Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

A New Dataset with Phonological Reconstructions in Lexibank

Frederic Blum and Carlos Barrientos
Department of Linguistic and Cultural Evolution
Max Planck Institute for Evolutionary Anthropology, Leipzig

Data in historical linguistics is typically presented in non-machine-readable formats, such as text-based supplementary material or even handwritten manuscripts. Many annotations and important facts are given in prose or remain within linguists' heads. Those problems make it difficult for non-experts in the specific field to understand the data, and to reproduce and replicate the results, and also limits the exposure that linguists receive for their hard work. Similar to previous blog posts on retro-standardizing data, we present the digitization of a dataset that includes phonological reconstructions. By representing this kind of data in CLDF, we can apply a variety of computer-assisted methods to assess the quality of the reconstructions.

1 Background

With the rise of computational methods (Wu et al. 2022), annotation tools (EDICTOR, List 2017), and software for working with standardized data (LingRex, List & Forkel 2022), the possibility for working with digital data in historical linguistics has hugely increased. Yet, few datasets that involve reconstructions are openly available in a machine-readable formats. For example, only 6 out of the 100 datasets in the current release of Lexibank (List et al. 2022) include reconstructions of proto-languages, and they are quite small datasets. Only one dataset (YangLalo, Yang 2011, 2021) features more than 350 concepts, and only two datasets feature more than 10 language varieties. All six datasets have expert cognate judgements, but no dataset provides explicit alignments of the cognates. This gap will be filled by our digitization and annotation of the Proto-Panoan data originally published by Oliveira (2014). The workflow is similar to previous retro-standardizations of lexical data (Blum 2021, List 2021), but includes over 500 lexical reconstructions for the proto-language for Proto-Panoan.

Computer-Assisted Language Comparison in Practice
Volume 6, Number 1, 43-51
URL: https://calc.hypotheses.org/6142/ DOI: 10.15475/calcip.2023.1.6

Published on 21/06/2023

The Pano language family is spoken in the Amazon lowlands of Peru, Bolivia, and Brazil. It comprises 32 languages, which are spoken by an estimated 40.000-50.000 speakers (Fleck 2013). A first detailed reconstruction was presented by Shell (1965), which, however, featured only sparse data for one of the primary branches. Based on data from new documentation projects as well as his own fieldwork, Oliveira (2014) published an advanced reconstruction of Panoan. Through the standardized rework of this dataset, we aim at presenting an example case of sharing data in linguistics that involves reconstructions and explicit alignments of the data. The dataset includes 517 reconstructions for 466 different concepts. The repository with all code and data that are presented here can be found at GitHub.

2 Workflow: From PDF to CLDF

In a first step, we copy/pasted the data into text format. The immediate first problem we had to deal with was that during the extraction of the data, some non-ASCII characters were not recognized and properly extracted on the raw data file. We broke down the problem into three tasks in order to tackle the problem and recover the missing characters in the data:

1. Parse all unknown characters to a Table
2. Create a replacement table for unknown characters
3. Apply replacement table(s)

The first task was to parse all the unknown characters into a table with ID and form. For this task, we wrote a Python script that looks like the following:

```
from collections import defaultdict

def parse_line(number, text, ddct):
    for i, char in enumerate(text):
        if char not in
"a
bcef
hijklmno
pqrstuvwxyzABCDE
FGHIJKLMNOPQRSTUVWXYZ0123456789,
>:-’“.”+/éçãâàôõúë={}*'<>~ê?óñ[]íÁÂüá();‘!":
            ddct[char] += [(number, text[i-5:i + 30])]
```

This code snippet defines a function called `parse_line` that iterates over each character in the text string using the `enumerate` function. The third line checks if the character is absent in the allowed character set. If the character is not allowed, a tuple containing the number and a substring of the text is appended to a list associated with the character in the ddct dictionary. This consists of 30 characters, starting from 5 characters before the current character. The next part of the code simply opens the file with the original raw data, and applies parse_line to each line.

```
ddct = defaultdict(list)
text = open('raw_oliveira.txt', 'r', encoding="utf8").read()
text = text.split("\n")

for l in text:
    start = l.split(".")[0]
    parse_line(start, l, ddct)
```

Finally, after processing all the lines, the new for-loop iterates over the items in ddct, sorting them by length in descending order. The loop prints the key, its code point, its corresponding line number, and the sub-string of the line where the key was located. The output of this script is the table with ID and form of all unknown characters.

```
for k, v in sorted(ddct.items(), key=lambda x: len(x[1]),
reverse=True):
    print(k, "\t", hex(ord(k)), "\t", v[0][0], "\t", v[0][1])
```

With this result, we continue with the second task, namely to create a replacement table for all unknown characters. To get an impression, the first rows from the table look like the following:

Char	Replacement
\uf0e9	ɨ
\uf0a7	ʂ
\uf053	ʃ
\uf022	ˈ
\uf042	β

We noticed that there where some extracted concepts that included non-systematic inconsistencies. For example, we had the following:

- “com, empossede” instead of “com, em posse de”
- “esp. deplanta” instead of “esp. de planta”

The same happened with some sources:

- ANONBY2010 instead of Anonby2010
- CHÁVEZ2012 instead of Sparing2012
- VALENZUELA2003;LORIOT;DAY1993 instead of Valenzuela2003;Loriot1993

In order to fix this, we created a replacement table both for concepts and for sources. The last step was to apply all these replacement tables to the data in order to get the best-possible text. At this point, however, the text contained still unparseable parts, therefore we modified them manually to avoid not working with them.

We wrote a script to help us with the replacements and to print us the erroneous lines that needed to be modified manually. For this purpose, we kept the original extracted data, but we continued working on a second file. With the first part of the code, we load the replacement tables and apply it to the data:

```
with open("replacements.tsv", encoding="utf8") as f:
rep = {}

for line in f.readlines():
    a, b = line.split("\t")
    if a != "CHAR":
        rep[eval('"'+a+'"')] = b.strip()

concept_replace = {}
with UnicodeDictReader("concept_replacements.tsv", delimiter='\t')
as reader:
    for line in reader:
        concept_replace[line["OLD"]] = line["NEW"]

source_replace = {}
    with UnicodeDictReader("source_replacements.tsv",
delimiter='\t') as reader:
    for line in reader:
        source_replace[line["OLD"]] = line["NEW"]
```

Then, we loaded the data and fragment them into numbered lists of information we needed:

```
with open("raw_oliveira-mod.txt", encoding="utf-8") as f:
    data = OrderedDict()
    additional_comments = []

    for line in f:
        line = "".join([rep.get(c, c) for c in line])
        idx, rest = line[:line.index(".")],
line[line.index(".")+1:].strip()
```

We made us of the systematic formating provided by Oliveira, like preceding reconstructions using an asterisk, and putting concepts into brackets, to derive the final form for all entries. During this process, we have tagged erroneous lines to be preceded with an exclamation mark. This way, we could easily adapt individual solutions for the different cases, or change the corresponding lines manually when necessary.

```
print("# Found {0} lines without spaces\n".format(len(bad_lines)))
print("Number | Line\n--- | ---")
for i, line in enumerate(bad_lines):
    print("{0:5} | {1}".format(i+1, line))
    print("")
    print("# Found {0} entries with individual problematic
entries\n")

print("Number | Line ID | Entry\n--- | --- | ---")

for i, (a, b) in enumerate(bad_entries):
    print("{0} | {1:20} | {2}".format(i+1, a, b))
```

As the script is extensively commented, we will not go into detail of the individual replacements. In cases where the author does not provide a concept for a form, we add an uncertainty-tag to a new column, and add the parsed proto-concept. We also parsed information on additional sources for individual entries from bracketed notes in the data, and added them to a specific column for sources. Finally, once all the problematic entries were fixed, we create the final version of the raw data, which is then employed during the CLDF conversion.

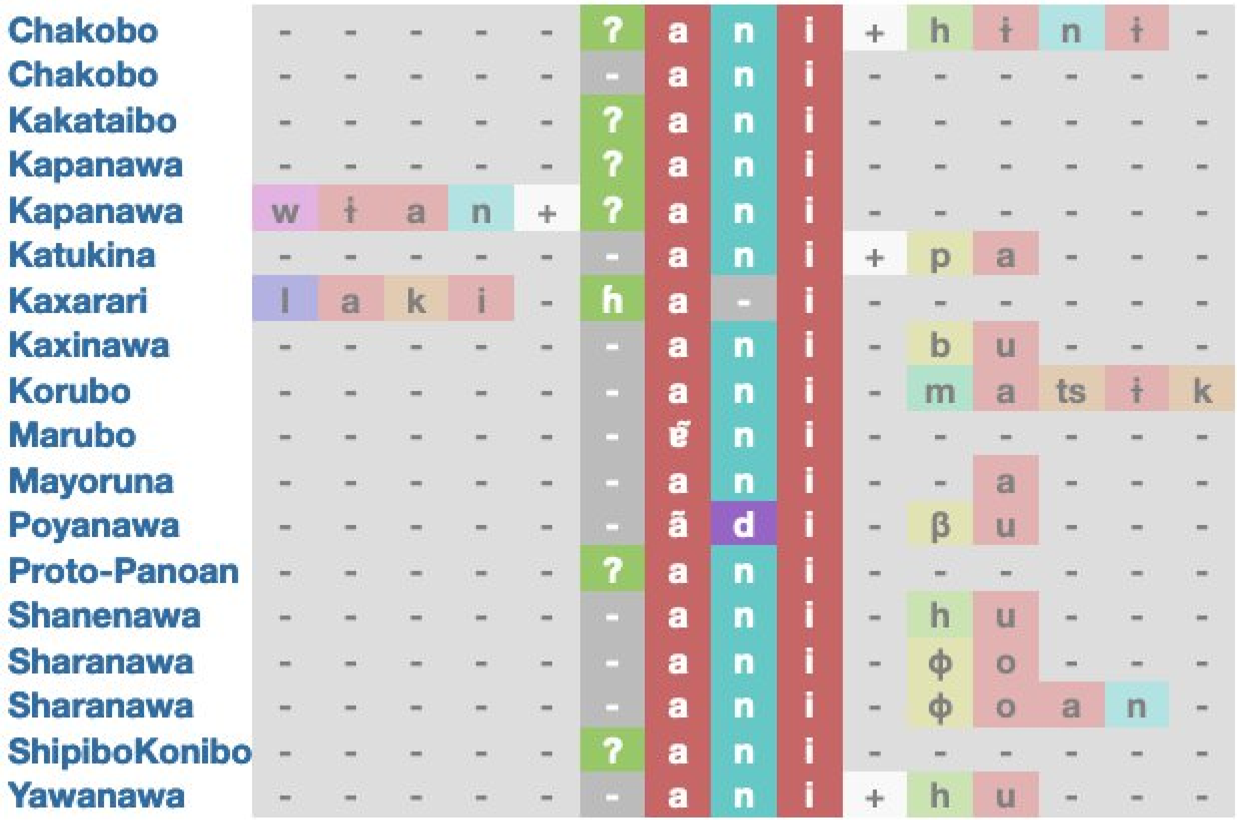
To retrieve the concepts of the proto-forms in the data, we wrote a small script `get_concepts.py`, where we gather the `proto-concepts` and `other_concepts` as the ones from the doculects. This conceptlist is uploaded as Oliveira-2014-517 to Concepticon (List et al. 2023).

We then followed up with the standard CLDF conversion for Lexibank-datasets using cldfbench (Forkel & List 2020). In order to upload the data to EDICTOR, the software with which we annotate and align the data, we convert the CLDF dataset to a SQLite database. The final dataset includes data from 16 languages with a coverage of more than 60%. In total, the dataset includes 7,307 lexemes, of which 292 lexemes have been tagged as singletons due to uncertain cognacy.

3 Annotating the Data in EDICTOR

3.1 Morpheme Segmentation

The first step in EDICTOR consists of preparing the alignments for all words in a cognate set. This involves the flagging of non-cognate material which is present in individual languages, and that is not reconstructed for the proto-language, such as derivational morphemes. While for some cognate sets this is not necessary, other sets have quite complex morphology involved. Preparing those alignments takes time, but sets up the database for high-quality inferences of correspondence patterns. We present an example of this trimming of alignment sites in Figure 1.



Chakobo	-	-	-	-	-	?	a	n	i	+	h	ɨ	n	ɨ	-
Chakobo	-	-	-	-	-	-	a	n	i	-	-	-	-	-	-
Kakataibo	-	-	-	-	-	?	a	n	i	-	-	-	-	-	-
Kapanawa	-	-	-	-	-	?	a	n	i	-	-	-	-	-	-
Kapanawa	w	ɨ	a	n	+	?	a	n	i	-	-	-	-	-	-
Katukina	-	-	-	-	-	-	a	n	i	+	p	a	-	-	-
Kaxarari	l	a	k	i	-	h	a	-	i	-	-	-	-	-	-
Kaxinawa	-	-	-	-	-	-	a	n	i	-	b	u	-	-	-
Korubo	-	-	-	-	-	-	a	n	i	-	m	a	ts	ɨ	k
Marubo	-	-	-	-	-	-	ẽ́	n	i	-	-	-	-	-	-
Mayoruna	-	-	-	-	-	-	a	n	i	-	-	a	-	-	-
Poyanawa	-	-	-	-	-	-	ã	d	i	-	β	u	-	-	-
Proto-Panoan	-	-	-	-	-	?	a	n	i	-	-	-	-	-	-
Shanenawa	-	-	-	-	-	-	a	n	i	-	h	u	-	-	-
Sharanawa	-	-	-	-	-	-	a	n	i	-	ɸ	o	-	-	-
Sharanawa	-	-	-	-	-	-	a	n	i	-	ɸ	o	a	n	-
ShipiboKonibo	-	-	-	-	-	?	a	n	i	-	-	-	-	-	-
Yawanawa	-	-	-	-	-	-	a	n	i	+	h	u	-	-	-

Figure 1: Removing non-cognate material for BIG

3.2 Uncertainty of Reconstructions

The original author incorporated uncertainty in his reconstructions in two exemplary ways. First, he tagged word forms explicitly if he had doubts about the cognacy of those lexemes. In order to infer the best sound correspondence patterns possible from this data, we excluded all those words from the respective cognate set and tagged them as singleton. As Oliveira's reconstruction is partially based on previous reconstructions established by Shell (1965) this tagging provides a critical reassessment of that earlier data. Second, he tagged as uncertain the segmental quality of certain reconstructed phones, for which one or two crucial languages do not have known reflexes. We can search for those cases explicitly in EDICTOR, making use of the search function. A computer-assisted check of those phones can provide a valuable starting point for further analysis and to check the systematicity of those uncertainty judgements.

4 Computational Analysis of Reconstructions

We can use the aligned data to perform a variety of computational checks, of which we will present one exemplary case. In a pre-processing step, singletons and multiple (292) and multiple entries per language/concept have been removed. Using the regularity-measures of Blum & List (2023) that are part of LingRex, we find that of the 735 inferred correspondence patterns, 318 occur at least twice over the dataset. This means that 417 patterns occur only once, making them a problematic case for the regularity of sound change. A closer investigation of those patterns will show whether they are due to erroneous alignments, or due to errors and sporadic sound changes in the data. For the word level, we count how many of the patterns in the cognate set are considered to be recurring, and establish a regularity threshold at 70%. From the 6128 remaining words, 4517 are above this threshold, and 1611 are considered to be irregular. 168 words are even below a threshold of 30%. Those cases should be investigated to see whether erroneous cognate judgments are part of the dataset. A visual comparison of the results applying different regularity threshold is presented in Figure 2.

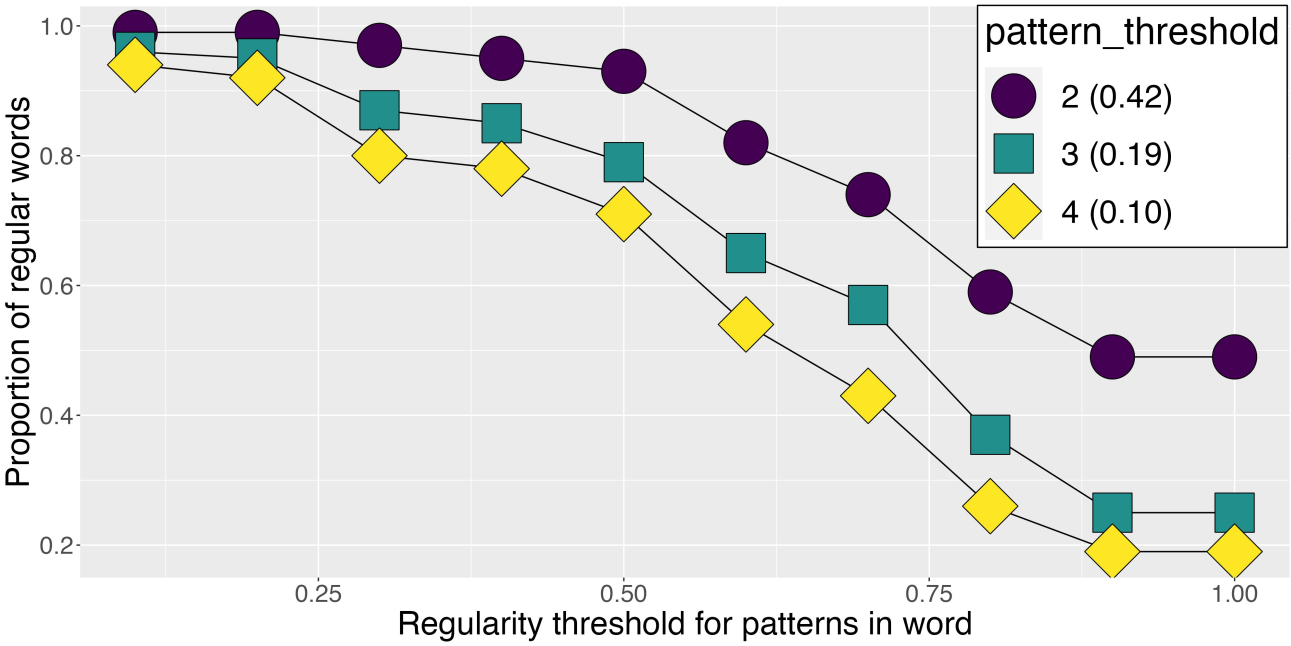


Figure 2: Word regularity for different thresholds for regularity in correspondence patterns and across the cognate set.

There are several interesting observations about this. First, around half of the entries are fully explainable with correspondence patterns that occur at least twice. Second, At a word-threshold of about 50%, a considerable drop-off in the proportion of regular words can be observed. Third, the differences between a pattern-threshold of 3 and a pattern-threshold of 4 is small. If a pattern occurs at least three times, the chances are high it will recur more often as well. The difference to correspondence patterns occurring only twice is considerable, however. Based on those very preliminary results, we might propose that a threshold of three is more stable than a lower threshold.

5 Conclusion and Outlook

We show how the digitization of a dataset from historical linguistics can offer new tools and perspectives on the computer-assisted comparison of phonological reconstructions. By adding explicit alignments of the data and removing non-cognate material, we arrive at high scores of regularity. By identifying patterns and cognate sets that are below certain regularity threshold, we can single out cases which should be analyzed more thoroughly. In the future, we will work on additional methods that help identifying problematic patterns and cognate sets, to provide a quantitative assessment of proto-language reconstructions. At the same time, we call for more researchers to share their data in machine-readable files. This step would ensure that historical linguistics heads towards the reproducibility and replicability of the results in the field.

Data and Code

You can find all the data and code we have used for this digitization at our GitHub-repository https://github.com/pano-tacanan-history/oliveiraprotopanoan. The current version (v1.0.0) is archived on Zenodo (10.5281/zenodo.8058801).

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 & Johann-Mattis List. 2023. Trimming Phonetic Alignments Improves the Inference of Sound Correspondence Patterns from Multilingual Wordlists. In Proceedings of the 5th Workshop on Research in Computational Linguistic Typology and Multilingual NLP, 52–64. Dubrovnik, Croatia: Association for Computational Linguistics. https://aclanthology.org/2023.sigtyp-1.6.

Fleck, David W. 2013. Panoan languages and linguistics. Anthropological Papers of the American Museum of Natural History 99. http://dx.doi.org/10.5531/sp.anth.0099.

Forkel, Robert & Johann-Mattis List. 2020. CLDFBench: Give your cross-linguistic data a lift. In 12th Conference on Language Resources and Evaluation, 6995–7002. https://aclanthology.org/2020.lrec-1.864.

List, Johann-Mattis. 2017. A web-based interactive tool for creating, inspecting, editing, and publishing etymological datasets. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. System Demonstrations, pages 9–12, Valencia. Association for Computational Linguistics. https://aclanthology.org/E17-3003.

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 & Robert Forkel. 2022. LingRex. Linguistic Reconstruction with LingPy. https://doi.org/10.5281/zenodo.6420629.

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.

List, Johann Mattis & Tjuka, Annika & van Zantwijk, Mathilda & Blum, Frederic & Ugarte, Carlos Barrientos & Rzymski, Christoph & Greenhill, Simon & Forkel, Robert (eds.) 2023. CLLD Concepticon 3.1.0 [Data Set]. Zenodo. https://doi.org/10.5281/zenodo.777762.

Oliveira, Sanderson Castro Soares de. 2014. Contribuições para a reconstrução do Protopáno. Brasília: Universidade de Brasília. https://repositorio.unb.br/handle/10482/17129.

Shell, Olive Alexandra. 1965. Pano Reconstruction. University of Pennsylvania.

Wu, Mei-Shin, Nathanael E. Schweikhard, Timotheus A. Bodt, Nathan W. Hill & Johann-Mattis List. 2020. Computer-Assisted Language Comparison: State of the Art. Journal of Open Humanities Data. 6(1). 2. https://doi.org/10.5334/johd.12.

Yang, Cathryn (2011): Lalo regional varieties: Phylogeny, dialectometry and sociolinguistics. Bundoora: La Trobe University.

Yang, Cathryn. (2021). CLDF dataset derived from Yang's "Lalo Regional Varieties" from 2011 (v3.0) [Data Set]. Zenodo. https://doi.org/10.5281/zenodo.5121829.