CLD: Software for Computational Language Description

Steven Abney

$\boxed{\text{DRAFT}}$ of 2019 Aug 3

1 Motivation

I would like to describe an application for computational language description, acronymically named CLD. The ultimate motivation for CLD derives from an envisioned paradigm of linguistic research, one that accepts the Chomskyan postulate that an account of the human capacity for language boils down to the question of unsupervised language learning, but approaches the question from a computational perspective, which is to say, from the perspective of data science and machine learning. Within this paradigm, the question of how a language can be learned *in principle* is considered to be prior to the psychological question of the properties and time course of human language acquisition, and an answer is considered to be unsatisfactory unless it is explicit enough to be implemented and tested on large, systematic data collections. Bloomfield hinted at something along these lines when he wrote that "we shall have to return to the problem of general grammar and to explain [the] similarities and divergences [among languages], but this study, when it comes, will be not speculative but inductive" [5]. Accordingly, I call the paradigm **inductive general grammar.**

The core of a theory of inductive general grammar is a learning algorithm. Given data from any natural language, it outputs a language description that is correct, which from a computational perspective means that it defines a relation between sentences S and semantic representations μ (sounds and meanings) that aligns with human judgments about expression and interpretation: whether S can be interpreted as μ, or, conversely, whether μ can be expressed as S.

A typical machine-learning experiment involves comparing the performance of a few learners on a given dataset. That is not sufficient for our purposes. We are not much interested in how well the learner does on a single language; what we really wish to know is how well it does on all possible human languages. To estimate that, we require a sufficiently large and sufficiently representative sample of languages, and for each, a dataset that is sufficiently large to learn the entire language. In short, we require a **universal corpus** of the world's languages [1].

In recent years, the computational-linguistic community has made progress toward the development of a universal corpus. The most significant resource is a set of treebanks that uses a common annotation scheme known as the **Universal Dependencies (UD)** framework [6]. The UD treebanks currently encompass some 70 languages. That is an order of magnitude more than were available a decade ago, but even so, it represents only about one percent of the world's languages, and endangered languages are heavily undersampled, particularly when one limits attention to the larger treebanks.

It has proven difficult to obtain suitable data for the remaining 99% of the world's languages, for which I use the term *low-resource languages* as a shorthand. But there are reasons to doubt that the problem is one of making audio recordings. Bird has shown that relatively large bodies of audio recordings can be collected in brief time periods [2, 3, 4]. Anecdotally, two Ojibwe tribes that I have interacted with have made audio-video recordings of immersion sessions for educational purposes, and as a result have accumulated thousands of hours of language data over the years. Such observations suggest that the culprit is not any difficulty in making primary recordings, but rather the low throughput of the standard documentary pipeline that leads from primary recordings to finished datasets. The standard tools used in documentary linguistics are sophisticated but place high demands on users; they typically emphasize finesse in annotation over streamlining and ease of use. There are few tools that emphasize speed and simplicity over sophistication of annotation, and CLD is designed to fill that gap. In particular, I hope to enable speakers of the language to contribute directly to the effort of documenting their language, and thereby to increase the size and diversity of a universal corpus.

The question arises immediately what might motivate a language speaker to contribute to a universal linguistic corpus. We cannot reasonably expect speakers of low-resource languages to be motivated by the rather esoteric aims of academic linguistic research. But speaker communities—particularly in the cases of languages in which transmission to the next generation is faltering—often do have a strong interest in assembling linguistic data for purposes of language instruction and preservation.

That provides the central design goal of the CLD application: to support a *mutually beneficial* collaboration between computational linguistics and speaker communities. The application is designed in equal parts as a tool that assists in human language learning, and as a platform for research in machine language learning. If the collaboration brings benefit to the community, it is more likely that they will be willing to release at least some vetted portion of the data for research use, and even in the absence of released data, computational-linguistic goals are served by incorporating language-learning algorithms and their evaluation into the platform. One may view the latter approach as bringing the algorithms to the data rather than the data to the algorithms.

Fortunately, solutions to some of the most challenging computational problems would be of genuine and significant benefit for language instruction. Automatic audio transcription is a key example, and automated analysis and translation are also of considerable utility. The first version of CLD does not incorporate solutions to the automated learning problems, but it does provide a platform for integrating them into language instruction.

Among speaker communities, immersion learning appears to be widely popular. CLD does not provide a traditional system-initiative instructional system like Duolingo or Rosetta Stone; rather, it particularly aims to support self-study that complements immersion learning.

2 Content

2.1 Simple texts

CLD is organized around texts, which mediate between audio recordings and the lexicon. The basic item is a **column** of tokenized text. Structurally, a text column is a list of sentences, optionally aligned with a translation that is a list of the same length. The translation can be viewed as a second column, so that the container, called a **simple text,** has the form of an array: its first column is the original text, the second column is the translation, and each row represents a sentence. A simple text is completely analogous to traditional facing-page format for text and translation.

Sentence is to be understood loosely: **translation unit** is a more accurate term. Nothing hinges on its size or grammatical status, only on it being a suitable unit for translation.

A sentence, in turn, is a list of word tokens. Only word tokens matter; punctuation marks are not treated as separate tokens, but are attached to adjacent words. That is, each token is optionally associated with leading and trailing punctuation characters. This approach is more intuitive for non-experts and simplifies the correspondence between tokens and lexicon.

Intermediate between text and lexicon is interlinear glossed text (IGT). An IGT view shows a single sentence, partitioned into glossed words corresponding to individual tokens.

2.2 Lexicon

Central to CLD is a tight integration between texts and lexicon. The character sequence making up a word token is called a **form.** The class of *forms* is very inclusive; it includes not only citation forms but also inflected forms, proper nouns, dialectal variants, misspellings, and so on. Anything that appears in a text is a form, as is any element that a user introduces when analyzing a text, such as bound morphemes or multi-word units.

The lexicon is a table whose keys are forms. This, again, is a much more inclusive conception than usual: inflected forms, misspellings, and so on all have lexical entries. For purposes of printing a lexicon, there will be an ability to designate a subset of entries as canonical entries, but that is not currently implemented.

The lexicon is generated automatically from the texts. It includes an index of all sentences in which a given form occurs, to provide backlinks from lexical entries to texts. That is, from any entry in the lexicon one may obtain a list of example sentences, which is the same as a concordance.

The only way to enter a form into the lexicon is by including the form in a text. In particular, a traditional dictionary is conceived as just another text, one whose "sentences" are single words. If one enters a traditional dictionary into CLD, one may enter the example sentences as a second text. (Nothing prevents one from mixing headwords and example sentences in a single text, but keeping them separate is probably more useful.)

Although one cannot manually enter a new form into the lexicon, one can enter information into the lexical entry for an existing form. There is a single lexical entry for each form, independent of any particular occurrence of the form in text. In the interest of keeping everything intuitive for non-experts, the current lexical fields are very general and simple.

First, to deal with misspellings, dialectal variants, orthographic variants, and the like, one does not correct the original text. Rather, it is possible to indicate that one form is a **variant of** another form, which we may call the *canonical form.* More generally, we define a *canonical form* to be any form that does not have a *variant-of* link. Backlinks are automatically created: if A is the canonical form of B, then B is included in A's list of **variants.**

A canonical form can be viewed as a conventional representative for an equivalence class of forms. Note that CLD chases *variant-of* links to find the ultimate canonical form. That is, if form A is a variant of form B, and form B is a variant of form C, then the canonical form of A is actually C, not B. (If there is a cycle, it is broken arbitrarily, but the software prevents the user from creating cycles under normal circumstances.)

A second relation is introduced for the relationship between an inflected form and its lemma. One may specify that a form **consists of** one or more other forms, which are its *constituents.* There is no requirement that the constituents exist independently in texts; entirely new forms may be introduced in the *consists-of* field. Nor are any assumptions made about how the constituents combine to create the derived form. It is possible that the derived form is simply the concatenation of the constituents, or the constituents may be a stem and infix, or a template and vowel sequence, or entirely abstract. It is permissible for constituents to overlap, and it is permissible for the list of constituents to be incomplete. Whether the order of constituents matters is also up to the user.

Again, backlinks are created automatically. The field **derived-forms** is automatically populated; if B is a constituent of A, then A is a derived form of B.

The only other field currently supported is **gloss.** This provides the word-by-word gloss used in interlinear glossed text.

2.3 Typography and orthography

Text entry and display are areas in which there is a tension between the desiderata of computational linguists and those of language speakers. Desiderata for computational linguists include the following:

- Since computational linguists are likely to work with many languages, including languages with which they have little familiarity, all functionality should be uniform across languages, and should in particular be available without the necessity of installing language-specific keyboards or the like.
- CLD files should be easy to process, without recourse to special libraries (even libraries as broadly available as XML-processing libraries). For this reason, tabular plaintext formats are used.
- CLD files should be processable even using legacy text-processing tools such as grep and awk. For this reason, we understand *plaintext* to mean *ASCII* plaintext.

In contrast, speakers of a language desire to enter and view texts using customary orthography and typography. There are a number of ways in which customary conventions run athwart of the desiderata just listed.

- Most languages use character sets that include non-ASCII characters.
- Many languages use language-specific **input methods** ("keyboards"). Chinese is a particularly complex example.
- Languages differ in their tokenization conventions. Some languages do not mark word breaks at all—Chinese is the obvious example. Vietnamese uses spaces to mark syllable boundaries rather than word boundaries. Most European languages use spaces as word separators but have complicating conventions, such as the use of hyphens or dashes without spaces.
- In most European languages, capitalization is sometimes lexicographically significant (the proper noun *May* is lexicographically distinct from the verb *may*) and sometimes not (all words are capitalized sentence-initially).

In CLD, a language-general representation is used internally, but language-specific customization is made available as an option where practicable. Let us use the term **language kit** loosely to include all functionality that is specific to a given language.

The main language kit element is a **romanization,** which is a mapping between an ASCII encoding (also known as a *practical orthography*) and Unicode. Romanized text consists solely of ASCII characters. (For example, the Arpabet may be viewed as a romanization for a subset of the IPA.) Romanized text is used internally to represent text and forms, and may always be used for text entry. One may think of a romanization as a generic input method, in which the ASCII text represents the keystrokes, and the mapping to Unicode gives the resulting text. In keeping with that analogy, when text is displayed, it is always converted to Unicode, using the romanization.

Input methods are too complex and vary too much from language to language to make it realistic to include them in CLD, but a user may *optionally* use an input method installed in the operating system when entering text. Let us distinguish between **romanized text entry,** in which keystrokes are interpreted as ASCII characters in romanized text, and **native text entry,** in which one uses an input method installed in the operating system. One may optionally enable native text entry on a per-language basis.[1]

To control the forms that appear in the lexicon, romanized text entry must conform to the convention of using spaces uniformly for word separation, even if that conflicts with language conventions. When entering romanized Chinese or Vietnamese text, one must include spaces as word breaks. In European languages, to distinguish hyphens from dashes, one must use spaces with dashes. (A hyphenated word is treated as a single token; one may break it into its constituents in the lexicon.) One must use only lexicographically-significant capitalization: sentence-initial words should not be capitalized.

If one uses native text entry, one may optionally include word-separation, tokenization, and/or decapitalization algorithms in the language kit. Alternatively, orthographic features such as sentence-initial capitalization or space-deletion around dashes may be activated in text display only.[2]

Word senses. One final issue that I include here, though not properly a typographic or orthographic issue, is the disambiguation of word senses. During text entry, one may flag a word as having a non-default sense by suffixing it with a **sense number,** a period being used to separate the sense number from the token proper. CLD does not distinguish between polysemy and homonymy, but each word sense has its own lexical entry. I have found it best to minimize the use of word senses, using them only for the starkest homonym distinctions, but the facility may be used as one sees fit.

One may use either a form with sense number or a form without sense number to access the lexicon. If a sense number is provided, one obtains a single lexical entry, and if not, one obtains a list of entries all sharing the same form. In text, a form without a sense number is treated as having sense 0, the default sense.

2.4 Recordings

Tokenized text mediates between audio and lexicon. We have discussed the connection to the lexicon; let us turn to the connection between text and audio recordings.

CLD assumes that an audio or audio-video file is given; *creating* recordings is outside its purview. For both practical and conceptual reasons, media files are kept in a media directory separate from the CLD texts and lexicons.

The practical reason is that an entire CLD corpus is usually smaller than a single media file, and file management (for example, under git) is simplified if one keeps such disparately-sized files separate. Also, for the sake of portability,

[1]One might expect CLD to automatically detect native text entry by the presence of non-ASCII characters, but it is not possible to automatically detect when a string consisting entirely of ASCII characters is intended as Unicode text rather than romanized text, and even a string intended as romanized text may contain non-ASCII characters like so-called "smart quotes."

[2]The display options are not currently implemented.

all CLD files apart from media use simple tabular ASCII formats, and keeping them separate from binary files again simplifies data management.

Conceptually, CLD files are viewed as **annotations** of audio files, and experience suggests that stand-off annotation, rather than integrated annotation, is more flexible and easier to manage. Note that this does not preclude texts that lack an audio representation, but the most complete case is an audio recording connected to the lexicon via a tokenized text.

CLD provides the ability to transcribe an audio recording in order to create the simple-text annotation. The interface is intended to be as simple and streamlined as possible. Transcription consists in marking the locations of units of interest, which are called **snippets.** Snippets are typically small, small enough that one can transcribe them immediately after hearing them, without replaying them. They may consist of single words, a few short words, or even just a part of a word.

A **transcript** is a list of snippets. A tokenized text called a *transcribed-text column* is automatically generated from a transcript by concatenating the snippets, separated by spaces. In the text, no distinction is made between the spaces between snippets and any that occur within a snippet. The space between two snippets may be suppressed by flagging the second snippet as a word continuation. Sentence breaks are introduced by flagging sentence-initial snippets.

A transcribed-text column differs from an original-text column in only two ways: the transcribed-text column is read-only, and its sentences and tokens are linked to audio.

2.5 Complex texts and stubs

A text in CLD is actually a container for elements that can occur in different combinations. The elements we have already discussed are the media file,[3] the transcript, the transcribed-text column, the original-text column, and the translation. A text is defined to be a recording if it contains a media file. A recording may, but need not, also contain a transcript, which automatically generates a transcribed-text column. A recording may also contain a translation, which is aligned with the transcribed-text column in the same way that a translation is aligned with an original-text column.

A text is defined to be a simple text if it contains an original-text column. Recordings and simple texts are mutually exclusive: an original-text column is not permitted if a media file exists.

There is one additional element, namely, a **table of contents (TOC).** A text that contains a TOC is a **complex text.** The presence of a TOC excludes all other elements. The TOC is a list of component texts. Each component text has a unique name, and may be accessed either by name or by position.

Finally, a newly created text contains no elements, and is called a **stub.** One may convert it to any of the other three text types by adding a media file,

[3]More precisely, a pointer to the media file.

original-text column, or TOC.

2.6 Languages and corpus

For the sake of simplicity, texts in CLD are always monolingual. Occasional foreign words that occur in otherwise monolingual text can be treated as forms like any other and marked as non-native. Parallel texts or other documents that contain passages in multiple languages can be subdivided into multiple monolingual texts. CLD does not currently provide a way of aligning texts across languages. These solutions may be awkward or unworkable for some documents, such as texts that contain a great deal of code-switching; CLD is not the appropriate tool for such documents.

As has already been discussed, central to CLD is the ability to cover a wide variety of languages in a uniform fashion, so as to support cross-linguistic study, and particularly the aims of inductive general grammar. Each language represents a sub-corpus. Each contains a lexicon and a list of texts. Texts are organized in a hierarchy, with complex texts as nonterminal nodes. At the same time, each text has a unique ID (that is, unique within the language), and there is an index that permits direct access to texts by ID, as well as iteration over all texts. There is a separate text index for each language.

Finally, at the highest level, languages are collected into a CLD **corpus.** The corpus is the CLD application file. It may range in size from a single text to a full-blown universal corpus.

2.7 Principles

In the course of the discussion, we have touched on some of the principles guiding the design of CLD. To summarize:

- The primary goals are the production of large quantities of simple, uniform data across multiple languages, and supporting language self-study that complements immersion learning.
- CLD does not aim to be all things to all users. It does not aspire for print-quality page description or coverage of every conceivable type of text. Other software already exists that serves the needs of producing archival-quality documents in which all details of grammatical and discourse structure are captured.
- Simplicity, generality, and intuitiveness are paramount.
- Equally important is robustness in the face of the variation that one encounters when documenting less well-studied languages. Using a *variant-of* field instead of insisting on orthographic regularity provides one example.
- Constraints on the user are minimized. The user is free to choose the orthography, or what constitutes a translation unit, or what sized snippets to use in transcription.

3 Overview of implementation

CLD is implemented in pure Python, and requires no installation beyond unpacking a tar file.

3.1 Desktop and web application

The application can either be run locally as a desktop application or remotely as a web application. The desktop application is implemented by running the web application within an internal web server. One may think of it as a desktop application that uses a web browser as user interface.

3.2 Permissions

There is a permissions system and user login support. Each item (text, language, corpus as a whole) has independent permissions. Default permissions are inherited from the containing item: protecting a language or complex text protects everything that it contains, as well.

Permissions are granted to particular users to perform particular actions. The actions are read, write, and administer. (Administration permission is needed in order to change an item's permissions.) One does not grant permission to perform an action directly, but rather one assigns users to *roles:* owner, editor, or reader. An owner may perform any action, an editor may read or write, and a reader may only read.

Roles may be assigned to groups, in which case every member of the group inherits the granted permissions. There is no fundamental distinction between groups and users: a group is simply a user that has members.

Permissions are only required in a multi-user context, which is to say, within the web application. When CLD is run as a desktop application, the user is set to *root,* which automatically has permission to perform any action.

3.3 Substrate, database, interface

The software is organized into three layers. The lowest layer is the substrate layer, which provides *robust files.* A robust file adds file locking, backup, and undo, which are essential in a multi-user context in which errors may occur during file edits. When one opens a robust file for writing, it is first locked, to assure that only one person writes the file at a time. The output stream that is returned to the caller actually writes to a temporary file. The original file remains untouched until the output stream is closed without error. At that point, the original file becomes a backup and the temporary file is moved into its place. The user may undo a write by replacing the original file with its backup.

The second layer is the database layer, which provides *persistent items.* Persistent items are software objects that are backed by robust files. In a web-application context, every request begins with a blank slate—in particular, one cannot afford an expensive start-up cost to load the corpus. When one accesses the corpus, its children (the languages) are instantiated but not immediately loaded. The contents of a persistent item are loaded lazily, the first time they are required. Persistent items also contain metadata, stored in the same physical file as the item contents. Metadata includes permissions, as well as e.g. text metadata such as title, author, etc.

I distinguish persistent directories from persistent files. Any container—text, language, corpus—is represented by a persistent directory. Persistent directories give the database a hierarchical structure. A persistent directory behaves like a hash table, mapping child names to persistent items representing the children.

The topmost layer is the interface layer. It also has a hierarchical structure, but it is only loosely connected to the hierarchical structure of the database. Its hierarchical structure corresponds to the pathname of a URL, terminating in a web page that represents a *viewer* or *editor* for a database item or a collection of database items or a piece of a database item. The interface layer is organized around the actions that a user may wish to perform, whereas the database layer is organized around the natural structure of the data.

4 Next steps

The next major steps are automatic transcription and morphological inference. There is limited prior work on automatic transcription. There is some work on mapping directly from audio to translation, but that work, like virtually all work in the speech recognition literature, assumes a significant amount of training data.

We are currently exploring an alternative approach, one with vastly reduced requirements for language-specific training data. The plan is to construct a language-independent automatic phonetic transcriber that produces a coarse phonetic transcription. The only language-specific component is then a pronunciation model that maps orthographic strings to phone sequences. The fact that most low-resource languages lack of a long history of literacy becomes an advantage: it implies that the relation between orthography and pronunciation is, in most cases, fairly transparent.

The second major research issue is morphological inference. We would like to determine, in particular, whether having word glosses can be exploited to improve automatic morphological analysis. The standard IBM model of automatic alignment of text and translation has been implemented, though not yet incorporated into the released system.

References

[1] Abney, Steven, and Steven Bird. The human language project: Building a corpus of the world's languages. *Proceedings of the ACL.* 2010.

[2] Bird, Steven. Natural language processing and linguistic fieldwork. *Computational Linguistics.* 2009.

[3] Bird, Steven. A scalable method for preserving oral literature from small languages. *International Conference on Asian Digital Libraries (ICADL).* 2010.

[4] Bird, Steven, Lauren Gawne, Katie Gelbart, and Isaac McAlister. Collecting bilingual audio in remote indigenous communities. *Coling.* 2014.

[5] Bloomfield, Leonard. *Language.* Holt, New York. 1933.

[6] Universal Dependencies. `https://universaldependencies.org/`, accessed 2019 July 19.