Tutorial on techniques for automated borrowing detection (supplement for the the paper «Automated methods for the investigation of language contact situations, with a focus on lexical borrowing»)

Johann-Mattis List

Abstract

While language contact has so far been predominantly studied on the basis of detailed case studies, the emergence of methods for phylogenetic reconstruction and automated word comparison – as a result of the recent quantitative turn in historical linguistics – has also resulted in new proposals to study language contact situations by means of automated approaches. This study provides a concise introduction to the most important approaches which have been proposed in the past, presenting methods that use (A) phylogenetic networks to detect reticulation events during language history, (B) sequence comparison methods in order to identify borrowings in multilingual datasets, and (C) arguments for the borrowability of shared traits to decide if traits have been borrowed or inherited. While the overview focuses on approaches dealing with lexical borrowing, questions of general contact inference will also be discussed where applicable.

Introduction

This very short tutorial will run the readers through a small number of examples showing how automated borrowing detection methods can be applied to a small dataset. Readers can either follow the instructions on

Installation requirements

You will need LingPy (List et al. 2018, https://github.com/lingpy/lingpy), in its most recent version (2.6.4). You can install LingPy with the package manager `pip` LingPy by typing:

```
$ pip install lingpy
```

Preparing the data

Before we start preparing the data, we need to download it. You find the IELex dataset (Dunn 2012), which we are going to use for download on this link (or see https://github.com/sequencecomparison/SupplementaryMaterial, where you find the file in the folder `benchmark/cognates`).

If you have downloaded the file `IEL.csv`, place it in the same folder in which you start your terminal.

We start by importing all relevant libraries.

```
from lingpy import *
from itertools import combinations
from collections import defaultdict
from lingpy.compare.phylogeny import PhyBo
```

Now, we load the file, and determine the languages we want to use for the study.

```
lex = LexStat('IEL.csv')
languages = ['English', 'German', 'French', 'Spanish', 'Dutch_List',
        'Italian', 'Breton_ST']
lex.output('tsv', filename='iel-subs',
        subset=True,
        rows=dict(doculect='in '+str(languages)))
```

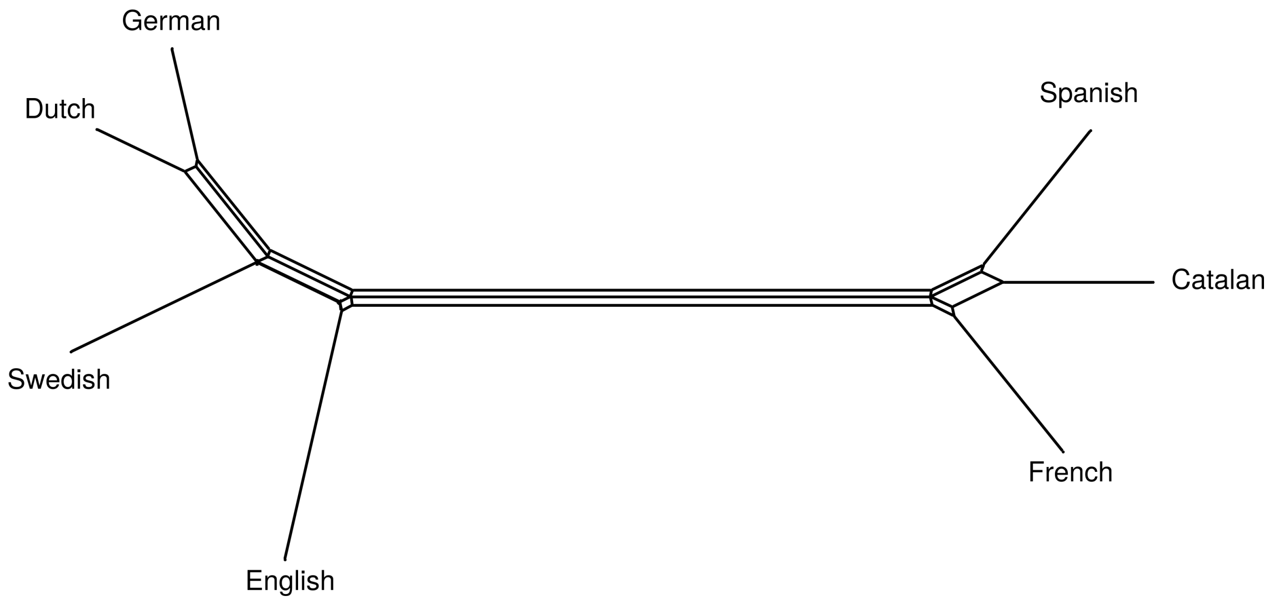


Figure 1: SplitsTree of the data

Computing distances to create data for SplitsTree

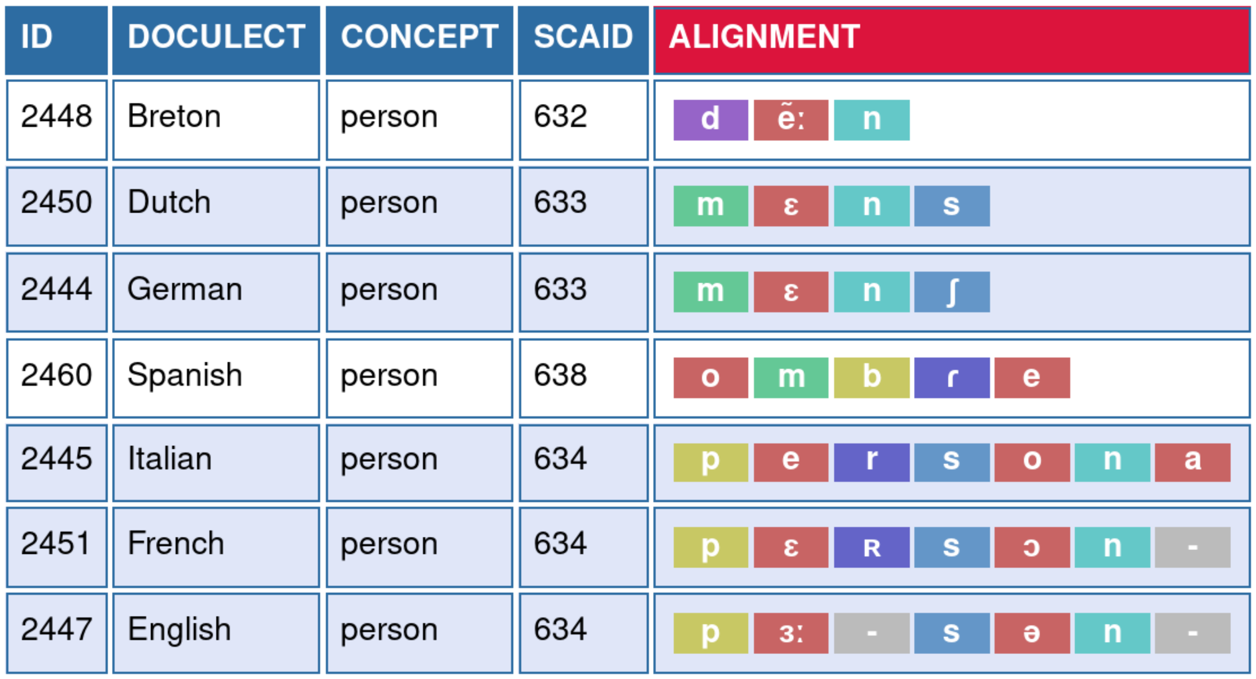
Now we can calculate distances, to be shown in the SplitsTree software package.

```
lex = LexStat('iel-subs.tsv')
lex.distances = lex.get_distances(method='sca', aggregate=True)
lex.output('dst', filename='distances')
```

The resulting file `distances.dst` looks as follows:

```
 7
Breton_ST   0.0000  0.7772  0.7629  0.6963  0.7602  0.6806  0.7026
Dutch_List  0.7772  0.0000  0.3600  0.7107  0.1485  0.6954  0.7300
English 0.7629  0.3600  0.0000  0.6735  0.3960  0.6837  0.7035
French  0.6963  0.7107  0.6735  0.0000  0.7136  0.1970  0.2850
German  0.7602  0.1485  0.3960  0.7136  0.0000  0.6985  0.7277
Italian 0.6806  0.6954  0.6837  0.1970  0.6985  0.0000  0.2600
Spanish 0.7026  0.7300  0.7035  0.2850  0.7277  0.2600  0.0000
# # Created using the LexStat class of LingPy-2.6.4
```

When removing the last line, it directly be copy-pasted to the data-panel in SplitsTree, or loaded as a proper file. The resulting networks is shown in Figure 1.



ID	DOCULECT	CONCEPT	SCAID	ALIGNMENT
2448	Breton	person	632	d ẽː n
2450	Dutch	person	633	m ɛ n s
2444	German	person	633	m ɛ n ʃ
2460	Spanish	person	638	o m b ɾ e
2445	Italian	person	634	p e r s o n a
2451	French	person	634	p ɛ ʀ s ɔ n -
2447	English	person	634	p ɜː - s ə n -

Figure 2: Alignment in EDICTOR software for "person".

Determine cognates and align the data

By identifying automatic cognates, we can also find sequences which are "too similar" to be cognate, like the example for English *person*.

```
lex.cluster(method='sca', threshold=0.45, ref='scaid')
alms = Alignments(lex, ref='scaid')
alms.align()
alms.output('tsv', filename='aligned')
```

To open and search the file, we recommend to use the EDICTOR tool (List 2017, http://edictor.digling.org), which provides an easy access to inspect the alignments, as shown in Figure 1.

Computing minimal lateral networks

To compute minimal lateral networks of the data, we can also use the LingPy software. We export one plot of the data, showing the inferred evolution of the words for 'person' in the data.

```
phy = PhyBo('aligned.tsv', ref='cogid', tree_calc='upgma')
phy.analyze()
phy.plot_concept_evolution('w-2-1', 'person', radius=0.8, outer_radius=0.1,
        proto='alignment', fileformat='pdf')
```

The resulting plot is show in Figure 2. As you can see from this plot, the automatically inferred evolutionary scenario for the words for "person" in the set of languages points to an independent origin of the word *person* in English. If one finds such a situation, this can be interpreted in such a way that either of the cognate sets which originate independently have been borrowed, which is – in our case – of course true of English *person*, being a borrowing from Romance.

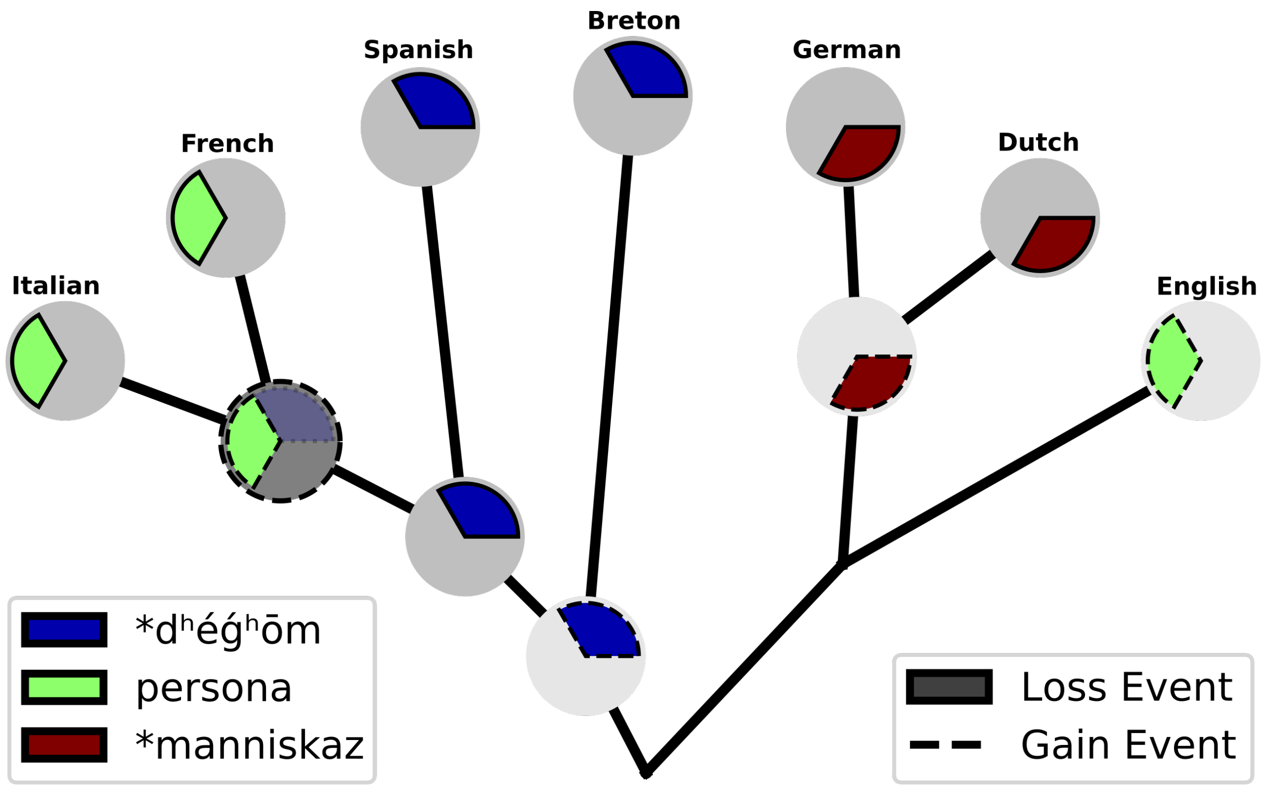


Figure 3: Minimal lateral network for “person”.

References

- Dunn, Micheal (2012): Indo-European Lexical Cognacy Database. Nimegen: Max Planck Institute for Psycholinguistics.
- List, Johann-Mattis, Simon Greenhill, Tiago Tresoldi, and Robert Forkel. 2018. "LingPy. A Python Library for Quantitative Tasks in Historical Linguistics." Jena: Max Planck Institute for the Science of Human History. 2018. http: //lingpy.org.
- List, J.-M. (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. 9-12.