Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

Querying Datasets with Cognates in the Lexibank Repository

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

Recently, I was asked by a colleague how one could query only those datasets in the Lexibank repository which come with cognate sets annotated by humans. While I first thought this could be done in a very straightforward way, I figeured out, when trying it myself, that the code still needs some workarounds. As a result, I thought it is best to share the solution I came up with in a blog post in order to make it accessible to colleagues who might be interested in inspecting and using the data provided by the Lexibank repository in more detail.

Starting Point

My starting point is the lexibank_analysed repository, that was the core of our study presenting the first version of Lexibank earlier this year (List et al. 2022). This repository (available on GitHub at https://github.com/lexibank/lexibank_analysed), offers Python code that — once installed — can be used to run all analyses we described in our original study via the commandline. Additionally, the Python code can be accessed in the form of a Python library which also offers access to all datasets.

The core of the repository is a the file lexibank.csv which you can find in the folder etc. This file contains the information about all individual Lexibank datasets which we included in our first release. These datasets, which have been published independently and archived with Zenodo, make up the core of the Lexibank repository as a repository that aggregates many standardized datasets and shows how they can be analyzed. The CSV file contains information on the detailed versions of each datasets and also on the "classification" of the dataset with respect to the detailed data each individual dataset offers.

As we describe in our paper, we tag datasets that come along with cognate sets as belonging to the CogCore subset of Lexibank. These datasets are marked with an x in the field CogCore in the lexibank.csv file. After installying the lexibank_analysed

Computer-Assisted Language Comparison in Practice
Volume 5, Number 10
https://calc.hypotheses.org/4872

Published on 31/10/2022

package (which can be done with pip, following the installation instructions provided by the package), one can directly access the CSV file and iterate over it, by first loading the Python library and then iterating over the CSV file and filtering out those datasets which are tagged as CogCore.

```
from cldfbench_lexibank_analysed import Dataset as Lexibank
lexibank = Lexibank()
for row in lexibank.etc_dir.read_csv("lexibank.csv", dicts=True):
    if row["CogCore"] == "x":
        print(row["Dataset"])
```

This small piece of code will list all datasets that make up the CogCore part of the Lexibank repository.

```
bdpa
blustaustronesian
bodtkhobwa
bowernpny
cals
...
```

Loading Datasets

In order to directly access these datasets, I recommend to load them with the help of the LingPy software package (List and Forkel 2022). LingPy has a command that allows to read a CLDF into a LingPy Wordlist, which is the base class to handle multilingual datasets in LingPy. In order to obtain access to the original datasets underlying the Lexibank repository, you need to run the download command first:

```
$ cldfbench download cldfbench_lexibank_analysed.py
```

This will download all individual datasets in their versions used in our study and place them into the folder raw of the repository.

Now that we have obtained all individual datasets, we can try and query them with the help of LingPy while looping over them. In order to make sure that we have loaded the datasets correctly, we can write some basic characteristics to a table, like the number of concepts, number of language varieties, and number of words, which are defined as basic characteristics of a LingPy Wordlist (height, width, and length).

```
from lingpy import Wordlist
from cldfbench_lexibank_analysed import Dataset as Lexibank
from tabulate import tabulate

lexibank = Lexibank()
table = []
for row in lexibank.etc_dir.read_csv("lexibank.csv", dicts=True):
    if row["CogCore"] == "x":
        # check for cognates table
        pth = lexibank.raw_dir.joinpath(row["Dataset"], "cldf")
        wl = Wordlist.from_cldf(
                pth.joinpath("cldf-metadata.json"),
                )
        table += [[row["Dataset"], wl.height, wl.width, len(wl)]]
print(tabulate(table, headers=["Dataset", "Concepts", "Languages", "Words"],
tablefmt="pipe"))
```

This will output a table with the information we were asking for.

Dataset	Concepts	Languages	Words
bdpa	519	538	50095
blustaustronesian	210	20	4358
bodtkhobwa	662	8	4720
bowernpny	344	190	44876
cals	184	88	15826
carvalhopurus	205	4	731
chaconarawakan	102	8	711
chaconbaniwa	243	14	2354
chaconcolumbian	128	69	9030
chacontukanoan	142	16	1542
…	…	…	…

We can extend this table now, by counting the cognate sets and the singletons. But before we do this, we need to define some exceptions, since five datasets which are assigned to the CogCore subset of Lexibank contain partial cognates, which need to be handled differently and should rather be excluded in our experiment (see Wu and List forthcoming for a detailed discussion of partial cognates). We do so by creating a list that stores all names of datasets with partial cognates.

```
partial = [
        "mannburmish",
        "bodtkhobwa",
        "houchinese",
        "liusinitic",
        "tuled",
        ]
```

What we also need to adjust is the individual columns we want to extract from the CLDF data and the namespace, that is, how we want to call these columns when loading them into a LingPy Wordlist class. Explaining all details of the different naming practices would go beyond the scope of this post, so I will just share the code I used here, to define both aspects. All that is important in this context is that the column that will store the cognates will be called cog in the LingPy Wordlist that we create.

```
columns = [
        'concept_name', 'language_id', 'value', 'form', 'segments',
        'cognacy', "cogid_cognateset_id"]

namespace = (
        ('concept_name', 'concept'),
        ('language_id', 'doculect'),
        ("form", "form"),
        ("value", "value"),
        ('segments', 'tokens'),
        ('cognacy', 'cognacy'),
        ('cogid_cognateset_id', 'cog')
        )
```

When querying the number of cognate sets, we first need to load the data along with the cognate table and then create what we call an etym_dict in LingPy. This etym_dict organizes the data as a Python dictionary, with the key being the identifier of a cognate set and the value corresponding to a list in which all individual word identifiers are listed for each of the reflexes of the individual cognate set. While we can retrieve the data structure with the command Wordlist.get_etymdict(ref="cog"), we need to add one workaround that checks for those cases in which the cognate set table of the CLDF dataset (in which the original data is stored, see Forkel et al. 2018) lists no values, indicating that no cognate judgment was made. We need to recode these as cognate sets for individual words, thus, the words assigned to these cognate sets are not cognate with any other words in the data.

With this workaround in place, our command to produce the table looks now as follows.

```
table = []
for row in lexibank.etc_dir.read_csv("lexibank.csv", dicts=True):
    if row["CogCore"] == "x" and row["Dataset"] not in partial:
        pth = lexibank.raw_dir.joinpath(row["Dataset"], "cldf")
        wl = Wordlist.from_cldf(
                pth.joinpath("cldf-metadata.json"),
                columns=columns,
                namespace=namespace
                )
        # workaround for some datasets
        new_id = 1
        for idx, cog in wl.iter_rows("cog"):
            if not cog:
                wl[idx, "cog"] = new_id
                new_id += 1

        # compute etymdict
        etd = wl.get_etymdict(ref="cog")

        # compute singletons
        cognates = len([cog for (cog, idxs) in etd.items() if len(
            [idx for idx in idxs if idx]) > 1])

        table += [
            [row["Dataset"], wl.height, wl.width, len(wl), cognates,
                len(etd)-cognates]
            ]
print(tabulate(
    table,
    headers=["Dataset", "Concepts", "Languages", "Words", "Cognates",
        "Singletons"],
    tablefmt="pipe"
    ))
```

The output in tabular form looks as follows.

Dataset	Concepts	Languages	Words	Cognates	Singletons
bdpa	519	538	50095	750	0
blustaustronesian	210	20	4358	321	2409
bowernpny	344	190	44876	4494	25054
cals	184	88	15826	718	181
carvalhopurus	205	4	731	206	0
chaconarawakan	102	8	711	135	141
chaconbaniwa	243	14	2354	298	312
chaconcolumbian	128	69	9030	1192	1316
…	…	…	…	…	…

Conclusion

This short tutorial shows how we can access those datasets in the Lexibank repository which come with expert cognate judgments. As one can see from the code itself, it still requires some work to get access to all the datasets, and we need some workarounds, since the ways in which cognate sets are coded still differ across the datasets. In the future, I hope that we can further standardize the data in order to make it easier to access those datasets in Lexibank that come with cognate codings. Nevertheless, given that we needed only 50 lines of code in order to access as many as 44 datasets with expert cognat codings in a homogeneous way, I think one can already see why it is useful to standardize data, and why Lexibank might be a useful resource for ongoing work in computational historical linguistics.

The code underlying this study has also been shared in the form of a public GitHub Gist (https://gist.github.com/LinguList/22bb5c7c31aafdfe83dcf32faac4c470).

References

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.

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.

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.

Wu, Mei-Shin and List, Johann-Mattis (forthcoming): Annotating cognates in phylogenetic studies of South-East Asian languages. Language Dynamics and Change 0.0. 1-25.