Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

Creating Custom Commands in CLDF: From Lexibank to Nexus Files

Frederic Blum
Department of Linguistic and Cultural Evolution
Max Planck Institute for Evolutionary Anthropology

Thanks to the CLDFBench Python package, CLDF datasets compiled with CLDFbench have a rich command-line utility that can easily be expanded by custom commands (Forkel et al. 2018, Forkel and List 2020) . Taking as an example the creation of a Nexus-file for phylogenetic analysis from an existing Lexibank Wordlist (List et al. 2022) , this tutorial will guide you through the necessary steps for writing a script that can be used as a CLDFBench command. You will then be able to apply the same workflow to other scripts you may want to use for a certain repository.

1 Prerequisites

A prerequisite for this tutorial is the existence of an up and running Lexibank dataset. You can find previous tutorials on the creation of such datasets (Blum 2021, List 2021). Here, we will use the crossandean-dataset with data from the Quechua language family (Blum et al. submitted).

2 Preparations

The first step is to create the script that you want to apply to the CLDF dataset. For our purpose, we can create an empty script with the name `make_nexus.py` and put it into a dedicated folder. For this, you can create a new folder in the main repository with the name of your dataset, followed by `commands`. In my case, the newly created folder was called `crossandeancommands`.

By having all custom commands in the same folder, you can add an entry point for the CLDF commands and direct it to that folder. This is done by adding the entry point to the `setup.py` file. In addition to the existing `lexibank.dataset` you can add

Computer-Assisted Language Comparison in Practice
Volume 5, Number 11
https://calc.hypotheses.org/4403/

Published on 28/11/2022

your own `cldfbench.commands`, and direct the entry point to the folder you created. The updated `setup.py`-file will have the following entry points:

```
entry_points={
        'lexibank.dataset': ['blumquechua=lexibank_crossandean:Dataset',],
        'cldfbench.commands': ['crossandean=crossandeancommands',],
},
```

The first term before the equal sign, in this case, `crossandean`, is the name by which you call the commands in the command line. The command to call the script will then later be the following: `cldfbench crossandean.make_nexus`. But of course, right now there is no content to your script, so this would have no effect.

3 Loading the data

The code in this script will be written in Python. We will do this in a new file `script.py`, where you will add all the code you want to be included in the command. Within the script, the first step is to import the dataset, so that you can access the CLDF data. You can easily import the dataset as a module from within Python, as shown in the following code block:

```
from lingpy import Wordlist
from lexibank_crossandean import Dataset as CA

def run(args):
dataset = CA()
wordlist = Wordlist.from_cldf(
                str(dataset.cldf_dir.joinpath("cldf-metadata.json").as_posix()),
    # columns to be loaded from CLDF set
    columns=("language_id","concept_name","segments","cogid" ),
    # Renaming some columns
    namespace=(
        ("language_id", "doculect"),
        ("concept_name", "concept"),
    )
)
```

In this call, you can also specify the relevant columns that you want to load. By using the `language_` or `concept_` prefix, you can access the information that is stored in the languages- or parameter-table. Afterwards you can rename the columns with the `namespace`-keyword by using a list of tuples `[("old_name", "last_name")]`.

There are also multiple ways to filter the dataset for different languages. In many cases you want to exclude some doculects that belong to certain subgroups, or doculects with low coverage of concepts. In other cases, you may want to exclude some concepts that are part of the dataset. This can all easily be done in the same script.

I first created a set of entries that I want to exclude. In my case, I chose the names of certain colonial varieties of Quechua. In the next step, I created an empty dictionary containing all headers of my original wordlist that will be expanded with the filtered data. In the final step, I iterate through the dataset and add a condition on what entries should be added to the new wordlist. In my case, this condition is that the doculect of the entry is not part of the blacklist of languages. However, the same logic would apply for concepts or any other column.

```
# These are the blacklists for the Nexus file
blacklist_languages = {
        'CuzquenoAntiguo',
        'Anonimo',
        'SantoTomas'
}

D = {0: [c for c in wordlist.columns]}  # defines the header

for idx in wordlist:
if wordlist[idx, "doculect"] not in blacklist_languages:
    D[idx] = [wordlist[idx, c] for c in D[0]]

wlnew = Wordlist(D)
```

4 Creating the Nexus file

You now have a wordlist that consists of all the items that you want to be part of your Nexus file. Now you import the `write_nexus` function from lingpy, and run the function on your wordlist by specifying the column with your annotation of cognate sets (e.g. `COGID`) and the Nexus-type, depending on the program you will use for analysis. In this function, you also specify an output-folder and the name of the file.

```
from lingpy.convert.strings import write_nexus

write_nexus(
        wlnew,
        ref="cogid",
        mode="BEAST",
        filename=str(datast.dir.joinpath('outputs', 'quechua_modern.nex'))
)
args.log.info("wrote data to file 'outputs/crossandean-beast.nex'")
```

If you now run `cldfbench crossandean.make_nexus` in the command line, you will create the Nexus file that follows your specification.

5 Conclusion

This tutorial led you through the creation of custom commands in CLDF datasets. We converted a lexibank dataset to a Nexus file that can be used for phylolinguistic analysis. However, you can put many more scripts in your commands-folder and run them through the command line individually. The creation of a Nexus-file is just one of many possibilities.

References

Blum, Frederic. 2021. Data Gathering in Times of a Pandemic: Upcycling Constenla Umaña's Data on the Chibchan, Lencan and Misumalpam Language Families, in Computer-Assisted Language Comparison in Practice 4 (5), 1-6. https://calc.hypotheses.org/2751.

Blum, Frederic and Barrientos Ugarte, Carlos and Poirier, Zoe and Ingunza, Adriano. Submitted. A phylolinguistic classification of the Quechua language family. https://doi.org/10.5281/zenodo.5497510

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, 1-10. https://doi.org/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 Language Resources and Evaluation Conference, pages 6995–7002, Marseille, France. European Language Resources Association. https://aclanthology.org/2020.lrec-1.864

List, Johann-Mattis. 2021. Converting the Vietic Dataset by Sidwell and Alwes from 2021 to CLDF, in Computer-Assisted Language Comparison in Practice 4 (9), 1-15. https://calc.hypotheses.org/2954.

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, 1-16. https://doi.org/10.1038/s41597-022-01432-0