Computer-Assisted Language Comparison in Practice

Tutorials on Computational Approaches to the History and Diversity of Languages

How to Compute Colexification Networks with CL Toolkit (How to do X in Linguistics 11)

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

A colexification network is a network consisting of concepts as nodes with weighted edges drawn between the nodes, indicating how often the concepts colexify across the data in a given sample of languages. Having seen how individual colexifications can be computed with the help of the CL Toolkit package in an earlier blog post, we will now see how this code needs to be extended in order to compute colexification networks.

1 Background

In a previous blogpost, I have illustrated how we can use the CL Toolkit package in order to compute colexifications from individual datasets available in Cross-Linguistic Data Formats. When being asked if I could also illustrate how colexification networks can be visualized, I saw that it would be useful to illustrate this in two more posts, where I first show how colexification networks can be computed (since we only computed colexifications in the previous post) and then show how we can visualize them with the visualization routine written in JavaScript (Mayer et al. 2014) underlying the CLICS database (https://clics.clld.org, Rzymski et al. 2020).

2 Colexification Networks

Colexification networks are a very straightforward way to represent the colexifications in a given cross-linguistic sample. In such a network, the nodes represent the concepts in a multilingual wordlist, and links between the nodes are drawn whenever languages in the sample colexify some of the concepts. The edges can be assigned weights which

Computer-Assisted Language Comparison in Practice
Volume 5, Number 7
https://calc.hypotheses.org/4311/

Published on 18/07/2022

represent how often a colexification is attested in the sample. Once colexification networks have been reconstructed, one can further analyze them with the help of methods for community detection, which help to partition the nodes in the network into groups which have denser connections among themselves than with other nodes from the network (Newman and Girvan 2004).

In my post from last month (List 2022), I have shown how the CL Toolkit package (List and Forkel 2021, https://pypi.org/project/cltoolkit) can be used to aggregate several datasets coded in the Cross-Linguistic Data Formats curated with the workflow underlying the Lexibank project (List et al. 2022). With little modifications to the original code, we can represent and store the data as a network and partition the nodes into communities.

3 Preliminary Requirements

Apart from the preliminary requirements mentioned in our previousu post (List 2022), two Python packages for the processing and analysis of graphs are required in order to run the code. Networkx (https://networkx.org, Hagberg 2009) is a very popular package for network operations in Python. Installing it with pip usually does not pose too many problems. The second package, iraph (https://igraph.org, Csardi and Nepusz 2006) is a very powerful collection of tools for network analysis written in C with interfaces to Python and R. Installation with pip may work by now, but in the past, it was important to install the C library first. The igraph website provides all information needed for installation.

As dataset for the computation of our colexification network, we will use the Intercontinental Dictionary Series (https://ids.clld.org, Key and Comrie 2016), which is available in Cross-Linguistic Data Formats (Forkel et al. 2018) and can therefore be directly loaded with the help of CL Toolkit.

In order to visualize the graph, I usually use Cytoscape (https://cytoscape.org, Smoot et al. 2011). We export the graph to the GML format, which can be easily imported into Cytoscape. There are, of course, other tools for graph visualization which can be used as well. In a follow-up post, I will furthermore illustrate how to visualize our graph in JavaScript, using the original visualization code presented in Mayer et al. (2014), employed by the CLICS database (https://clics.clld.org, Rzymski et al. 2020).

4 Computing Colexification Networks

We start by importing the relevant libraries. From the Python standard library we use the defaultdict class and the combinations function, as well as the html package, which we need to write the network to file. Apart from CL Toolkit and pycldf, which we need to

load the data from CLDF, we import networkx, and from LingPy (https://lingpy.org, List and Forkel 2022), we take a function that converts graphs from networkx to igraph, allowing us to make use of the advanced community detection methods provided by the igraph package.

```
from collections import defaultdict
from itertools import combinations
import html

from cltoolkit import Wordlist
from pycldf import Dataset
import networkx as nx
from lingpy.convert.graph import networkx2igraph
```

Next, we define our function to retrieve colexifications for an individual language and store them in a dictionary. Here, we slightly modify the function we used last time, this time storing the Form objects from CL Toolkit, which allow us to retrieve essential information on the origin of these forms later when creating the network.

```
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 wordlist, in our example, we only load the data from the Intercontinental Dictionary Series, which we assume is stored in a folder with the name ids.

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

In order to assemble the colexifications, we now iterate over the languages in our wordlist.

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

We can now reconstruct the graph from the colexifications which we have assembled in the dictionary cols. In order to do so, we first define the graph and then iterate over the entries in our dictionary. We want to know how frequently each concept occurs in our data and store this as a node attribute of each node (corresponding to a concept) in the graph. To calculate the weights, we take the number of language families in which a colexification is attested, but we could alternatively also take the number of languages or the number of words.

```
G = nx.Graph()
for (nA, nB), data in cols.items():
    if nA not in G.nodes:
        G.add_node(
                nA,
                concept=nA,
                frequency=1,
                )
    else:
        G.nodes[nA]["frequency"] += 1
    if nB not in G.nodes:
        G.add_node(
                nB,
                concept=nB,
                frequency=1,
                )
    else:
        G.nodes[nB]["frequency"] += 1

    languages = list(data)
    words, families = [], []
    for item in data.values():
        words += [item[0][0].id]
        families += [item[0][0].language.family]
```

```
  G.add_edge(
        nA,
        nB,
        languages=";".join(languages),
        families=";".join(families),
        weight=len(set(families)),
        words="/".join(words),
        )
```

Having constructed the graph, we can now compute communities from it, using the Infomap algorithm (Rosvall and Bergstrom 2008), provided by the igraph package. This yields a bunch of nodes assigned to the same community as output, which we directly add to our networkx graph as a node attribute.

```
IG = networkx2igraph(G)
for i, comm in enumerate(
      IG.community_infomap(
         vertex_weights="frequency",
         edge_weights="weight")):
   for node in comm:
      G.nodes[IG.vs[node]["Name"]]["infomap"] = str(i+1)
```

All individual nodes in our graph now have an “infomap” attribute which allows us to assign them to one distinct community.

All that is needed to do now is to write the graph to file. Here, the the networkx package will throw an error when writing data to file that contains non-ASCII characters. To avoid this, networkx converts Unicode characters to HTML entities. To reverse this, we use a workaround and generate the GML line by line, converting all HTML entities back to Unicode with the help of the unescape function of the html package of the Python standard library.

```
with open("graph.gml", "w") as f:
   for line in nx.generate_gml(G):
      f.write(html.unescape(line)+"\n")
```

Having created the graph stored in the file graph.gml, it is now easy to open Cytoscape and import the file. When loading the network the first time, Cytoscape won’t offer any layout. Selecting the “group-attributes”-layout will yield a large graph with all inferred communities and connections inferred across them, as shown in Figure 1.

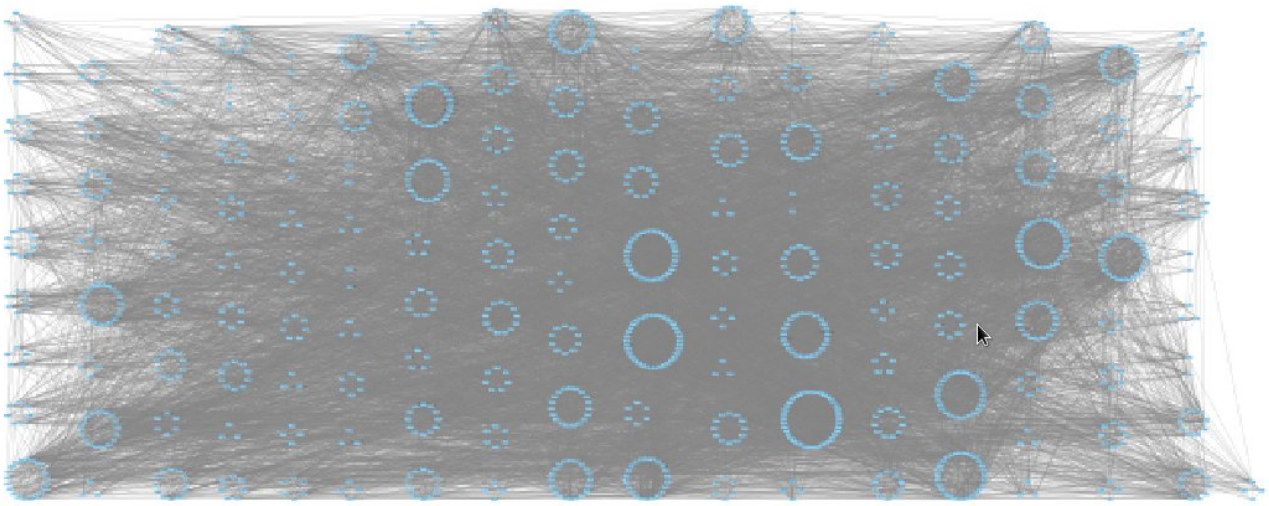

Figure 1: Full graph using Cytoscape's "group-attributes" layout.

Zooming into any of the communities will offer very interesting patterns. The following community, for example, shows concepts related to "time", with "sun" as a central concept that often colexifies with "day".

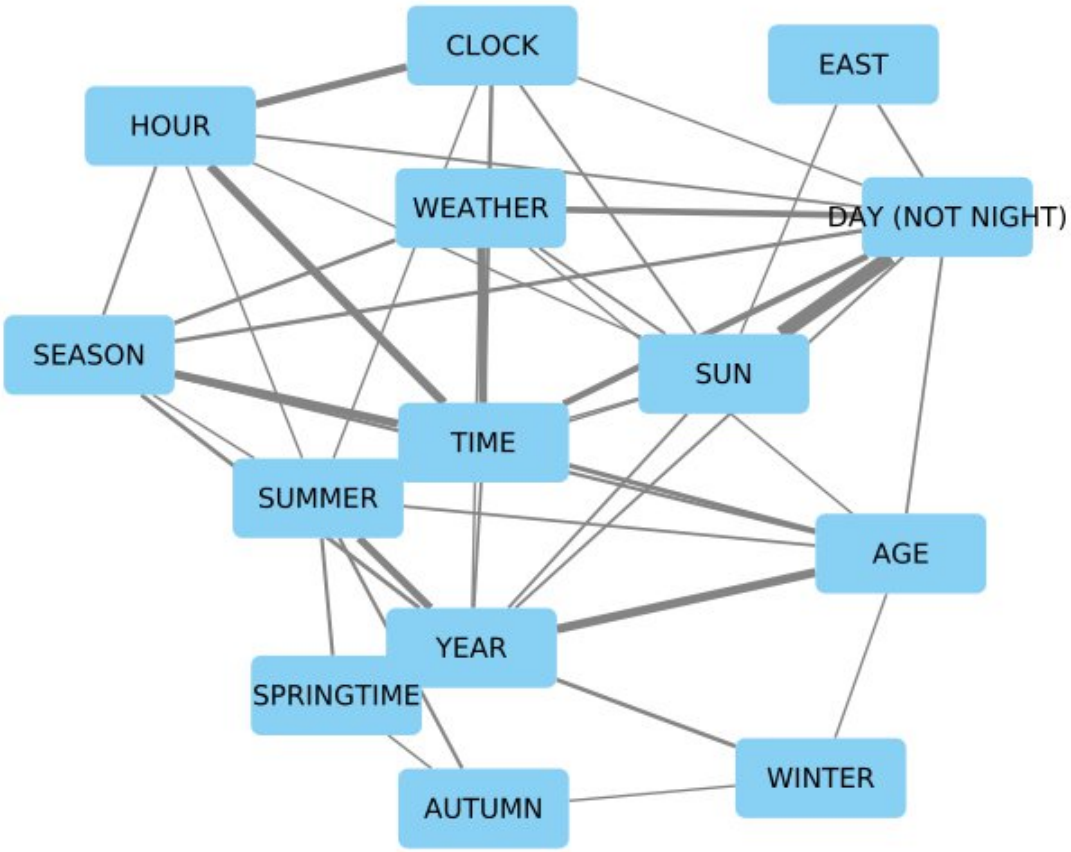

Figure 2: Community clustering concepts around "time"

5 Outlook

Having seen how we can create and explore colexification networks with the help of CL Toolkit, networkx, igraph, and visualize them with Cytoscape, I will continue this little series of blog posts next month by showing how we can visualize the data interactively in JavaScript, using the code we developed for the original CLICS database. The code discussed in this post is available in the form of a GitHub GIST that can be found at https://gist.github.com/LinguList/35adb68afa1500bfbdecdad1630307bd.

References

Gábor Csárdi and Tamás Nepusz (2006): The igraph software package for complex network research. InterJournal. Complex Systems .1695. https://igraph.org

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. https://doi.org/10.1038/sdata.2018.205

Hagberg, Aric (2009): NetworkX. High productivity software for complex networks. http://networkx.lanl.gov/index.html.

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

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]. Leipzig: Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/cltoolkit

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. https://lingpy.org

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 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://digling.org/calc/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.

Newman, M. E. J. (2004): Analysis of weighted networks. Physical Review E 70.5. 056131.

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

Smoot, Michael E. and Keiichiro Ono and Johannes Ruscheinski and Peng-Liang Wang and Trey Ideker (2011): Cytoscape 2.8: new features for data integration and network visualization. Bioinformatics 27.3. 431-432. https://cytoscape.org