A New Python Library for the Manipulation and Annotation of Linguistic Sequences

Robert Forkel[1] and Johann-Mattis List[12]
[1] DLCE. Max Planck Institute for Evolutionary Anthropology, Leipzig
[2] Chair of Multilingual Computational Linguistics, University of Passau, Passau

The Python package linse (https://pypi.org/project/linse) offers various methods for the manipulation and annotation of sequences. In this short overview, we summarize its major functionalities and provide some information on its background and how we intend to develop it further in the future.

1 Introduction

Many tasks that need to be routinely carried out in computational linguistics in general and computational historical linguistics in specific deal with the manipulation and annotation of linguistic sequences. Since many structures in linguistics can be modeled as sequences, there are quite a few computational tasks of relevance, but few libraries are available that would offer reference implementations of major tasks. With the linse package (Forkel and List 2024, Version 0.1, https://pypi.org/project/linse), that was published a couple of weeks ago, we are trying to make a first step towards such a library.

2 Background

The linse package grew out of the desire to make it easier to maintain the large amount of different modules in the LingPy library (List and Forkel 202a3, Version 2.6.13, https://pypi.org/project/lingpy). Over the last years, LingPy has assembled several methods for sequence manipulation and annotation that were in part poorly tested and had become difficult to maintain. As part of a general plan to try to cut down LingPy's functionality to a core, focusing on sequence comparison and automated cognate detection, we had already stopped to develop LingPy's methods further. Our plan was to successively cut out useful methods and routines from LingPy's codebase in order to put

Computer-Assisted Language Comparison in Practice
Volume 7, Number 1
URL: https://calc.hypotheses.org/6959 DOI: 10.15475/calcip.2024.1.3
Published under a Creative Commons Attributions 4.0 LICENSE
Published on 25/03/2024

them in dedicated libraries, using newly established libraries also for the publication of newly developed methods. The major goals behind this strategy were to increase test coverage, to reduce dependencies for smaller packages, and to offer the possibility to use functionality that was so far only available in LingPy without having to install the full package.

The strategy to redistribute certain parts of LingPy's methods in dedicated smaller packages with fewer dependencies is reflected in the PyloCluster package (List and Forkel 2021, https://pypi.org/project/pylocluster), offering LingPy's basic implementations of the UPGMA (Sokal and Michener 1958) and the Neighbor-joining algorithms (Saitou and Nei 1987) for distance-based phylogenetic reconstruction. The strategy to publish new methods in a new package with a clearer structure and goal is reflected in the LingRex package (List and Forkel 2023b, Version 1.4.1, https://pypi.org/project/lingrex), which offers many methods that have been developed recently (compare, for example, List et al. 2023) and often includes original test cases from the papers in which the new methods were introduced.

While PyloCluster allows users to make use of the UPGMA and the Neighbor-joining algorithm without having to install LingPy with its long tail of dependencies, we originally wanted linse to provide lightweight access to the major routines for sequence manipulation and annotation that were hidden inside LingPy. When starting to work on the package, however, we realized that there were additional sequence manipulation methods that we had started to use in the context of different projects (such as Lexibank, List et al. 2022 and CLTS, List et al. 2021, Version 2.1.0, https://clts.clld.org), and we decided to include these as well, to provide a package dedicated to sequences in cross-linguistic approaches to historical and typological language comparison.

3 Basic Structure of the Package

In its current form, linse assembles methods and functions for the manipulation and annotation os linguistic sequences across four major modules. The segment module offers methods to segment a raw string into a sequence (including methods to tokenize strings transcribed in IPA or to convert strings in SAMPA to IPA).

The subsequence module offers basic routines to compute all possible prefixes, suffixes, affixes, and substrings from a sequence (these routines are important for the computation of partial colexifications as described in List 2023).

The annotate module offers basic methods for the annotation of sequences, which we understand as any method that takes a sequence as input and returns a sequence of the same length, containing annotations for each segment of the original sequence. Here, users find methods to convert segmented phonetic transcriptions into various sound class systems (including the system by Dolgopolsky 1964 and the ASJP code by Holman et al.

2011), including now also direct access to the standardized version of the IPA proposed by the CLTS project (see Anderson et al. 2018 and Anderson et al. 2023).

The transform module offers some functions to transform one sequence into another sequence, including a reference implementation for the idea to manipulate sequences with the help of conversion tables. The idea to manipulate sequences with the help of conversion tables was originally proposed in the context of orthography profiles (Moran and Cysouw 2018), but has now been expanded to allow for any kind of transformation and manipulation of sequences (List 2023), including, for example, the grouping of sounds into evolving units (List et al. 2024).

Additional modules of the package provide helper functions and new data types that are useful in the context of sequence manipulation and annotation. The models module provides access to the sound class models in LingPy (originally proposed in List 2014), the profile module provides routines that can be used to create an initial draft orthography profile from a list of strings in phonetic transcription that can later be used to segment and successively convert the data to some standardized version of the IPA and has proven useful in the creation of the Lexibank repository (List et al. 2022). The typedsequence module provides a set of clearly defined and in part nested sequence types for linguistic analysis, starting from a segment, consisting of one and more characters, followed by a morpheme, consisting of one and more segments, and a word, consisting of one and more morphemes, up to a phrase consisting of one and more words. These types are useful and important for the internal representation of linguistic sequences in various methods, including, for example, partial cognate detection (List et al. 2016) or interlinear-glossed text (Forkel 2023).

An important feature of linse is furthermore that it provides direct access to most of the data underlying the CLTS project. This means that users can convert sequences in phonetic transcription to the feature system underlying CLTS without having use the pyclts package (List et al. 2024, Version 3.2.0, https://pypi.org/project/pyclts) in combination with the CLTS data (which turned out to be difficult for some users in the past). Since linse includes the most recent version of CLTS in its static form (for the generation of unseen IPA segments, one must still use pyclts along with the CLTS data), installing linse (which has no additional requirements) offers fast access to the standardized IPA version and the feature system offered by CLTS.

4 Examples

In the following, we will provide a couple of examples that show how the linse library can be used in practice. Since the library itself does not have any dependencies, all you need to run it is a Python installation with Python 3.8 and higher (https://python.org). To install the library, you best use the Python package manager pip.

```
$ pip install linse
```

Alternatively, you can also download the data from the GIT repository hosted on GitHub (https://github.com/lingpy/linse) and install the package from there.
Having installed linse, you can test the four basic modules in interactive Python sessions. Thus, you can first convert a sequence from SAMPA to IPA and then segment it into distinctive sounds.

```
>>> from linse.segment import ipa, sampa2ipa
>>> ipa_string = "".join(sampa2ipa("t_hOxt@R"))
>>> segmented_ipa = ipa(ipa_string)
>>> print (ipa_string, ">", " ".join(segmented_ipa))
tʰɔxtəʁ > tʰ ɔ x t ə ʁ
```

If you want to generate all possible substrings of a given input sequence (note that substring is a specific term that refers to consecutive subsequence where no characters inside a sequence have been left out, see [ ]), you can use the substring method from the subsequence module.

```
>>> from linse.subsequence import substrings
>>> substrings("abcd")
['abcd', 'abc', 'bcd', 'ab', 'bc', 'cd', 'a', 'b', 'c', 'd']
```

To convert your segmented IPA sequences to sound classes or to the feature names used in CLTS, you can use the annotate module.

```
>>> from linse.annotate import soundclass, clts
>>> classes = soundclass(segmented_ipa, "asjp")
>>> features = clts(segmented_ipa)
>>> for a, b, c in zip(segmented_ipa, classes, features):
...     print("{0:2}".format(a), ">", b, ">", c)
tʰ > t > aspirated voiceless alveolar stop consonant
ɔ  > o > rounded open-mid back vowel
x  > x > voiceless velar fricative consonant
t  > t > voiceless alveolar stop consonant
ə  > I > unrounded mid central vowel
ʁ  > G > voiced uvular fricative consonant
```

To transform a sequence into another sequence in a very flexible manner using conversion tables, you can use the transform module and the SegmentGrouper class. In order to do so, you should first define a conversion table, which you can then load into the SegmentGrouper class and apply directly to any string representation of a sequence that you want.

```
>>> from linse.transform import SegmentGrouper
>>> table = [["Sequence", "IPA"], ["th", "tʰ"], ["@", "ə"], ["an", "ã"]]
>>> sg = SegmentGrouper.from_table(table)
>>> sg("th@tan", column="IPA")
['tʰ', 'ə', '«t»', 'ã']
```

As you can see from the output, the segment t was not defined in the table. As a result, it is marked specificaly by putting it into the specific quotation marks. While this example looks very much like a typical use case of an orthography profile, the conversion tables in linse can contain any symbol, since we explicitly removed any further semantics that would restrict their use to specific use cases. This means specifically, that the space symbol is just one symbol among many, while space is the major unit indicating segmentation in orthography profiles. This means, one can essentially use conversion tables for the regrouping of previous segmentations.

```
>>> table = [["Sequence", "ReGroup"], ["t h", "t.h"], ["a n", "a.n"], [" ", "NULL"]]
>>> sg = SegmentGrouper.from_table(table)
>>> [segment for segment in sg("t h a n", column="ReGroup") if segment !=
"NULL"]
['t.h', 'a.n']
```

5 Outlook

The linse package is a small software library with a very specific application range. Specifically because it is small and lightweight, however, we hope that it will prove useful for our colleagues who need methods for sequence manipulation in their work. In the future, we hope to test the package further and to expand its application range to include some more basic methods and algorithms that prove useful in our work. At the moment, there are no concrete plans on concrete methods, but we are quite convinced that the version 0.1 in which we have published the linse package by now, won't be the last version of the library.

References

Anderson, Cormac and Tresoldi, Tiago and Chacon, Thiago Costa and Fehn, Anne-Maria and Walworth, Mary and Forkel, Robert and List, Johann-Mattis (2018): A Cross-Linguistic Database of Phonetic Transcription Systems. Yearbook of the Poznań Linguistic Meeting 4.1. 21-53. https://doi.org/10.2478/yplm-2018-0002

Anderson, Cormac and Tresoldi, Tiago and Greenhill, Simon J. and Forkel, Robert and Gray, Russell D. and List, Johann-Mattis (2023): Variation in phoneme inventories: quantifying the problem and improving comparability. Journal of Language Evolution. 1-20. https://doi.org/10.1093/jole/lzad011

Dolgopolsky, Aron B. (1964): Gipoteza drevnejšego rodstva jazykovych semej Severnoj Evrazii s verojatnostej točky zrenija [A probabilistic hypothesis concering the oldest relationships among the language families of Northern Eurasia]. Voprosy Jazykoznanija 2. 53-63.

Forkel, Robert (2023): PyIGT: Handling interlinear glossed text with Python [Software Library, Version 2.1.0]. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/pyigt

Holman, Eric. W. and Brown, Cecil H. and Wichmann, Søren and Muller, André and Velupillai, Viveka and Hammarstrom, Harald and Sauppe, Sebastian and Jung, Hagen and Bakker, Dik and Brown, Pamela and Belyaev, Oleg and Urban, Matthias and Mailhammer, Robert and List, Johann-Mattis and Egorov, Dimitry (2011): Automated dating of the world's language families based on lexical similarity. Current Anthropology 52.6. 841-875.

List, Johann-Mattis (2014): Sequence comparison in historical linguistics. Dusseldorf:Dusseldorf University Press. https://doi.org/10.1515/9783110720082

List, Johann-Mattis (2023): Inference of partial colexifications from multilingual wordlists. Frontiers in Psychology 14.1156540. 1-10. https://doi.org/10.3389/fpsyg.2023.1156540

List, Johann-Mattis (2023): Sequence Manipulation with Orthography Profiles in JavaScript. Computer-Assisted Language Comparison in Practice 6.2. 65–72. https://doi.org/10.15475/calcip.2023.2.3

List, Johann-Mattis and Anderson, Cormac and Tresoldi, Tiago and Forkel, Robert (2021): Cross-Linguistic Transcription Systems [Dataset, Version 2.1.0]. Jena:Max Planck Institute for the Science of Human History. https://clts.clld.org

List, Johann-Mattis and Anderson, Cormac and Tresoldi, Tiago and Forkel, Robert (2024): PyCLTS. A Python library for the handling of phonetic transcription systems [Software Library, Version 3.2.0]. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/pyclts

List, Johann-Mattis and Forkel, Robert (2021): PyloCluster. Basic functionalities for distance-based clustering procedures in Python [Software Library, Version 0.1.0]. Leipzig:Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/pylocluster

List, Johann-Mattis and Forkel, Robert (2023): LingPy. A Python library for quantitative tasks in historical linguistics [Software Library, Version 2.6.13]. Passau: MCL Chair at the University of Passau. https://pypi.org/project/lingpy

Johann-Mattis List and Forkel, Robert (2023): LingRex: Linguistic reconstruction with LingPy [Software Library, Version 1.4.1] . Leipzig: Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/lingrex

Johann-Mattis List and Forkel, Robert (2023): LinSe: Manipulation and annotation of linguistic sequences [Software Library, Version 0.1.0] . Leipzig: Max Planck Institute for Evolutionary Anthropology. https://pypi.org/project/linse

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

List, Johann-Mattis and Hill, Nathan W. and Forkel, Robert and Blum, Frederic (2023): Representing and computing uncertainty in phonological reconstruction. In: Proceedings of the 4th Workshop on Computational Approaches to Historical Language Change. Singapur: 22-32.

List, Johann-Mattis and Hill, Nathan W. and Blum, Frederic and Juárez, Cristian (2024): Grouping sounds into evolving units for the purpose of historical language comparison. Open Research Europe 4.34. 1-8. [Preprint, under review, not peer-reviewed] https://doi.org/10.12688/openreseurope.16839.1

List, Johann-Mattis and Lopez, Philippe and Bapteste, Eric (2016): Using sequence similarity networks to identify partial cognates in multilingual wordlists. In: Proceedings of the Association of Computational Linguistics 2016 (Volume 2: Short Papers). Association of Computational Linguistics 599-605.

Moran, Steven and Cysouw, Michael (2018): The Unicode Cookbook for Linguists: Managing writing systems using orthography profiles. Berlin:Language Science Press.

Saitou, N. and Nei, M. (1987): The neighbor-joining method: A new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 4.4. 406-425.

Sokal, Robert. R. and Michener, Charles. D. (1958): A statistical method for evaluating systematic relationships. University of Kansas Scientific Bulletin 28. 1409-1438.

Supplementary Material
Data and code can be found at https://pypi.org/project/linse

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.