Computing Detailed Colexifications with Missing Data Information from the CLICS^4 Collection

David Snee and Johann-Mattis List
Chair for Multilingual Computational Linguistics
University of Passau

CLICS^4 offers a refined structural representation of cross-linguistic colexification patterns but retains an implicit representation of missing data. This obscures whether the lack of a colexification in a language for paired concepts is due to its true absence in the language, or due to missing data on the concept or word form level. We introduce a straightforward workflow that can be applied to individual datasets from CLICS^4 to identify cases of colexification via a three-way attestation scheme. Our approach captures the presence or absence of a colexification in CLICS^4, but it also explicitly encodes the presence or absence of data at the level of the original questionnaire, or the individual language, elicited with the help of the questionnaire.

1 Introduction

CLICS^4 (Tjuka et al. 2025) provides a much clearer picture of cross-linguistic colexification data than what has been provided by CLICS^3 (Rzymski et al. 2020), making up for certain problems observed when working with CLICS^3. Among the changes introduced in CLICS^4 are that (1) only one language variety per Glottocode was selected, using the frequency of attested concepts as the criterion for selection; (2) broad concepts known to colexify senses are split, individual forms for the individual senses are sampled (e.g. “ARM OR HAND” being reflected as “ARM” and “HAND”, respectively), (3) certain concepts were represented by more common ones (e.g., “KNOW” → “KNOW SOMETHING”); and (4) not all concepts were sampled, instead a cutoff was made, resulting in 1730 different concepts in CLICS^4.

CLICS^4 also represents the colexification data differently, offering a structural dataset view in which each of the colexifications observed in CLICS^4 is treated as a feature for each language. We list the feature value as 1 if the colexification is observed and 0 if it is not observed. A feature is not listed if either of its two colexifying forms is missing from the data.

Computer-Assisted Language Comparison in Practice
Volume 9, Number 1
URL: https://calc.hypotheses.org/9164 DOI: 10.15475/calcip.2026.1.2
Published under a Creative Commons Attributions 4.0 LICENSE
Published on 23/02/2026

However, while we have a detailed account of whether a form in the data in CLICS^4 is missing, we do not know why this would be the case. We can identify at least three different reasons, namely that (1) the form is not expressed in the language ("mosquitoes" do not exist in mountainous regions of the Himalayas); (2) the form was not elicited in the concept list (it was never asked for); and (3) the form was elicited in the concept list, but not for particular languages, resulting again in missing data.

While we cannot decide for case (1), we can easily identify case (2), and based on this also case (3). However, given that the CLICS^4 data do not offer direct accounts on these missing data types, it is important for us to add this analysis to the dataset. In the following, we will show how this can be done.

2 Data Preparation

2.1 Install Packages

To begin adding this layer of analysis, we assume you have Python (https://python.org, 3.8 or higher) and GIT installed on your system and that you are operating within a virtual environment with command line access. Once you have this confirmed, you need to install pycldf and tqdm to run the workflow.

```
$ pip install pycldf
$ pip install tqdm
```

2.2 Download Data

Subsequently, we must download the individual datasets we wish to work with in the latest versions used in CLICS^4 to be able to work with the concept lists, as these are not stored within CLICS^4 itself. The datasets are retrieved programmatically using GIT via the GitPython library.

```
$ pip install GitPython
```

We then create a new folder, raw, and copy the file lexibank.tsv from the etc directory of the CLICS^4 dataset repository on GitHub into our project repository. This file is essentially the same as the one that is used as part of the Lexibank database (List et al. 2022), but CLICS^4 does not use all of the data that are aggregated by Lexibank. To download the datasets, we run the short script shown below. For illustrative purposes, our current test case downloads only the first 20 datasets listed in lexibank.tsv. This can, of course, be extended as required.

```python
import csv
from git import Repo
from pathlib import Path

count = 0
with open("lexibank.tsv") as f:
    for row in csv.DictReader(f, delimiter="\t"):
        if row["ClicsCore"] == "x":
            print(f"Cloning {row['Organization']} /
{row['Dataset']}...")
            pth = Path(__file__).parent / "raw" / row["Dataset"]
            if not pth.exists():
                Repo.clone_from(
                    f"https://github.com/{row['Organization']}"
                    "/{row['Dataset']}.git",
                    pth,
                    branch=row["Version"]
                )
            else:
                print("... quitting, path exists")
            count += 1
            if count == 20: # extend as required
                break
```

This reads in the major information on our selected repositories in CLICS^4 and then successively downloads them, with the versions that made it into CLICS^4. Note that we found one error in the organization of the keypano dataset [presented in Miller and List (2024)], so we recommend using the updated lexibank.tsv file from the Codeberg repository included in the supplementary material of this study.

2.3 Compile SQLite-Version of CLICS^4

In order to obtain CLICS^4 in SQLite, please execute both steps below in the command line:

```
$ git clone https://github.com/clics/clics4 --depth=1 --branch=v0.5
$ cldf createdb clics4/cldf/Wordlist-metadata.json clics4-
wordlist.sqlite3
```

This may take a considerable amount of time but it speeds up all further operations. For convenience, we have uploaded the SQLite database containing the CLICS^4 wordlist to Zenodo, where you can find it online, providing Version 0.5 of CLICS^4 (see https://doi.org/10.5281/zenodo.18562928). Therefore, you can alternatively download the SQLite database from the command line via Zenodo using the curl command:

```
$ curl 'https://zenodo.org/records/18562928/files/clics4-
wordlist.db?download=1' --output clics4-wordlist.sqlite3
```

3 Missing Data Computation

3.1 Filtering for Selected Languages and Datasets

For our input data, we share a preliminary list of 20 Glottocodes of interest and store these Glottocodes in the file languages.csv. These 20 languages are loaded from this list using the code below:

```
import csv
import codecs

# load Glottocodes
with codecs.open("languages.csv", "r", "utf-8") as f:
    glottocodes = [line.strip() for line in f if line.strip()]
print('[i] loaded Glottocodes')
```

Next, the 20 downloaded datasets from the raw directory are loaded.

```
# get datasets in raw directory
raw_dir = Path(__file__).parent / "raw"
raw_datasets = [p.name for p in raw_dir.iterdir() if p.is_dir()]
print('[i] loaded datasets')
```

We then prepare our 20 selected languages and 20 selected datasets for later filtering using SQLite.

```
# sample languages and sample datasets for filtering
sample_glottos = ",".join("?" for _ in glottocodes)
sample_datasets = ",".join("?" for _ in raw_datasets)
```

3.2 Restricting the Number of Concepts

We also share a preliminary list of 20 concepts of interest. These concepts are stored in the file concepts.csv and our analysis is restricted to only those concepts as they occur for our selected languages in our selected datasets. We load the data as follows in Python:

```
import csv
import codecs
# load concepts from CSV
with codecs.open("concepts.csv", "r", "utf-8") as f:
    concepts = [c.strip() for c in f.read().strip().split("\n")]
print('[i] loaded concepts')
```

It is also important for us to load the list of modifications shared as part of CLICS^4. These modifications reflect cases where broad concepts known to colexify were split into more specific senses (e.g. “ARM OR HAND” being reflected as “ARM” and “HAND”, respectively), or where certain senses were replaced by more common alternatives. Here, for convenience, we download the file of modified concepts from the etc folder in CLICS^4 and load the data as follows:

```
# load concept modifications from TSV
with codecs.open('concept-modifications.tsv', 'r', 'utf-8') as f:
    modifications = {}
    for row in csv.DictReader(f, delimiter='\t'):
        modifications[row['Source']] = row['Targets'].split(' // ')
print('[i] loaded concept modifications')
```

Note that we can modify this further, by using Python’s pathlib. However, for the sake of simplicity, this step is omitted here. Consequently, we must load the code from the base folder from which we carry out all our operations and in which we placed the CSV files.

Also, in case some concept labels in our concepts.csv contain single quotes, these are escaped to later prevent SQLite syntax errors.

```
# escape single quotes in concepts to avoid syntax errors
escaped_concepts = [c.replace("'", "''") for c in concepts]
concept_string = "»«".join(escaped_concepts)
```

3.3 Create the SQLite Query

Now we connect to the SQLite database and construct the query that helps us to retrieve only those concepts from our concept list (concepts.csv) that are attested in the selected languages and datasets.

```
import sqlite3

# connect to SQLite
db = sqlite3.connect("clics4-wordlist.sqlite3")
cursor = db.cursor()

# SQL query
cmd = f"""
SELECT
    l.cldf_id,
    l.cldf_name,
    l.cldf_glottocode,
    l.cldf_latitude,
    l.cldf_longitude,
    l.cldf_contributionReference,
    l.Family,
    p.cldf_concepticonReference,
    p.concepticon_gloss,
    f.cldf_id,
    f.cldf_segments
FROM LanguageTable AS l, ParameterTable AS p, FormTable AS f
WHERE
    l.cldf_id = f.cldf_languageReference
    AND p.cldf_id = f.cldf_parameterReference
    AND l.cldf_glottocode IN ({sample_glottos})
    AND l.cldf_contributionReference IN ({sample_datasets})
    AND '«{concept_string}»' LIKE '%«' || p.concepticon_gloss ||
'»%';
"""

cursor.execute(cmd, glottocodes + raw_datasets)rows =
cursor.fetchall()
```

3.4 Prepare the Wordlist Data

Next, we can load the wordlist data by fetching the result of this query, and we can also make a lookup for the languages by their identifier.

```python
from collections import defaultdict

datasets = defaultdict(lambda: defaultdict(list))
langs = {}
varieties = defaultdict(list)

# organize rows by dataset/language and store language info
for row in rows:
    datasets[row[5]][row[0]] += [row]
    langs[row[0]] = row[1:7]
    varieties[row[0]] += [row]

print('[i] fetched wordlist data')
```

3.5 Search for Missing Data

We are now ready to search directly for the two different kinds of missing data that we discussed above. For this purpose, we use Python's pathlib, since it is more convenient for the handling of nested paths.

```python
from pathlib import Path

# retrieve concept lists from data
missing_from_data = {}
missing_from_language = {}
```

```
for dataset, languages in datasets.items():
    # assemble all concepts from the concept list
    print(f'[i] analyzing concepts for dataset {dataset}')

    with codecs.open(
            Path(__file__).parent / "raw" / dataset / "cldf" /
            "parameters.csv",
            "r",
            "utf-8") as f:
        clist = []
        for row in csv.DictReader(f, delimiter=','):
            clist += modifications.get(row['Concepticon_Gloss'],
            [row['Concepticon_Gloss']])
        missing_from_data[dataset] = list(set([c for c in concepts
        if c not in clist]))

    for language, words in languages.items():
        current_concepts = []
        for word in words:
            current_concepts += modifications.get(
                word[8], [word[8]])
        missing_from_language[language] = list(set(
            [c for c in concepts
             if c in clist and c not in current_concepts]))
```

At this stage, we have a clear account for each selected language in the CLICS^4 database. Thus, for the language with ID “753”, we can now check for those concepts from our concept list that could be expressed, since they occur in the concept list of the bodtkhobwa dataset for this language, but were not expressed in this variety. We can also inspect what concepts from our concept list are missing from the bodtkhobwa concept list.

```
# Inspect missing forms and concepts for a selected language ID and
dataset
lang_id = '753'
dataset_id = 'bodtkhobwa'
lang_name = langs[lang_id][0]
dataset_name = langs[lang_id][4]

print(f"[i] sample language: {lang_name}")
print(f"[i] {lang_name} has {len(missing_from_language[lang_id])}"
      f"missing form(s): "f"{missing_from_language[lang_id]}")
print(f"[i] {dataset_name} has
{len(missing_from_data[dataset_id])}"
      f"missing concept(s): "f"{missing_from_data[dataset_id]}")
```

This produces the following output for this language:

```
[i] sample language: Rahung
[i] Rahung has 1 missing form(s): ['OLDER SISTER (OF MAN)']
[i] bodtkhobwa has 7 missing concept(s): ['HILL', 'CALCULATE',
'HUNGER', 'PAIN', 'LAMP', 'BLAME', 'SCARED']
```

Our output shows that there is 1 attested concept for this variety in bodtkhobwa from our concept list without a corresponding form. It also indicates that there are 7 concepts in our concept list that are not attested in the concept list of bodtkhobwa.

3.6 Searching for Colexifications

At this point, we could retrieve the colexifications from the structural data from CLICS[4]. However, since the full dataset is very large, it would not be very efficient to process everything, especially when we are currently only interested in a small subset of languages, datasets and concepts. Instead, we suggest recomputing colexifications on the fly for our selected data. This does not require much time, as we can use a hash-table (or a dictionary) to efficiently compare words within each language, allowing us to avoid blowing up the search space [List (2022)]. We thus apply this approach here now, using the following code:

```python
# detect colexifications
print("[i] checking colexifications")
colexifications = defaultdict(list)

for variety, words in varieties.items():
    cols = defaultdict(list)
    for word in words:
        cols[word[10]] += [word[8]]

    for colexified in cols.values():
        # Deduplicate to avoid identical concepts pairing with
themselves
        unique_concepts = list(set(colexified))
        if len(unique_concepts) > 1:
            for a, b in combinations(unique_concepts, r=2):
                if a > b:
                    colexifications[a, b] += [variety]
                else:
                    colexifications[b, a] += [variety]

print(f"[i] found {len(colexifications)} colexifications")
```

Having inferred all colexifications from the data, we now check for individual missing data types. Since missing data can involve two concepts that make up a colexification, concept A and concept B, we characterize both concepts using our three-way scheme.

We first code for the colexification itself as present (1), absent (0, if both concepts are attested), or unattested due to missing data (-1, if one or both concepts lack a translation). We then add the information for the individual concepts, coded as present (1), absent from the language but present in the concept list (0), or absent from the concept list (-1).

The scheme can be easily adjusted, but it captures all the information we can reliably infer from CLICS^4 data at present. To track how long this operation takes, we use the tqdm package to provide active feedback.

```
from tqdm import tqdm

# prepare data per language
wpl = {k: set([w[8] for w in v]) for k, v in varieties.items()}

col_by_lang = {(a, b): {l: [0, 0, 0] for l in langs} for a, b in
colexifications}

for (a, b), matches in tqdm(colexifications.items(), desc="iterate
over colexifications"):
    for l, ldat in langs.items():
        if l in matches:
            col_by_lang[a, b][l] = [1, 1, 1]
        else:
            if a in wpl[l] and b in wpl[l]:
                # true missing data
                col_by_lang[a, b][l] = [0, 1, 1]
            else:
                col_by_lang[a, b][l][0] = -1
                if a in missing_from_language[l]:
                    col_by_lang[a, b][l][1] = 0
                elif a in missing_from_data[ldat[4]]:
                    col_by_lang[a, b][l][1] = -1
                if b in missing_from_language[l]:
                    col_by_lang[a, b][l][2] = 0
                elif b in missing_from_data[ldat[4]]:
                    col_by_lang[a, b][l][2] = -1
```

Depending on the number of inspected languages, datasets, and concepts, the resulting data can be quite large when written to file. We chose this format anyway as we assume that this format can be handled properly with a modern computer and that a tabular format is what will be needed for any future processing. In our test case here our output to colexifications.tsv is relatively small, as our sample is restricted to 20 languages, 20 datasets, and 20 concepts.

```
# write the output to TSV
with codecs.open("colexifications.tsv", "w", "utf-8") as f:
    # header
    f.write("\t".join([
        "Concept_A",
        "Concept_B",
        "Language_ID",
        "Language",
        "Glottocode",
        "Dataset",
        "Family",
        "Colexification",
        "Attestation_A",
        "Attestation_B"
    ]) + "\n")

    for (a, b), langs_data in tqdm(col_by_lang.items(), desc="write
to file"):
        for l, vals in langs_data.items():
            ldat = langs[l]
            row = [
                a,           # Concept_A
                b,           # Concept_B
                l,           # Language_ID
                ldat[0],     # Language
                ldat[1],     # Glottocode
                ldat[4],     # Dataset
                ldat[5],     # Family
                str(vals[0]),  # Colexification
                str(vals[1]),  # Attestation_A
                str(vals[2])   # Attestation_B
            ]
            f.write("\t".join(row) + "\n")
```

4 Inspection of Results

Even from our small sample, major take-home messages from this experiment can be found from the numbers. We have a lot of concepts missing already at the level of the original questionnaires used. Thus, large parts of missing data can be explained by the fact that the questionnaires were too small and did not cover the concepts of interest. A review of the colexifications.tsv file also indicates that we have many cases of missing translations for attested concepts.

Nevertheless, in our small sample we find 6 colexifications. We can see in colexifications.tsv that these colexifications occur in some languages and not in others. Importantly, we can distinguish whether a non-occurrence in a language reflects a true absence of the colexification or is due to missing data, down to the level of the individual concept. Additional data coverage statistics, like the number of missing concepts, the number of concepts missing in individual varieties and not in the questionnaires, and the like, can easily be computed from the data produced in this format.

An advantage of this dataset is also that missing data are not provided in implicit form, as we have been doing in CLICS^4, where only information on known colexifications (1) and known missing colexifications (0) was listed in the structural dataset. We decided against additionally listing missing data in CLICS^4 as the resulting dataset would have been too large to share in CLDF formats (Forkel et al. 2018). But our example here shows that we can easily retrieve these data in a targeted manner, specifically when we wish to retrieve these colexification data for a specific subset of languages, datasets, and concepts.

5 Conclusion

What should follow now from this example would be a more thorough evaluation of the colexification findings with respect to the numbers. We know that the data in CLICS^4 are quite skewed and that for only a few concepts we find a reliable overlap across multiple languages. However, what this means for individual samples of colexifications has not been investigated so far. The code provided here can help to close this gap.

References

Forkel, Robert, Johann-Mattis List, Simon J. Greenhill, Christoph Rzymski, Sebastian Bank, Michael Cysouw, Harald Hammarström, Martin Haspelmath, Gereon A. Kaiping, and Russell D. Gray. 2018. Cross-Linguistic Data Formats, Advancing data sharing and re-use in comparative linguistics. Scientific Data 5 (180205): 1–10. https://doi.org/https://doi.org/10.1038/sdata.2018.205.

List, Johann-Mattis. 2022. “How to compute colexifications with CL Toolkit (How to Do x in Linguistics 10).” Computer-Assisted Language Comparison in Practice 5 (6). https://calc.hypotheses.org/4266.

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

Miller, John, and Johann-Mattis List. 2024. Adding standardized transcriptions to Panoan and Tacanan languages in the Intercontinental Dictionary Series.” Computer-Assisted Language Comparison in Practice 7 (2): 69–77. https://doi.org/10.15475/calcip.2024.2.3.

Rzymski, Christoph, Tiago Tresoldi, Simon Greenhill, Mei-Shin Wu, Nathanael E. Schweikhard, Maria Koptjevskaja-Tamm, Volker Gast, et al. 2020. “The Database of Cross-Linguistic Colexifications, Reproducible analysis of cross-linguistic polysemies.” Scientific Data 7 (13): 1–12. https://doi.org/10.1038/s41597-019-0341-x.

Tjuka, Annika, Robert Forkel, Christoph Rzymski, and Johann-Mattis List. 2025. Advancing the Database of Cross-Linguistic Colexifications with new workflows and data. In Proceedings of the 16th International Conference on Computational Semantics, 1–15. https://aclanthology.org/2025.iwcs-main.1.

Supplementary Material

Data and code can be found at https://codeberg.org/calc/colex-coverage (Version 1.0).

Acknowledgements

We thank George Gillet for bringing up the idea of computing missing data in CLICS^4 in a more principled manner.

Funding Information

This project has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (Grant agreement No. 101044282). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.