Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

How to Run EDICTOR 3 Locally

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

EDICTOR3 offers many ways of comparing language data with computer-assisted methods. This study offers a short overview of how to run EDICTOR3 locally, without the need for uploading the data to a server or being connected to the internet, while maintaining all the functionalities. In a first step, we will show how one can download a Lexibank dataset and create different types of files that one can use with EDICTOR. We will then proceed to present the possibility of running an EDICTOR server locally and to edit the dataset that one has downloaded.

1 Introduction

Computer-assisted methods for language comparison have come a long way in recent years (Wu et al. 2020). Methods for automatizing steps of the comparative method have been developed, and many tools and workflows have already been presented in this series. Here, we will focus on the EDICTOR3 application (List et al. 2025, https://edictor.org). In contrast to previous versions which were run online, the accompanying Python 3 package (https://pypi.org/project/edictor) offers the possibility to run your own EDICTOR server locally. The article will lead you through the necessary steps to work with this application, and showcase different ways of preparing your data.

We will use the `blumpanotacana` dataset (Blum et al. 2024) for this article. The dataset features data from 21 language varieties of several language families, the largest being the Pano-Takana family. All languages of the dataset are spoken in the Amazon region in South America. The repository follows the Cross-Linguistic Data Formats (CLDF, https://cldf.clld.org, Forkel et al. 2018), specifically the Lexibank structure (https://lexibank.clld.org, List et al. 2022). It has a couple of files for the maintenance of the repository, as well as folders for metadata (`etc/`), the raw data (`raw/`), and the CLDF data (`cldf/`). All data is stored in tabular formats and linked through certain identifiers. Using CLDF ensures that the data is easily accessible and re-usable with different tools, exemplified through its use in this article. A detailed description how to

Computer-Assisted Language Comparison in Practice
Volume 8, Number 1
URL: https://calc.hypotheses.org/8143 DOI: 10.15475/calcip.2025.1.1

Published on 27/01/2025

create such a dataset for lexical comparisons and how to provide the necessary standardization is given in Blum et al. (2024).

2 First steps

2.1 Preliminary Information

Windows users might have to run some additional steps. Please follow the instructions provided by Snee (2024), which should be sufficient to follow all steps described in the following. Note also that — as an application written for Python 3 — depending on your configuration, you may have to use the command `python3` instead of the command python in order to follow all steps discussed in the following. Since Python 3 has been the new standard Python version for a very long time now, with Python 2 applications still in use being the exception, we use the command `python` throughout this tutorial, assuming that it should be no problem to adjust the command in your individual settings.

2.2 Installing EDICTOR3

Before installing EDICTOR3, we need to make sure to have a clean Python environment. We achieve this by setting up a virtual environment exclusively to be used with EDICTOR3. Mac and Linux users, you can prepare your virtual environment with the help of the following commands. Windows users find additional information on how to set up a virtual environment online (https://pypi.org/project/edictor/).

```
$ python -m pip install virtualenv
$ python -m venv venv/edictor
$ source venv/edictor/bin/activate
```

We can now install the EDICTOR3 application. We can again simply use thy Python package manager pip to install the edictor package.

```
$ pip install edictor
```

It is recommended to run the installation with enabling support for some other Python packages. This makes sure to have the full functionality of EDICTOR3 available, to which we will turn later. You can run the following command to install the extended packages LingPy (https://pypi.org/project/lingpy, List and Forkel 2023), LingRex (https://pypi.org/project/lingrex, List and Forkel, 2024), and LexiBase (https://pypi.org/project/lexibase) along with EDICTOR.

```
$ pip install "edictor[lingpy]"
```

Note that you can also install the three packages directly using pip.

```
$ pip install lingpy
$ pip install lingrex
$ pip install lexibase
```

If you now run the command edictor server, a browser window should open where you can access some base files and examples. Windows users will have to open the browser window directly, by passing the default URL http://localhost:9999. However, we do not have our own data ready yet, so we can close the window. This step is only for verifying that the installation has worked.

2.3 Preparing the Data

We can now turn to the preprocessing of the data we want to analyze. The first step is to download the data. As mentioned before, we will exemplify this with the blumpanotacana dataset. However, you may chose any other Lexibank dataset and replace the name accordingly (check the GitHub organization at https://github.com/lexibank/ for individual Lexibank datasets). The datasets are all curated on GitHub, and the released versions are stored on Zenodo. If you have git installed on your computer, you can simply clone the repository from GitHub:

```
$ git clone https://github.com/pano-takanan-history/blumpanotacana
$ cd blumpanotacana
```

Otherwise, you can also download the dataset directly from Zenodo (https://zenodo.org), via its DOI (https://doi.org/10.5281/zenodo.10214016). The exact command might differ between operating systems. The following example is from MacOS 15.2 for v1.2 of `blumpanotacana`.

```
$ curl https://zenodo.org/records/13149588/files/pano-tacanan-history/blumpanotacana-v1.2.zip --output blumpanotacana.zip
$ unzip blumpanotacana.zip
$ mv pano-tacanan-history-blumpanotacana-5532c2a/ blumpanotacana/
$ cd blumpanotacana
```

3 Accessing the Data

We have two options to access the data with EDICTOR: (a) using a TSV-file, or (b) using an SQLite database. For the first option, you only need to create a TSV-file using the same edictor package with the following command:

```
$ edictor wordlist --dataset=cldf/cldf-metadata.json --name=blumpanotacana --
addon="cognacy:cogid,alignment:alignment"
```

The command uses the wordlist sub-command, which is installed when selecting the `[lingpy]` option during installation of the edictor package, or when directly installing LingPy along with EDICTOR3. The first argument is the metadata file of the dataset, and `--name` provides the name of the file you create. The list in `--addon` provides additional columns, based on data from the CLDF dataset. In this case, we want the existing cognate judgements and manual alignments to be included in the EDICTOR dataset. You now receive an output file blumpanotacana.tsv. If you now run edictor servercommand again, select the tab Files in the web interface and then Open File «blumpanotacana.tsv». You can now edit and analyze the data.

The second option is to load the dataset as an SQLite database. For this purpose, we need to create such a database (again with the help of the EDICTOR package). This merely requires you to add the option `--sqlite` to the command shown above. This command will create an SQLite database called blumpanotacana.sqlite3.

```
$ edictor wordlist --dataset=cldf/cldf-metadata.json --
name=blumpanotacana --sqlite --
addon="cognacy:cogid,alignment:alignment"
```

The advantage of using SQLite databases is that any modifications that you make to your database when editing it can be stored immediately, without requiring you to saving the data actively. When working with SQLite datasets, EDICTOR will also automatically create a version history that tracks each modification done manually to the data.

However, the additional features and complexity that come with the SQLite database make it necessary to provide some information to EDICTOR how to use it. This is provided in the form of a configuration file. We recommend to store this kind of file in a new folder, which one can call `edictor/`. Within that directory, we can create a new folder where we will store all the necessary configuration files.

```
$ mkdir edictor
```

The configuration is stored in JSON format. for our example, you can create a file `config.json` that looks like the following:

```
{
  "sqlite": "./",
  "links": [
    {
      "url": "edictor.html",
      "data": {
        "file": "blumpanotacana",
        "remote_dbase": "blumpanotacana",
        "doculects":
"Matses|Kakataibo|Amahuaca|Moseten|Isconahua|Tacana",
        "columns":
"DOCULECT|FAMILY|CONCEPT|FORM|TOKENS|COGID|COGIDS|ALIGNMENT|PATTERN
S|NOTE"
      },
      "name": "Panoan (local)"}
  ]
}
```

The sqlite argument specifies the location where the database is stored. The value `./` means that it is in the current folder. Make sure that this is the case, and move the blumpanotacana.sqlite3 to the `edictor/` folder.

```
$ mv blumpanotacana.sqlite3 edictor/
$ cd edictor
```

The information in file and remote_dbase needs to be adjusted according to your database file. The `doculects` attribute specifies which language varieties you want to edit, and the attribute in `columns` points to the columns you want to be able to work with when editing your data in EDICTOR. Lastly, the name specifies the name under which you can open the database in EDICTOR. Note that you can use this to create different subsets of the same SQLite database, e.g. by specifying a second access within the links list in the config.json file. This subset accesses the same local database file (`blumpanotacana.sqlite3`), but a different list of languages, creating a virtual subset of the data.

```
 {
  "sqlite": "./",
  "links": [
    {
      "url": "edictor.html",
      "data": {
        "file": "blumpanotacana",
        "remote_dbase": "blumpanotacana",
        "doculects":
"Matses|Kakataibo|Amahuaca|Moseten|Isconahua|Tacana",
        "columns":
"DOCULECT|FAMILY|CONCEPT|FORM|TOKENS|COGID|COGIDS|ALIGNMENT|PATTERN
S|NOTE"
      },
      "name": "Panoan (local)"},
      "url": "edictor.html",
      "data": {
        "file": "blumpanotacana",
        "remote_dbase": "blumpanotacana",
        "doculects": "Araona|Tacana|Cavinena",
        "columns":
"DOCULECT|FAMILY|CONCEPT|FORM|TOKENS|COGID|COGIDS|ALIGNMENT|PATTERN
S|NOTE"
      },
      "name": "Tacanan (local)"}
  ]
}
```

4 Saving Data

Most importantly, if you decided to go for the TSV-based approach, you need to explicitly save and download your data. You can do so in the top right, where small icons indicate the respective functions. Alternatively, press CTRL + S (save) and then CTRL

+ E (download) on your keyboard. EDICTOR will always create a backup file with a time stamp that contains the data as it was before saving it the last time, and replace the content of the original file with the new content. This means, even if you decide that you work without SQLite, you can conveniently edit your data and go back in history to older versions. Since EDICTOR always stores a full version of your TSV file, however, the data may grow with time, so that you may want to actively curate your TSV file using some management system.

If you are running EDICTOR3 with an SQLite file, this becomes unnecessary. All changes are stored directly in the database, and you do not need to save or download the changes you made. You can even check the history of your annotations, but currently, you can only do so by querying the BACKUP table of the SQLite database. This table will always store an old version of a given cell content as a triple of the identifier for the word, the name of the column, and the previous value, along with information on the time of modification and the user. Querying it is thus possible to check what happened, if problems emerge, but it requires some experience with SQL and SQLite databases.

5 Outlook

This tutorial has provided a basic introduction into how to run EDICTOR locally on your computer, even without a working internet connection. The full functionality of EDICTOR3 itself is described in List and van Dam (2024). A similar workflow as the one presented in this study can be used to to analyze one's own data, or one can extract different combinations of data from the Lexibank repository to import them in EDICTOR.

References

Blum, F. and Barrientos, C. (2024). “A New Dataset with Phonological Reconstructions in CLDF” in Computer-Assisted Language Comparison in Practice, 6.1: 43–51 [first published on 21/06/2023]. https://doi.org/10.15475/calcip.2023.1.6

Blum, F., Barrientos, C., Zariquiey, R., & List, J.-M. (2024). A comparative wordlist for investigating distant relations among languages in Lowland South America. Scientific Data, 11(1). https://doi.org/10.1038/s41597-024-02928-7

Forkel, R., List, J.-M., Greenhill, S. J., Rzymski, C., Bank, S., Cysouw, M., Hammarström, H., Haspelmath, M., Kaiping, G. A., & Gray, R. D. (2018). Cross-Linguistic Data Formats, advancing data sharing and re-use in comparative linguistics. Scientific Data, 5(1), 1–10. https://doi.org/10.1038/sdata.2018.205

List, J.-M., Forkel, R., Greenhill, S. J., Rzymski, C., Englisch, J., & Gray, R. D. (2022). Lexibank, a public repository of standardized wordlists with computed phonological and lexical features. Scientific Data, 9(1), 1–16. https://doi.org/10.1038/s41597-022-01432-0

List, J.-M, van Dam, K. P., & Blum, F. (2024). EDICTOR 3. An Interactive Tool for Computer-Assisted Language Comparison [Software Tool, Version 3.1]. MCL Chair at the University of Passau. https://edictor.org

List, J.-M. and K. van Dam (2024): Computer-Assisted Language Comparison with EDICTOR 3 [Invited Paper]. In: Proceedings of the 5th Workshop on Computational Approaches to Historical Language Change. 1-11. https://aclanthology.org/2024.lchange-1.1

List, J.-M. and R. Forkel (2024): LingRex: Linguistic reconstruction with LingPy [Software, Version 1.4.2]. Max Planck Institute for Evolutionary Anthropology: Leipzig. https://pypi.org/project/lingrex

List, J.-M. and R. Forkel (2023): LingPy. A Python library for quantitative tasks in historical linguistics [Software Library, Version 2.6.13]. Version 2.6.13. MCL Chair at the University of Passau: Passau. https://pypi.org/project/lingpy

Snee, D. (2024). "Using CLDFBench and PyLexibank on Windows" in Computer-Assisted Language Comparison in Practice, 7.2: 103-109 [first published on 18/12/2024], https://doi.org/10.15475/calcip.2024.2.6

Wu, M.-S., Schweikhard, N. E., Bodt, T. A., Hill, N. W. & List, J.-M. (2020). Computer-Assisted Language Comparison: State of the Art. Journal of Open Humanities Data, 6(1), 2. https://doi.org/10.5334/johd.12

Supplementary Material
Data and code mentioned in the tutorial are all long-term archived and can be downloaded from the links provided in the study.
Funding Information
This project has received funding from the Max Planck Society as part of the CALC^3 project (https://calclab.org). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.