Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

How to Visualize Colexification Networks with JavaScript and D3 (How to do X in Linguistics 12)

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

Having seen how colexifications can be inferred and how colexification networks can be computed in previous posts, this post concludes our mini series in showing how computed colexification networks can be visualized interactively, using a JavaScript application based on the popular visualization library D3.

1 Background

When we published the first version of the Cross-Linguistic Colexification database in 2014 (List et al. 2014), which can still be accessed online (see https://clics.lingpy.org), although we have long flagged it as being out of date, one of the major components we included in this otherwise not very professional web application was a visualization component that made use of the popular D3 visualization library (Bostock 2011) in order to provide an interactive visualization of individual parts of the colexification networks in the CLICS database, which users could use to explore the data from multiple perspectives at the same time. This visualization component, which we discussed in detail in a paper by Mayer et al. (2014) has proven so successful that we kept using it in the new installations of CLICS in 2018 (List et al. 2018) and 2020 (Rzymski et al. 2020. When I decided to write this small series of posts that introduce step-by-step how one can compute colexification networks with our new tools (List and Forkel 2021) that make active use of existing CLDF datasets published along with the Lexibank repository (List et al. 2022), I realized that it would also be useful to show how this interactive visualization tool could be applied to individual CLDF datasets.

I have neither the knowledge nor the time to provide a detailed introduction to the whole JavaScript application that we use to visualize CLICS data. For this reason, this

Computer-Assisted Language Comparison in Practice
Volume 5, Number 8
https://calc.hypotheses.org/4351/

Published on 24/08/2022

tutorial won't focus on the JavaScript code. What I will do instad is providing a short introduction of what users must do in order to create their own JavaScript application that allows them to browse their colexifications interactively. IN order to do so, I will provide the major JavaScript package in slightly modified form from the original code by Mayer et al. (2014), which was mainly written by Thomas Mayer, and then show, how those parts of the data which the application needs can be computed. The code itself builds on the code examples which I have explained in previous posts, but differs with respect to some additional data which are now written to file.

2 Getting Started

Users who wish to thest this code on their own data or on their own selection of CLDF datasets can just clone these datasets from the Lexibank repository (https://github.com/lexibank) and then modify the code accordingly.

Since the code for the visualization now also requires some static files in HTML, CSS, and JavaScript, I decided to share the code this time in the form of a public GitHub repository instead of a GitHub Gist. This repository can be found at https://github.com/clics/clicsviz. In order to use it, just clone the repository and install the Python requirements.

```
$ git clone https://github.com/clics/clicsviz
$ cd clicsviz
$ pip install -r requirements.txt
```

We will need the same dependencies as those mentioned in the two previous posts written before (List 2022a and List 2022b, so I won't give a detailed introduction into these packages here but rather refer interested readers to the previous posts.

In order to illustrate the code, I will use the data underlying the Intercontinental Dictionary Series (Key and Comrie 2016) along with an additional dataset on Chinese, which should be stored in the folder cldf-datasets and can both be easily cloned from GitHub with the help of GIT.

```
$ cd cldf-datasets $ git clone https://github.com/intercontinental-dictionary-series/ids.git
$ git clone https://github.com/intercontinental-dictionary-series/yuchinese.git
```

3 Computing the Data

The application that will contain the JavaScript code we need for the interactive browsing of CLICS data is stored in the folder app of the repository. here, we have one folder with

the JavaScript code (js), one folder in which we will store the individual graphs in JSON format (cluster), and one folder in which we will add our individual configuration data, a JavaScript file storing the names for Infomap clusters (infomap-names.js), a file providing information on the locations of the language varieties in GeoJSON format (langs.Geo.json), and a file providing access to the individual words from the CLDF datasets (words.json). These different files all need to be produced by us in order to make this app work.

We start by importing the libraries that we will need.

```
from cltoolkit import Wordlist
from pycldf import Dataset
from collections import defaultdict
from itertools import combinations
import json
from lingpy.convert.graph import networkx2igraph
from networkx.readwrite import json_graph
from clldutils.misc import slug
import networkx as nx
```

We now first define a function that converst a graph that is provided as a networkx object to JSON format.

```
def graph2json(graph, filename):
    """ Compute graph to JSON format. """
    # convert to json
    jdata = json_graph.adjacency_data(graph)
    # write to file
    f = open(filename+'.json','w')
    json.dump(jdata,f)
    f.close()
```

We then define the function for the computation of the colexifications which we already used in the previous posts.

```
def get_colexifications(language, data):
    """
    Compute colexifications and add them to the data dictionary.
    """
    tmp = defaultdict(list)
    for form in language.forms:
        if form.concept:
```

```
            tmp[form.form] += [(form.concept.concepticon_gloss, form)]

    for forms, colset in tmp.items():
        concepts = set([f[0] for f in colset])
        for (cA, fA), (cB, fB) in combinations(colset, r=2):
            if cA != cB:
                data[cA, cB][language.name] += [(fA, fB)]
```

We can now load the data into wordlists and also start right away to compute our colexifications.

```
wl = Wordlist([
    Dataset.from_metadata("cldf-datasets/ids/cldf/cldf-metadata.json"),
    Dataset.from_metadata("cldf-datasets/yuchinese/cldf/cldf-metadata.json")
    ])

cols = defaultdict(lambda : defaultdict(list))
all_languages = []
all_words = {}
for language in wl.languages:
    if language.family and language.glottocode and language.latitude:
        print("[i] analyzing language {0}".format(language.name))
        all_languages += [language]
        get_colexifications(language, cols)
        for form in language.forms:
            all_words[form.id] = [form.form, form.form]
```

From here, we create a networkx-Graph object from the inferred colexifications.

```
# get all concepts involved in colexifications
G = nx.Graph()
for (nA, nB), data in cols.items():
    if nA != nB:
        if nA not in G.nodes:
            G.add_node(
                nA, words=0, languages=0, families=0, frequency=1,
                concept=nA, ID=nA)
        if nB not in G.nodes:
            G.add_node(
                nB, frequency=1, concept=nB, ID=nB, words=0,
                languages=0, families=0)
        languages = list(data)
```

```
    words, wofam, families = [], [], []
    for lng, forms in data.items():
        words += [forms[0][0].id]
        wofam += ["/".join([
            forms[0][0].id,
            forms[0][1].id,
            forms[0][0].form,
            forms[0][0].language.id,
            forms[0][0].language.family])]
        families += [forms[0][0].language.family]
    G.add_edge(
            nA,
            nB,
            languages=";".join(languages),
            families=";".join(families),
            weight=len(set(families)),
            words="/".join(words),
            wofam=";".join(wofam)
            )
```

From this graph, we can now computed clusters with the help of the Infomap algorithm (Rosvall and Bergstrom 2008) and directly write them to JSON.

```
IG = networkx2igraph(G)
concepts = {}
for comm in IG.community_infomap(edge_weights="weight"):
nodes = [IG.vs[n]["Name"] for n in comm]
if len(nodes) > 3:
mynode = slug(nodes[0])
subgraph = nx.subgraph(G, nodes)
print('Writing Graph for {0}'.format(mynode))
graph2json(subgraph, "app/cluster/"+mynode)
for node in nodes:
concepts[node] = mynode
```

Having done so, we now only need to create the GeoJSON file of the languages.

```
# write language geojson file
features = []
for language in all_languages:
    if language.latitude:
        geom = {
```

```
            "type": "Point",
            "coordinates": [
              float(language.longitude),
              float(language.latitude)
              ]
            }
        prop = {
            "language": language.id,
            "lat": float(language.latitude),
            "lon": float(language.longitude),
            "name": language.name,
            "glottocode": language.glottocode,
            "key": language.id,
            "marker-color": "#00ff00",
            "coverage": len(language.forms),
            "family": language.family,
            "source": language.dataset
            }
        features += [{
          "geometry": geom,
          "properties": prop,
          "type": "Feature"
          }]
```

And finally write this file along with the remaining “user” files to the user folder in the app directory.

```
with open("app/user/langsGeo.json", "w") as f:
  json.dump({"features": features}, f, indent=2)
with open("app/user/words.json", "w") as f:
  json.dump(all_words, f, indent=2)
with open("app/user/infomap-names.js", "w") as f:
  f.write("var INFO = "+json.dumps(concepts, indent=2)+";\n")
```

4 Exploring the Data

That’s all that needs to be done. When using the application, you may either need to create a local server to browse the data or make sure that you disable the specific security precautions of webbrowsers like Firefox and GoogleChrome. These prevent the loading of files that are stored on your local computer. If you do not know how to disable the security on your web browser, I recommend to just use Python to create a local server in order to browse the files.

This can again be done from the command line by typing:

```
$ cd app
$ python http.server 8000
```

In order to access the application, you only need to open the address http://localhost:8000 in your favorite webbrowser in order to inspect the colexification networks interactively. The following screenshot gives an example of the network that contains the concept FOOTPRINT in the data.

5 Conclusion

This is the final post on my miniseries discussing how colexifications can be computed, how they can be modeled in the form of colexification networks, and how these colexification networks can then be visualized. I hope that this information will prove useful for those who are either interested in investigating the rather large collection of CLDF datasets which we have been assembling during the past years or in creating their own CLDF datasets in order to explore them with the help of interactive colexification networks.

The code and data that you need to carry out the analyses described here are also available in the form of a GitHub repository, accessible at https://github.com/cldf/clicsviz.

References

Bostock, M. and Ogievetsky, V. and Heer, J. (2011): D3: Data-Driven Documents. IEEE Transactions on Visualization \& Computer Graphics (Proc. InfoVis) 17.12. 2301-2309.

List, Johann-Mattis, Thomas Mayer, Anselm Terhalle, and Matthias Urban (2014). CLICS: Database of Cross-Linguistic Colexifications. Marburg: Forschungszentrum Deutscher Sprachatlas (Version 1.0, online available at http://CLICS.lingpy.org, accessed on 2022-8-22).

List, Johann-Mattis and Forkel, Robert (2021): CL Toolkit. A Python Library for the Processing of Cross-Linguistic Data [Software Library, Version 0.1.1]. Geneva:Zenodo. https://pypi.org/project/cltoolkit

Key, Mary Ritchie and Comrie, Bernard (2016): The Intercontinental Dictionary Series. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://ids.clld.org

List, Johann-Mattis and Greenhill, Simon J. and Anderson, Cormac and Mayer, Thomas and Tresoldi, Tiago and Forkel, Robert (2018): CLICS^2. An improved database of cross-linguistic colexifications assembling lexical data with help of cross-linguistic data formats. Linguistic Typology 22.2. 277-306.

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. 1-6. https://calc.hypotheses.org/4266

List, Johann-Mattis (2022): How to Compute Colexification Networks with CL Toolkit (How to do X in Linguistics 11). Computer-Assisted Language Comparison in Practice 5.7. 1-8. https://calc.hypotheses.org/4311

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

Mayer, Thomas and List, Johann-Mattis and Terhalle, Anselm and Urban, Matthias (2014): An interactive visualization of cross-linguistic colexification patterns. In: : Visualization as added value in the development, use and evaluation of Linguistic Resources. Workshop organized as part of the International Conference on Language Resources and Evaluation. 1-8.

Rosvall, M. and Bergstrom, C. T. (2008): Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences 105.4. 1118-1123.

Rzymski, Christoph and Tiago Tresoldi and Simon Greenhill and Mei-Shin Wu and Nathanael E. Schweikhard and Maria Koptjevskaja-Tamm and Volker Gast and Timotheus A. Bodt and Abbie Hantgan and Gereon A. Kaiping and Sophie Chang and Yunfan Lai and Natalia Morozova and Heini Arjava and Nataliia Hubler and Ezequiel Koile and Steve Pepper and Mariann Proos and Briana Van Epps and Ingrid Blanco and Carolin Hundt and Sergei Monakhov and Kristina Pianykh and Sallona Ramesh and Russell D. Gray and Robert Forkel and List, Johann-Mattis (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