Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

How to Compute Colexifications with CL Toolkit (How to do X in Linguistics 10)

Johann-Mattis List
Department of Linguistic and Cultural Evolution
Max Planck Institute for Evolutionary Anthropology

Colleagues often ask us how they could receive more detailed information on specific languages and colexifications in the CLICS database. With the publication of the CL Toolkit package, which allows to merge several CLDF datasets on the fly, carrying out analyses on certain parts of the data underlying the CLICS database is now much easier than before. In order to illustrate this, this tutorial shows how colexifications for a selected number of languages can be computed from two distinct datasets that are included in CLICS (Version 3).

1 Background

Ever since the CLICS database has been published the first time, scholars wanted to carry out specific analyses with the data underlying the database. This has led to some misunderstandings where scholars started to crawl the data from the online interface. Since CLICS is itself an aggregation of currently 30 datasets (Version 3.0 Rzymski et al. 2020), one directly should use the original datasets, which are openly shared in CLDF formats (Forkel et al. 2018), as clearly indicated on the repository presenting the code that we used to create CLCIS[3] (see https://github.com/clics/clics3).

It also happens that scholars ask me to help them obtain certain statistics which cannot be directly retrieved from the online interface of the CLICS database. While accessing the data via the CLDF datasets was a bit tedious in the past, requiring some knowledge of the pycldf package (Forkel and Bank 2021) and the specifics of CLDF, accessing wordlists provided in CLDF is now much easier than with the publication of the CL Toolkit package (List and Forkel 2021). In the following, I will illustrate how CL Toolkit can be used to extract colexifications for particular language families from individual CLDF datasets and write them to a text file that can later be analyzed further.

Computer-Assisted Language Comparison in Practice
Volume 5, Number 6
https://calc.hypotheses.org/4266

Published on 08/06/2022

2 CL Toolkit

CL Toolkit is a Python package that provides convenient access to CLDF wordlists by providing clear ways to access certain those properties of CLDF datasets which we need for the purpose of cross-linguistic studies. Most importantly, CL Toolkit allows you to load several CLDF datasets at once. While the package keeps language varieties distinct, since we argue that the aggregation of lexical data from different sources for the same language should not be done without extensive quality control regarding transcription and orthography, it allows to merge datasets by their common concepts, as reflected by Concepticon identifiers and glosses, which we provide for most lexical datasets that we assembled in CLDF in the past (see List et al. forthcoming for an overview).

In order to install CL Toolkit in the most recent version, we recommend to use pip.

```
$ pip install cltoolkit
```

3 Loading Wordlists with CL Toolkits

Before we can load data with CL Toolkit, we need to import the library. Since CL Toolkit uses pycldf, we also need to import the Dataset class from the pycldf package.

```
from cltoolkit import Wordlist
from pycldf import Dataset
```

Having done this, you can now load datasets of your choice into a CL Toolkit Wordlist. For our illustration here, we will use the CLDF dataset ids (Key and Comrie 2016) and the northeuralex dataset (Dellert et al. 2020). These need to be downloaded before, which can be most conveniently done with the help of git.

```
$ git clone https://github.com/intercontinental-dictionary-series/ids.git
$ git clone https://github.com/lexibank/northeuralex.git
```

Now we can finally load these two datasets into our Wordlist.

```
wl = Wordlist([
   Dataset.from_metadata("ids/cldf/cldf-metadata.json"),
   Dataset.from_metadata("northeuralex/cldf/cldf-metadata.json"),
   ])
```

If you want to have access to the standardized transcriptions, which are available for some CLDF wordlists which we have prepared in the past, you need to load the pyclts

package (List et al. 2020, see Anderson et al. 2018 for details on the CLTS reference catalog on Cross-Linguistic Transcription Systems and our standardization procedure for phonetic transcriptions). In order to use this, you need to download the CLTS data first.

```
$ git clone https://github.com/cldf-clts/clts.git
```

To load the CLTS data with Python, you then need to import the CLTS class and initialize it by pointing to the folder in which the data can be found.

```
from pyclts import CLTS
clts = CLTS("clts")
```

When loading a wordlist now, you need to add the bipa attribute of the initialized CLTS class as a keyword.

```
wl = Wordlist([
   Dataset.from_metadata("northeuralex/cldf/cldf-metadata.json"),
   ts=clts.bipa
   ])
```

4 Computing Colexifications

Computing colexifications can be done in a very efficient manner, using hash-tables, better known as “dictionaries” in Python. The major ideas is that — in order to find out if the words for a concept A and a concepts B are the same — we do not need to iterate over all word pairs at the same time. What we can do instead is creating a dictionary with a word form as a key, and adding concepts as values to a list in this dictionary. Only later do we need to iterate over the keys in the dictionary and then check for all those cases in which the value contains more than one concept, and where these concepts differ.

With CL Toolkit, we have the additional advantage that we have direct access to the forms in very language which we have loaded into our Wordlist object. With the help of standard library functions like the defaultdict, which allows us to fill a dictionary even if a key has not yet been added, and the combinations function, which allows for a quick iteration over possible pairs in a list, we can therefore easily create a short and fast function that retrieves all colexifications in our aggregated datasets.

Note that the values for the colexifications are stored in a dictionary (called data here), in which concept pairs are the keys and the values are a dictionary itself, consisting of a language identifier for the respective language showing the colexification as the key, and the concrete word form that reflects the colexification as the value.

The function to compute colexifications for a CL Toolkit Wordlist object thus can be written as follows below.

```
from collections import defaultdict
from itertools import combinations

def get_colexifications(language, data):
    """Compute colexifications and add them to the data dictionary."""
    tmp = defaultdict(set)
    for form in language.forms:
        if form.concept:
            tmp[form.form].add(form.concept.concepticon_gloss)

    for forms, colset in tmp.items():
        if len(colset) > 1:
            for cA, cB in combinations(sorted(colset), r=2):
                data[cA, cB][language.name] += [form]
        else:
            c = colset.pop()
            data[c, c][language.name] += [form]
```

Note that we need to sort the set of concepts before adding them to our data dictionary, in order to avoid that we have to fill the dictionary with the links in both directions.

5 Inferring Colexifications from the Data

In order to apply the code now, not much else has to be done. Since we are only interested in particular languages from particular families, we need to filter the list of languages. This can be easily done and illustrates the advantages of Cl Toolkit for the manipulation of CLDF datasets.

```
cols = defaultdict(lambda : defaultdict(list))
all_languages = []
for language in wl.languages:
    if language.family in ["Nakh-Daghestanian", "Turkic"] or \
            language.glottocode in ["jude1256", "russ1263"]:
        print("[i] analyzing language {0}".format(language.name))
        all_languages += [language.name]
        get_colexifications(language, cols)
```

Having assembled the colexifications for the set of languages in our sample, we can now extract only those concept pairs which are colexified with at least one time in our data.

```
colexified = [(cA, cB) for cA, cB in cols if cA != cB]
```

With the concepts involved in colexification having been extracted, we can now finally write the colexification data to file. Since we also traced for each concept, if it has a valid form in the various languages, we can include this information now, and distinguish between clear evidence for a colexification and missing data (marked by a question mark).

```
matrix = []
for cA, cB in combinations(colexified, r=2):
    row = [cA, cB]
    for language in all_languages:
        if cols[cA, cA][language] and cols[cB, cB][language]:
            row += ["1" if language in cols[cA, cB] else "0"]
        else:
            row += ["?"]
    matrix += [row]
with open("colexifications.tsv", "w") as f:
    f.write("ConceptA\tConceptB\t"+"\t".join(all_languages)+"\n")
    for row in matrix:
        f.write("\t".join(row)+"\n")
```

That is all, with this small code snippet, you can extract the colexifications in CLDF datasets without having to access the CLICS database itself.

The code and data that you need to carry out the analyses described here are also available in the form of a GitHub Gist, accessible at https://gist.github.com/LinguList/af661e8f5254a8bf939ef2c5f8fc6e81.

References

Anderson, Cormac and Tresoldi, Tiago and Chacon, Thiago Costa and Fehn, Anne-Maria and Walworth, Mary and Forkel, Robert and List, Johann-Mattis (2018): A Cross-Linguistic Database of Phonetic Transcription Systems. Yearbook of the Poznań Linguistic Meeting 4.1. 21-53.

List, Johann-Mattis and Forkel, Robert (2021): CL Toolkit. A Python Library for the Processing of Cross-Linguistic Data [Software Library, Version 0.1.1]. Max Planck Institute for Evolutionary Anthropology: Leipzig. https://pypi.org/project/cltoolkit

Johannes Dellert and Thora Daneyko and Alla Munch and Alina Ladygina and Armin Buch and Natalie Clarius and Ilja Grigorjew and Mohamed Balabel 1 · Hizniye Isabella Boga and Zalina Baysarova and Roland Muhlenbernd and Johannes Wahle and Gerhard Jager (2020): NorthEuraLex: a wide-coverage lexical database of Northern Eurasia. Language Resources & Evaluation 54. 273–301.

Forkel, Robert and List, Johann-Mattis and Greenhill, Simon J. and Rzymski, Christoph and Bank, Sebastian and Cysouw, Michael and Hammarstrom, Harald and Haspelmath, Martin and Kaiping, Gereon A. and Gray, Russell D. (2018): Cross-Linguistic Data Formats, advancing data sharing and re-use in comparative linguistics. Scientific Data 5.180205. 1-10.

Forkel, Robert and Bank, Sebastian (2021): PyCLDF. Max Planck Institute for the Science of Human History: Jena. https://pypi.org/project/pycldf

Key, Mary Ritchie and Comrie, Bernard (2016): The Intercontinental Dictionary Series. Leipzig:Max Planck Institute for Evolutionary Anthropology.

List, Johann-Mattis and Forkel, Robert and Greenhill, Simon J. and Rzymski, Christoph and Englisch, Johannes and Gray, Russell D. (forthcoming): Lexibank, A public repository of standardized wordlists with computed phonological and lexical features. Scientific Data. 1-31.

Johann-Mattis List and Cormac Anderson and Tiago Tresoldi and Robert Forkel (2020): PyCLTS. A Python library for the handling of phonetic transcription systems [Software Library, Version 3.0.0]. Max Planck Institute for the Science of Human History: Jena. https://pypi.org/project/pyclts

Rzymski, Christoph and Tiago Tresoldi and Simon Greenhill and Mei-Shin Wu and Nathanael E. Schweikhard and Maria Koptjevskaja-Tamm and Volker Gast and Timotheus A. Bodt and Abbie Hantgan and Gereon A. Kaiping and Sophie Chang and Yunfan Lai and Natalia Morozova and Heini Arjava and Nataliia Hubler and Ezequiel Koile and Steve Pepper and Mariann Proos and Briana Van Epps and Ingrid Blanco and Carolin Hundt and Sergei Monakhov and Kristina Pianykh and Sallona Ramesh and Russell D. Gray and Robert Forkel and List, Johann-Mattis (2020): The Database of Cross-Linguistic Colexifications, reproducible analysis of cross- linguistic polysemies. Scientific Data 7.13. 1-12.