Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

PyEDICTOR: A Small Python Package that Integrates LingPy, EDICTOR, and CLDF

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

With the introduction of CLDF as a major format for data exchange, there is an increased need in handy solutions for the conversion of CLDF to the formats required by computer-assisted tools like LingPy and EDICTOR, which allow to preprocess data automatically or to curate data by adding detailed annotations. With the publication PyEDICTOR, there is now a very lightweight software package that is supposed to provide first solutions for a successful integration of CLDF with these existing tools for computer-assisted language comparison.

1 Introduction

The introduction of CLDF as our standard format for the handling of lexical wordlist data has greatly changed the field of computer-assisted language comparison. Thanks to the Cross-Linguistic Data Formats initiative (Forkel et al. 2018), it was possible to drastically improve the replicability and transparency of our workflows for the aggregation of large lexical data collections from existing datasets. These workflows are the core of databases like the Database of Cross-Linguistic Colexifications (CLICS, https://clics.clld.org, Rzymski et al. 2020) or the recently published Lexibank repository (List et al. 2022), and have proven their usefulness in a larger number of publications in which both datasets have been used so far (see Jackson et al. 2019 for CLICS and Winter et al. 2022 for Lexibank).

Since CLDF differs from the single-file formats used in the LingPy software library for quantitative language comparison (https://lingpy.org, List and Forkel 2022) and the EDICTOR tool for the curation of etymological data in historical linguistics (https://digling.org/edictor, List 2021), additional steps are now needed to convert a CLDF dataset to the formats used by LingPy and EDICTOR as well as to produce

Computer-Assisted Language Comparison in Practice
Volume 5, Number 9
https://calc.hypotheses.org/4432

Published on 26/09/2022

transparent CLDF datasets from data curated in EDICTOR or preprocessed with the help of LingPy. While the latter can be easily done with the help of the Lexibank workflow (List et al. 2022), which builds on the CLDFBench ackage (https://pypi.org/project/cldbench, Forkel and List 2020, the conversion of CLDF datasets to the formats required by LingPy and EDICTOR has been a bit less straightforward so far.

Thus, while LingPy can read CLDF datasets in a rather flexible way and allows to output them in the typical TSV-files used in LingPy applications, the relevant functions are not very well documented in the library and require a certain amount of repetitive typing, specifically in those cases, where one wants to customize the export. An additional problem is the conversion of CLDF data to the specific SQLITE database structures which are needed to run EDICTOR on the webserver, which has the advantage that one can log in with one's personal account details and modifications one makes to the data are automatically stored when using the EDICTOR application.

With PyEDICTOR (https://pypi.org/projects/pyedictor, List 2021) there is now a first attempt to provide a very lightweight Python software package, currently available in version 0.3.0, which was written for the purpose of allowing for a convenient integration of data available in the form of an EDICTOR application. PyEDICTOR allows to convert a CLDF dataset via the commandline to a TSV-file that can be directly used by LingPy or loaded into the EDICTOR application. Additionally, users can even convert the data to the SQLITE-triple-structure, which is needed to access a dataset through EDICTORS with a server. In addition, one can quickly download EDICTOR datasets which are stored on the server, and directly analyze them in LingPy.

2 Installation

The installtion of PyEDICTOR can be done in a straightforward manner with the help of the package manager PIP:

```
$ pip install pyedictor
```

3 Converting CLDF Data to EDICTOR Format

For the conversion of CLDF data to EDICTOR (or LingPy) formats, one can most conveniently use the commandline interface of PyEDICTOR, which can be invokved with the edictor command.

Thus, assuming you want to investigate a CLDF dataset, like the `utoaztecan` dataset (Greenhill et al. forthcoming), with the help fo the EDICTOR application, all you have to do to get the data into the right format is to clone the repository and convert it later.

```
$ git clone https://github.com/lexibank/utoaztecan.git
$ cd utoaztecan
$ edictor wordlist --name=utoaztecan
```

This will create a TSV file utoaztecan which has the most relevant columns available from the CLDF dataset, as you can check with LingPy:

```
from lingpy import *
from tabulate import tabulate
wl = Wordlist("utoaztecan.tsv")
table = []
table += [["Languages", wl.width]]
table += [["Concepts", wl.height]]
table += [["Words", len(wl)]]
print(tabulate(table))
```

This will output:

```
--------- ----
Languages   46
Concepts   121
Words     5813
--------- ----
```

This dataset will only have the basic number of columns (doculect, concetp, value, form, tokens). To import more specific column values, you can extend the command:

```
$ edictor wordlist --name=utoaztecan --addon="cogid_cognateset_id:cognacy"
```

This will create the column cognacy in the target file utoaztecan.tsv, which we address by its LingPy-internal name cogid_cognateset_id (`cogid` refers to the CLDF cognate table). In this way, by adding specific “addons”, you can flexibly expand what parts of the CLDF data are shown in your resulting TSV file (adding existing Glottocodes, for example, could be done with adding `"language_glottocode:glottocode"` as an addon). You can check if this worked by extending the Python snipped shown above.

```
from lingpy import *
from tabulate import tabulate
wl = Wordlist("utoaztecan.tsv")
table = []
table += [["Languages", wl.width]]
table += [["Concepts", wl.height]]
table += [["Words", len(wl)]]
table += [["Cognates", len(wl.get_etymdict(ref="cognacy")]]
print(tabulate(table))
```

This will then yield the following output.

```
--------- ----
Languages   46
Concepts   121
Words     5813
Cognates  1371
--------- ----
```

By adding the flag `--sqlite`, the output will be in SQLITE, following the format that you will need for an EDICTOR server-based web application. Additionally, you can even point to a script that you can use to preprocess and modify the data for you. Thus, if you wanted to have automated cognate sets added to the Uto-Aztecan dataset, you could write a little script `preprocessing.py` that computes cognates and then point to it via the keyword `--preprocessing` from the commandline.

As script, you need to define a function run which takes as an argument the wordlist and returns a wordlist or a wordlist-like object (e.g., a dictionary).

```
from lingpy import *

def run(wordlist):
    lex = LexStat(wordlist)
    lex.cluster(
        method="sca",
        threshold=0.45,
        ref="autocogid")
    D = {0: wordlist.columns+["autocogid"]}
    for idx in lex:
        D[idx] = [lex[idx, h] for h in D[0]]
    return D
```

This script will now add a new cognate identifier based on one of the automated cognate identification methods from LingPy.

```
$ edictor wordlist --name=utoaztecan --preprocessing=preprocessing.py
```

This can be easily seen when checking the first lines of the resulting TSV file.

```
$ head utoaztecan.tsv

ID  DOCULECT    CONCEPT VALUE   FORM    TOKENS  NOTE
AUTOCOGID
119 Mono    I   nɯ  nɯ  n ɯ     2
275 Tubatulabel I   nɯ̄ nɯ̄ n ɯ̄/ɯː     2
412 Serrano I   nɯ̄?    nɯ̄?    n ɯ̄/ɯ ʔ        2
536 Cahuilla    I   ne  ne  n e     2
663 Cupeno  I   nəʔ nəʔ n ə ʔ       2
790 Luiseno I   nō  nō  n ō/oː      2
921 Gabrielino  I   nō  nō  n ō/oː      2
922 Gabrielino  I   nō̰mɑʔ  nō̰mɑʔ  n ō̰/oː m ɑ ʔ       38
1049    Hopi    I   nɯʔ nɯʔ n ɯ ʔ       2
```

4 Loading Data from EDICTOR Applications

All the EDICTOR applications where scholars actively annotate data from the server-based version of the EDICTOR application can be freely accessed, but not be modified without permission. We do not usually share the URLs where they can be found, but also do not think that it is a big secret or that the data should be specifically saved from people watching it, as we assume that it would not be possible to use the data for anybody who might want to take advantage of other peoples unpublished cognate set annotations.

In order to access specific parts of the data which is stored in an EDICTOR application, one can use URL parameters. With these parameters, one can specify, for example, which languages one wants to work with, or which concepts one wants to deal with in a specific run.

Thus, the following URL would allow you to have a small selection of the data that was used in an earlier study by Sagart et al. (2019) on the phylogeny of Sino-Tibetan languages.

```
https://digling.org/edictor/?remote_dbase=sinotibetan&amp
;file=sinotibetan&amp;preview=200&amp;async=true&amp;samp
a=TOKENS&amp;basics=DOCULECT|CONCEPT|IPA|TOKENS|NOTE|COGI
D|BORROWING&amp;doculects=Atsi|Bai_Jianchuan&amp;concepts
=above|all&amp;columns=DOCULECT|CONCEPT|IPA|TOKENS
```

While you can open this URL and export the data to the TSV-format from within the EDICTOR application without problems, you can also load the same data directly with the help of the fetch-command provided by the PyEDICTOR packages.

```
from pyedictor import fetch
from lingpy import Wordlist

wl = fetch(
   "sinotibetan",
   concepts=["above", "all"],
   languages=["Atsi", "Bai_Jianchuan"],
   columns=["DOCULECT", "CONCEPT", "IPA", "TOKENS", "COGID"],
   base_url="https://digling.org/edictor/",
   to_lingpy=True
)

print("Wordlist has {0} languages and {1} concepts.".format(
   wl.height, wl.width))
```

This will print to the terminal, that the wordlist has 2 languages and 2 concepts.

5 Outlook

I do not consider the PyEDICTOR package as some kind of final solution for the integration of CLDF with our tools for computer-assisted language comparison, but — for the time being — rather as a test package that makes it much easier to handle frequently recurring tasks. Future discussions and a thorough evaluation of the functionality provided now may well yield an alternative package in the future or provide solutions for certain functionalities from within established software packages as LingPy directly. For now, however, my daily coding already profits a lot from the package, as it allows me to prepare CLDF datasets quickly for analyses in LingPy or inspection in EDICTOR without requiring me to use the Python console or customized Python scripts.

References

List, Johann-Mattis (2021): EDICTOR. A web-based tool for creating, editing, and publishing etymological datasets. Version 2.0.0. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://digling.org/edictor/

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. DOI: 10.1038/sdata.2018.205

Forkel, Robert and List, Johann-Mattis (2020): CLDFBench. Give your Cross-Linguistic data a lift. In: Proceedings of the Twelfth International Conference on Language Resources and Evaluation. 6997-7004.

Greenhill, Simon J. and Haynie, Hannah J. and Ross, Robert M. and Chira, Angela and List, Johann-Mattis and Campbell, Lyle and Botero, Carlos A. and Gray, Russell D. (forthcoming): A recent Northern origin for the Uto-Aztecan family. *Language* 0.0.

Joshua Conrad Jackson and Joseph Watts and Teague R. Henry and List, Johann-Mattis and Peter J. Mucha and Robert Forkel and Simon J. Greenhill and Russell D. Gray and Kristen Lindquist (2019): Emotion semantics show both cultural variation and universal structure. *Science* 366.6472. 1517-1522. DOI: 10.1126/science.aaw8160

List, Johann-Mattis and Forkel, Robert (2022): LingPy. A Python library for quantitative tasks in historical linguistics [Software Library, Version 2.6.9]. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/lingpy

List, Johann-Mattis and Forkel, Robert and Greenhill, Simon J. and Rzymski, Christoph and Englisch, Johannes and Gray, Russell D. (2022): Lexibank, A public repository of standardized wordlists with computed phonological and lexical features. *Scientific Data* 9.316. 1-31. DOI: 10.1038/s41597-022-01432-0

List, Johann-Mattis (2021): PyEDICTOR [Python library, Version 0.3.0]. https://pypi.org/project/pyedictor

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. DOI: 10.1038/s41597-019-0341-x

Sagart, Laurent and Jacques, Guillaume and Lai, Yunfan and Ryder, Robin and Thouzeau, Valentin and Greenhill, Simon J. and List, Johann-Mattis (2019): Dated language phylogenies shed light on the ancestry of Sino-Tibetan. *Proceedings of the National Academy of Science of the United States of America* 116. 10317-10322.DO: 10.1073/pnas.1817972116

Bodo Winter and M\'arton S\'oskuthy and Marcus Perlman and Mark Dingemanse (2022): Trilled /r/ is associated with roughness, linking sound and touch across spoken languages. *Scientific Reports* 12.1.