpt from Blaser and Chica (2008)

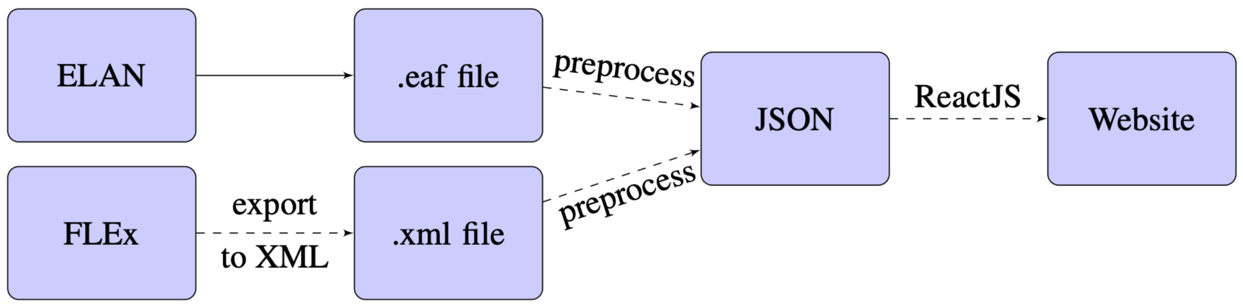


Figure 5: Data pipeline from FLEx/ELAN to website display

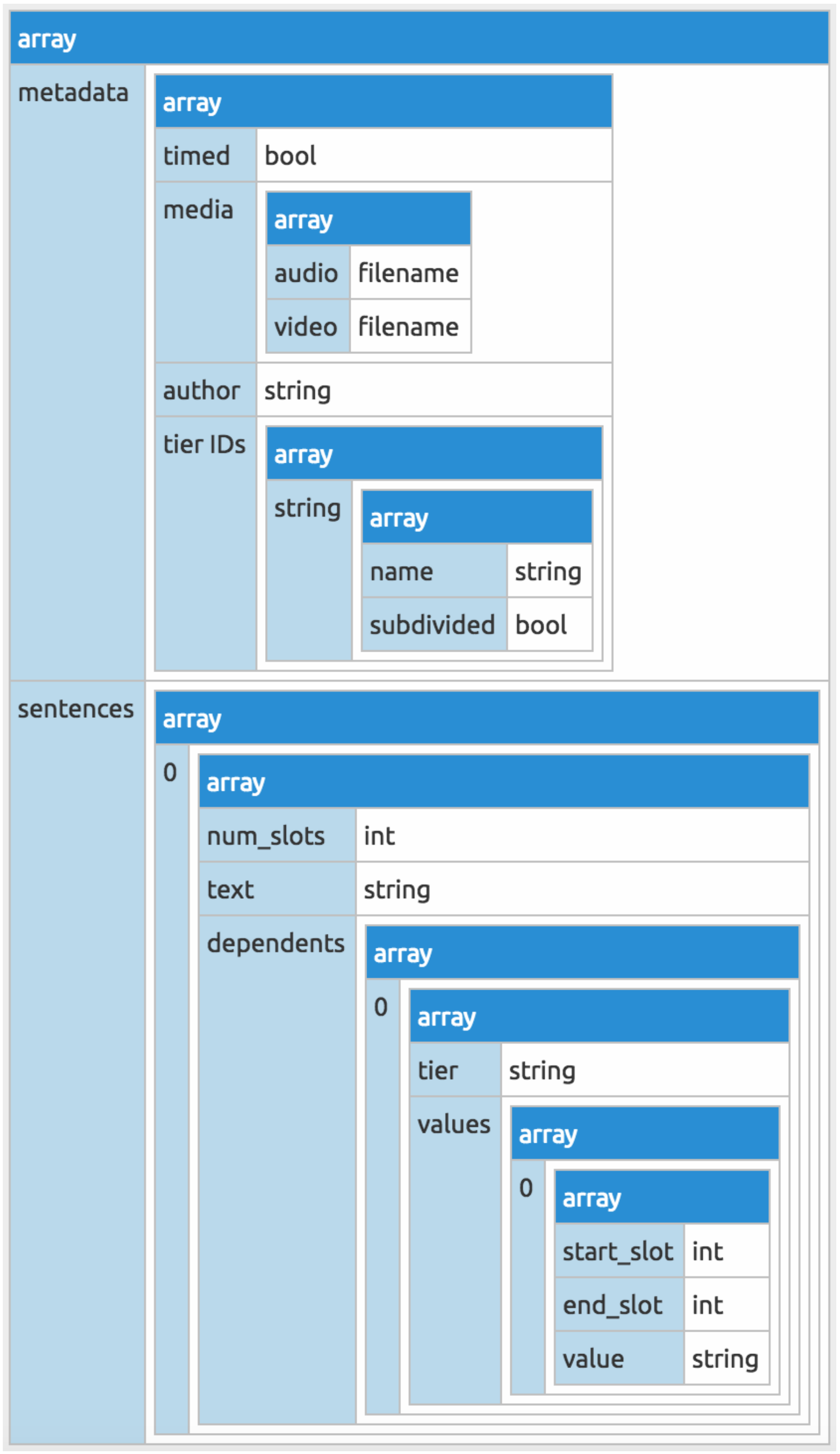
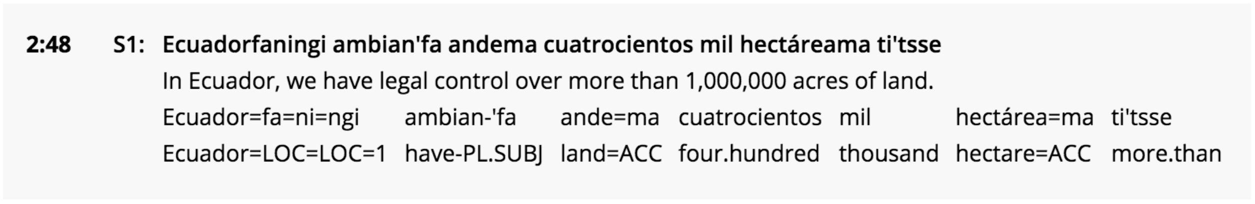


Figure 6: Selections from the JSON data schema



2:48 **S1:** **Ecuadorfaningi ambian'fa andema cuatrocientos mil hectáreama ti'tsse**

In Ecuador, we have legal control over more than 1,000,000 acres of land.

Ecuador=fa=ni=ngi	ambian-'fa	ande=ma	cuatrocientos	mil	hectárea=ma	ti'tsse
Ecuador=LOC=LOC=1	have-PL.SUBJ	land=ACC	four.hundred	thousand	hectare=ACC	more.than

Figure 7: Sample glossed display

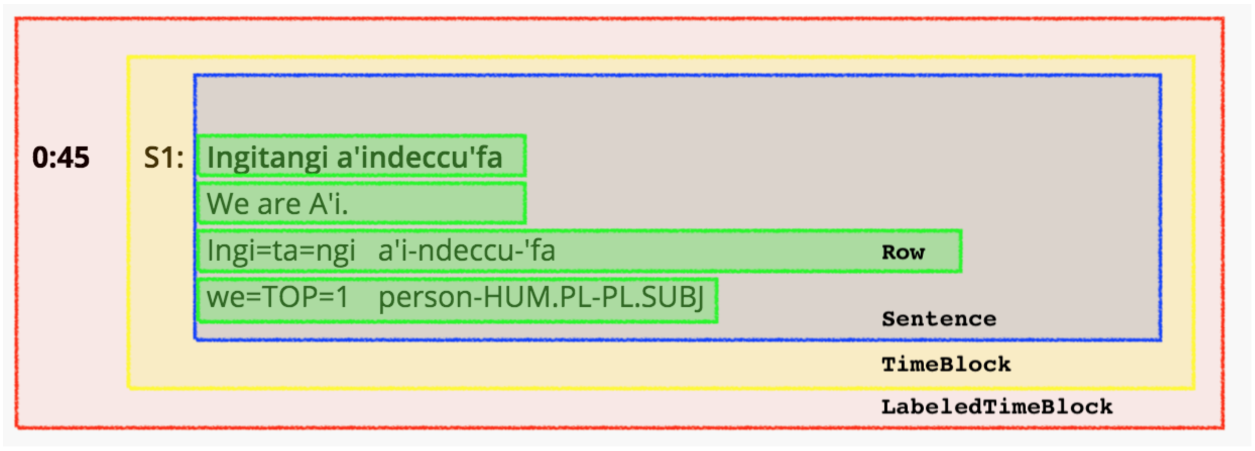


Figure 8: Example components