Computer-Assisted Language Comparison with EDICTOR 3

Johann-Mattis List and **Kellen Parker van Dam**
Chair for Multilingual Computational Linguistics
University of Passau
Passau, Germany

Abstract

Computer-assisted approaches to historical and typological language comparison have made great progress over the past two decades. Specifically for the classical tasks of historical language comparison, many computational methods have been published that mimic certain steps of the traditional workflow of the comparative method. In contrast to the diversity of new computational methods, there is only a limited number of interactive tools and interfaces that help scholars to curate and refine their data both before and after the application of computational methods. One of the few publicly available interfaces is EDICTOR (https://edictor.org), an interactive tool for computer-assisted language comparison. EDICTOR has been around for some time, and allows scholars to annotate and align cognate sets in various ways. With EDICTOR 3, the original tool has been enhanced, offering not only new features for data annotation, but also providing the possibility to use purely automatic methods for initial cognate detection, phonetic alignment, and correspondence pattern inference in an integrated workflow.

1 Introduction

The traditional comparative method in historical linguistics relies on a multitude of techniques for historical language comparison that have been established to compare languages systematically in order to shed light on their internal and external history (Ross and Durie, 1996). While having been traditionally carried out manually for more than 200 years (see Atkinson 1875 for an early and detailed description of the method), the last two decades have seen many attempts to provide automatic approaches for various individual steps of the comparative method and beyond (List, forthcoming(a)), reflecting some kind of a *quantitative turn* in historical linguistics (Geisler and List, 2022). Among the automated approaches most directly addressing the individual steps underlying the traditional workflow of the comparative method, we find methods for the detection of cognate words (List, 2012a; Jäger et al., 2017; Dellert, 2018), methods for pairwise and multiple phonetic alignment (Prokić et al., 2009; List, 2012b; Kilani, 2020), and methods for the identification of regular sound correspondence patterns (List, 2019).

While these methods have been shown to work rather well for language families with a shallow time depth (List et al., 2017), with phylogenetic trees inferred from automatically annotated cognate sets showing only minor differences to phylogenetic trees inferred from manually annotated cognate sets (Rama et al., 2018), the black box character by which these automatic methods arrive at their results, along with their failure to find deep etymological relations (Greenhill et al., 2023), has prevented scholars from switching to completely automated workflows. At the same time, however, the manual compilation of etymological datasets, where scholars compare thousands of words across dozens and at times even hundreds of languages, has reached its practical limits.

In a situation where computational methods cannot be used to replace humans and humans cannot cope with increasing amounts of digital data, computer-*assisted* solutions – as opposed to fully computer-*based* or fully manual – may offer an alternative in combining the best of both worlds by uniting the efficiency of computers with the accuracy of human annotation. In 2017, it was tried to put this idea into praxis by proposing a new framework for *Computer-Assisted Language Comparison* (CALC) that would not only try to enhance existing methods for computational historical language comparison, but also seek to develop web-based tools that could serve as an interface between computational and manual approaches, allowing for an interactive workflow in which data – which must be provided in human- and machine-

Proceedings of the 5th Workshop on Computational Approaches to Historical Language Change, pages 1–11

readable form – would be constantly passed back and forth between computers and machines (List, 2017b). Instead of identifying cognate sets from scratch in larger datasets, the idea was to employ automated methods for the pre-processing of linguistic data and then have human experts correct these initial analyses. Given that the correction of pre-processed data would be done in a dedicated web-based tool, it would also be possible to test the consistency of human annotation and offer annotators additional possibilities to explore cross-linguistic data in order to improve their analysis.

With the introduction of the EDICTOR tool (see Version 1.0, https://one.edictor.org, List 2017a), a first step in this direction was carried out. EDICTOR offered a web-based interface to annotated cognates in multilingual wordlists and align them at the same time. Via its simplified data structure (using a single TSV file to represent words, cognates, and alignments in multilingual wordlists), EDICTOR was also integrated with the LingPy library (List and Moran, 2013) that provided access to automatic methods for cognate detection and phonetic alignments. Later enhancements of EDICTOR (see Version 2.0 at https://two.edictor.org, Version 2.1 at https://two-1.edictor.org) have offered more features for annotation, but the basic character of the tool as a purely web-based application that could be used for annotation but not for the conduction of automatic methods has not changed since then.

With EDICTOR 3, not only new features, but also some substantial modifications are introduced to the tool. As of Version 3, EDICTOR will not only be distributed as a web-based tool that can be accessed via its URL (https://edictor.org), but also in the form of a software package written in Python that can be locally installed and does not require internet access to run. The advantage of this new architecture is that EDICTOR can also integrate with external software packages and thus allow for a true exchange with external software packages that provide enhanced methods for basic steps of the comparative method.

2 Background

Although tools and interfaces that would assist linguists in the annotation of etymological data have been around for several decades now, the number of linguists who would make active use of these tools is rather small. One of the first software packages that offered full support for various important tasks in historical language comparison is the STARLING database program, originally designed and created by Sergey Starostin (1953-2005). The origins of the software go back to the early 1990s (Starostin, 2000b). In its core, STARLING is a database system that offers users the possibility to create small databases consisting of multiple tables linked with each other. The software offers dedicated functionality to annotate cognates, to check for the individual sounds in a given wordlist, and to carry out distance-based phylogenetic reconstruction analyses based on the implementation of several ideas proposed by Starostin (Starostin, 2000a).

A second interactive tool for computer-assisted language comparison that is important to mention in this context is RefLex (Segerer and Flavier, 2015). Unlike STARLING, which comes as a software package that has to be installed on the users computers, RefLex is entirely web-based, written mostly in PHP, and accessed by connecting to the server maintained by the RefLex authors. Using RefLex requires a user account, and data must be imported and exported from the internal database. Originally designed to analyze data from African languages, RefLex offers many general functionalities that are very useful for etymological analysis in historical linguistics, including an alignment editor by which cognate sets can be aligned manually, methods to match elicitation glosses for concepts across different sources, and the possibility to annotate cognates in multilingual wordlists.

As impressive and useful as STARLING and RefLex are on their own, both tools have major drawbacks that prompted the development of an alternative interface for computer-assisted language comparison, taking nevertheless a lot of inspiration from the other tools. A major drawback of STARLING is that it does not work well on Unix systems, given that it is based on the now outdated *dBase* database management system that only runs on Windows operation systems. The major disadvantage of RefLex is that it requires a server with users having to log into the system when using it. This means that the tool can only be used with an active internet connection. In addition, import and export options have always been limited in RefLex and it was – for example – never clear how alignments could be exported to text files in order to use them in combination with other software tools.

As a result of these drawbacks, work began already in 2014 to work on my own interactive

tool for computer-assisted language comparison. The goal was to design a tool that would offer functionality similar to those features that were considered most useful in STARLING and RefLex, while at the same time offering a closer integration with automatic methods, most importantly those offered by the LingPy software package for quantitative tasks in historical linguistics (https://lingpy.org, List and Moran 2013).

The first version of this tool (that would later become the core of the CALC framework) was published in 2017 under the title *EDICTOR* (short for *Etymological Dictionary Editor*, List 2017a) and successfully employed to annotate the data underlying a larger phylogenetic analysis of Sino-Tibetan languages (Sagart et al., 2019). The first version of EDICTOR was written in JavaScript and was made accessible in the form of a website that users could access by opening the URL. Since the tool was entirely client-based, users could independently load their files to the JavaScript sandbox and later save them after editing. Data was never sent to any server, but was only edited inside the users browsers on their client systems. EDICTOR offered basic modules to annotate cognate sets (both full and partial cognates, see List et al. 2016), these cognate sets could be aligned with the help of a specific alignment editor, and rudimentary methods were available in order to check sound correspondences for language pairs. With EDICTOR 2 (List, 2021a), this functionality was further expanded by adding methods for the exploration and annotation of *morpheme glosses* (Hill and List, 2017; Schweikhard and List, 2020), extended exploration and export options for cognate sets (including direct export to the NEXUS format used in many phylogenetic applications, see Maddison et al. 1997 and Forkel 2023 for details on the format), and an initial interactive correspondence pattern browser that would display correspondence patterns inferred with the help of the method by List (2019).

EDICTOR has played a crucial role in the further development of computer-assisted techniques on historical language comparison. The tool proved not only important in the creation of reliable datasets (of which quite a few were later included in the Lexibank repository, List et al. 2022). It turned out that the tool was also crucial for the development of new computer-based methods, where it was used to visualize findings in order to check preliminary results and to create high-quality data for testing of new methods for which test and training data were usually lacking. Thus, in the retrospective, it would not have been possible to develop the method for partial cognate detection presented in List et al. (2016) without EDICTOR, since it would not have been possible to create the data that was later used to test the method. Similarly, the algorithm for the inference of sound correspondence patterns presented in List (2019) would not have been possible without the interactive sound correspondence pattern browser, which was crucial for the development of the new method, allowing to inspect findings immediately.

However, with time, EDICTOR also accumulated a considerable number of bugs and strange behaviors. Certain design problems that were not identified as such in the beginning later turned out to be problematic, and users' design suggestions or bug reports could often not be addressed immediately.

There were different reasons for the slow process with respect to the development of the tool. On the one hand, for scientists who develop tools and software packages, there is always a tension between the time they spend on development and the time they spend on proper research, since development is not necessarily seen as truly scientific work. On the other hand, many problems that the tool was supposed to handle turned out to be much harder than expected. As a result, solutions often were not available, and it was instead necessary to enter very detailed discussions on the proper modeling of particular problems before any changes to the tool could be made.

Not all of these problems can be solved with EDICTOR 3, but in contrast to previous releases of EDICTOR, EDICTOR 3 tries to set several new standards for the future development of the tool. As a result, the list of features was for the first time not only expanded, but certain features that had never proven to be useful for historical language comparison, were also discarded.

3 A New EDICTOR Version

3.1 Overview

EDICTOR 3 is a web-based tool that allows its users to carry out several steps of the comparative method interactively, producing data that can be digested by computer programs. EDICTOR 3 comes in two forms. Users can access the tool via its URL at https://edictor.org or download the source code

and create another instance of the tool on a local or public server. Additionally, users can also install a local version of EDICTOR 3 on their own computer and access EDICTOR 3 locally, with no active internet connection being required. Both the public and the local version of EDICTOR 3 basically support the same functionalities, but the local version allows to save and access files stored locally (as pure files or with the help of an SQLite database) without having to go through the upload procedure that passes local data to the JavaScript sandbox. In addition, only the local version allows to obtain high quality cognate judgments, phonetic alignments, and sound correspondence patterns from dedicated Python packages (see § 3.3).

Like previous versions of the EDICTOR, EDICTOR 3 is organized in panels that allow to initiate actions or provide additional views on the data. The core is a multilingual wordlist that stores data in tabular form in a TSV file (see List et al. 2018 for an overview on the basic format). Panels are currently grouped into three basic modules. The *Edit* module offers basic functionality to edit data in various forms (see § 3.2), the *Compute* module offers methods for cognate detection, phonetic alignments, and correspondence pattern inference (see § 3.3), and the *Analyze* module offers additional tools by which the data can be analyzed and inspected (see 3.4).

3.2 Editing Data

In EDICTOR 3, data can be edited in five different ways. The most basic way to edit the data is to use the *Wordlist* panel that allows to edit data in a way similar to a spreadsheet editor but with some additional functionalities that facilitate the annotation of cognates and phonetic transcriptions. New functionality has been added that allows users to group segment data morphologically and to modify the representation of sounds by grouping distinct sounds into evolutionary units (List et al., 2024). The *Cognate Sets* and *Partial Cognate Sets* panels allow to edit cognate sets in a principled way. he *Morpheme Glosses* panel, introduced with EDICTOR 2 (see List 2021b), offers enhanced functionality for the annotation of morphological data with the help of morpheme glosses (Hill and List, 2017; Schweikhard and List, 2020). Finally, the *Correspondence Patterns* panel, which had been introduced earlier, now offers the possibility to edit correspondence patterns *actively* and to identify and mark exceptions in the reflexes of individual cognate sets (List, forthcoming(b)).

3.3 Computing Data

So far EDICTOR has not allowed users to compute data. The only exception was the alignment of individual of cognate sets, where EDICTOR offered the possibility to align words in the interactive window prior to carrying out manual refinements. With EDICTOR 3, basic methods for *cognate detection*, *phonetic alignment* (multiple sequence alignment), and *correspondence pattern detection* are now available as part of the newly introduced *Computing* module of the tool.

For each of the three tasks, two basic solutions are offered to the users. When running with Python internally and having installed the required software packages, the data is passed to Python and the dedicated methods are used to carry out the task. If EDICTOR 3 is accessed via the website, simplified implementations of the three methods in JavaScript are being used.

The basic approach for cognate detection is the LexStat method for full cognates (List, 2012a) or its counterpart for partial cognates (List et al., 2016). Implementations for both methods are available from LingPy (https://pypi.org/project/lingpy, List and Forkel 2023a, Version 2.6.13). The fallback function is based on matching consonant classes, as originally introduced by Dolgopolsky (1964) and then popularly employed in the STARLING package (see Turchin et al. 2010 for a detailed description and List 2014 for details on the implementation). For partial cognates, the approach is adjusted in order to be applied to individual morphemes rather than full words.

The basic approach for phonetic alignments of multiple sound sequences is based on the *Sound-Class based Alignment* method (List, 2012b). The method itself breaks down the complexity in linguistic sequences by converting phonetic transcriptions to sound classes and then conducts traditional multiple sequence alignment analyses using an adjusted version of the T-Coffee algorithm (Notredame et al., 2000). The method is also implemented in LingPy. As a fallback method, EDICTOR 3 employs a very simple and very fast method for multiple alignments that runs in linear time. This method first selects the longest sequence among the candidate sound sequences and then aligns all remaining sequences one by one with this longest sequence. The individual align-

ments are stored and later combined in such a way that all individual gaps introduced in the longest sequence are preserved. Despite the simplicity of the approach, it yields useful results in the majority of cases. When being confronted with complex alignment tasks, it clearly lags behind the SCA approach. However, since the method was created to speed up manual alignments, it is always easier to align sound sequences automatically in a first step and then refine them in a second step manually than starting the alignment manually from scratch.

The basic approach for correspondence pattern detection follows the method proposed in List (2019), which makes use of a greedy approach to solve the *minimum clique cover problem* in undirected networks (Bhasker and Samad, 1991) to group alignment sites (individual columns of a multiple alignment) into clusters from which correspondence patterns can be inferred. This method is implemented in the LingRex package (https://pypi.org/project/lingrex, List and Forkel 2023b, Version 1.4.2). The fallback method offered by EDICTOR 3 is based on a much simpler strategy that uses the sorting method QuickSort (Hoare, 1962) to arrange compatible alignment sites next to each other in a table and then groups those alignment sites that are compatible to each other into correspondence patterns. In contrast to the method by List (2019), this approach does not guarantee to find an exhaustive clique cover of the alignment site network. For the practical purpose of getting a first grouping of alignment sites into correspondence patterns, however, it has turned out to be very useful to speed up the process of manually annotating correspondence patterns.

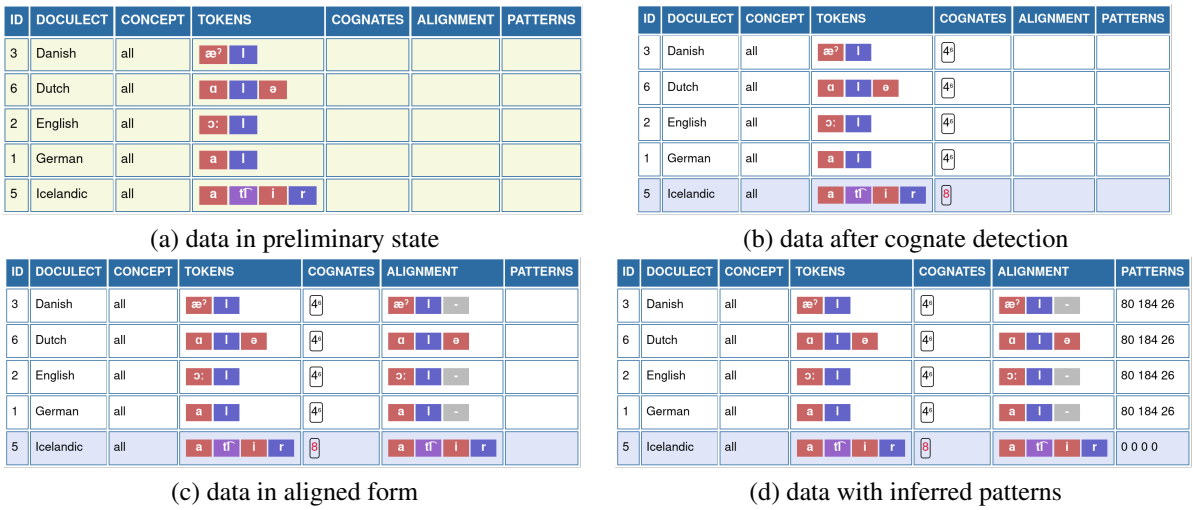
The three methods in combination equip users with a workflow that starts from a raw wordlist in phonetic transcription, then identifies cognates, aligns them, and finally infers correspondence patterns from the data. In this form, the workflow accounts for the majority of the individual steps of the classical comparative method (Ross and Durie, 1996). What it leaves out are methods for phonological reconstruction and phylogenetic reconstruction. Since classical phonological reconstruction, however, builds on previously identified correspondence patterns (Anttila, 1972), EDICTOR 3 offers a very solid basis to build phonological reconstructions on top of explicitly annotated sound correspondence patterns. For phylogenetic reconstruction, the aforementioned option to export data to the NEXUS format comes in handy.

3.4 Analyzing Data

EDICTOR 3 not only supports editing and computing of multilingual wordlist data but also allows to inspect the data in different ways through the *Analyze* module of the tool. With the help of the *Sounds* panel, users can inspect the individual sounds in individual language varieties, by comparing how frequently and in which words they occur and how they fit into classical phoneme inventory tables. The *Colexifications* panel allows for a quick investigation of full and partial colexifications, the former referring to cases of polysemy or homophony, in which a word form expresses two or more concepts in a wordlist (François, 2008), and the latter referring to those cases where morphemes with identical forms recur across different words (List, 2023). The panel offers additional functionality to visualize full and partial colexifications with the help of bipartite networks (Hill and List, 2017). The *Correspondences* panel allows users to compare the sound correspondences inferred from the pairwise alignments of two language varieties. While this functionality may seem much less useful and important, it may prove useful in those cases where one the focus lies on specific relations between two language varieties, such as – for example – in the case of two alternative proposals for phonological reconstruction (Pulini and List, 2024). The *Cognates* panel allows for a detailed inspection o the distribution of cognate sets across a multilingual wordlist, providing a tabular view in which each column is reserved for one language and each row represents one cognate set. Through this specific panel, users can also export their cognate sets to the above-mentioned NEXUS format, which can then be fed to dedicated software packages for phylogenetic reconstruction.

3.5 Implementation

EDICTOR 3 is implemented as a web-based application written in JavaScript. The newly introduced local server functionality that allows users to employ the tool locally is implemented in Python. The code base is curated on GitHub (https://github.com/digling/edictor, and archived as part of the Python Package Index (https://pypi.org/project/edictor). For users who prefer to use the tool without installing it, the most recent version of EDICTOR can be accessed from https://edictor.org, a development version is usually accessible from https://dev.edictor.org, and earlier



(a) data in preliminary state

(b) data after cognate detection

(c) data in aligned form

(d) data with inferred patterns

Figure 1: Integrated computer-assisted workflow in EDICTOR 3. The screenshots represent the different stages by which analyses with LingPy and LingRex applied to the Germanic wordlist data are carried out in the interactive mode.

versions are accessible from https://one.edictor.org (Version 1.0), https://two.edictor.org (Version 2.0), and https://two-1.edictor.org (Version 2.1). Issues in the code as well as discussions about particular features are typically handled via GitHub's issue tracker (https://github.com/digling/edictor/issues).

4 Examples

In the following, we will try to illustrate the new features and ideas that made it into EDICTOR 3 with the help of three examples. These consist of (1) an integrated computer-assisted workflow that can be used to compute cognates, alignment, and correspondence patterns from scratch, (2) an illustration of the new functionalities for the annotation of correspondence patterns, and (3) a discussion of the new approaches that allow to speed up the process of manipulating data provided in phonetic transcription.

4.1 Integrated Computer-Assisted Workflow

To illustrate how an integrated computer-assisted workflow can be carried out with the help of EDICTOR 3, we make use of a small dataset of 110 basic concepts translated into seven Germanic languages. This dataset was originally compiled by Starostin (2005) and later adjusted later adjusted to the format required by EDICTOR and LingPy for testing purposes (List, 2014). The dataset itself can be accessed directly from EDICTOR 3, by opening the landing page (https://edictor.org) and then navigating to the tab *Examples*, where it can be selected under the title `Germanic Wordlist (List 2014)`. Screenshots that illustrate the different stages of the workflow are shown in Figure 1 (a-d).

The analysis itself shown in this example fails to identify the Icelandic wordform as being cognate with the forms in the other languages, which is most likely due to the specific phonetic transcriptions chosen. For computer-assisted purposes, however, the ultimate accuracy of any algorithm is much less important than the general reliability and – as neatly illustrated in this example – the integration with interactive tools that allow scholars to quickly preprocess a given dataset automatically in order to refine the individual findings in a second stage.

4.2 Inspecting and Editing Correspondences

Correspondence patterns inferred by the automatic workflow shown in the previous section can be further edited and modified by the user. Patterns are reflected in the form commonly employed by EDICTOR and LingRex. Patterns are defined with respect to the phonetic alignment. Sites in an alignment are grouped into patterns by assigning them common integers that serve as identifiers and must be greater than zero. The value 0 itself is reserved for those cases in which an alignment site is not assigned to *any* pattern in the data. This holds for cases of singletons (words that are not cognate with any other words in a given dataset) or where the method for correspondence pattern detection cannot find enough evidence to group the data further (e.g. for cognate sets that do not have enough reflexes in the data).

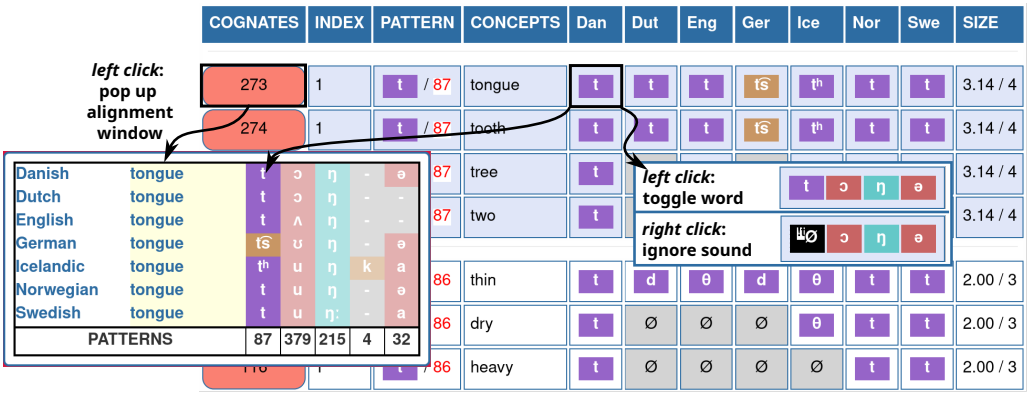


Figure 2: Correspondence patterns inferred by the computational workflow. Additional editing of corresponcence patterns is possible by clicking into the pattern identifiers in the column *PATTERN* and editing values there directly.

Figure 2 shows how correspondence patterns are visualized in EDICTOR 3. Each alignmen site is listed in one row of a table, preceded by the cognate set identifiers, followed by the position of the site in the alignment, followed by the pattern identifier and the concepts reflected by the cognate sets. For each language, the alignment site value is then listed in fixed order. This order itself can be edited by the user in order to put related language varieties together or to put proto-languages in front. Clicking on a particular sound will show the full word, allowing users to toggle between different views, highlighting particular sounds or individual words in which the sounds occur. With the help of a right mouse click, the pattern can be toggled in such a way that the particular sound is ignored when assembling the pattern, using *inline alignments* for the representation (List, forthcoming(b)). This allows users to explicitly ignore certain reflex sounds from correspondence patterns that might show unexpected results. Ultimately, this comes close to a formal version of Grimm's handling of Germanic data, when he noted exceptions from the consonant shift that he had observed (Grimm, 1822). By putting exceptions at the side, one can collect them to try and resolve them later.

As can be seen from the example illustration, the automated workflow has no problem in detecting the classical correspondence of the affricate initials in German corresponding to alveolar stops in the other Germanic languages. This proves again the usefulness of computer-assisted approaches in increasing the efficiency of linguistic annotation.

4.3 Segmenting and Grouping

The wordlist panel in EDICTOR 3 comes with a new feature that allows for an improved editing of sound sequences. Already in the first version, it was possible to insert phonetic transcriptions in SAMPA / X-Sampa (see Gibbon et al. 1997, 60-108 for a specification of SAMPA), which would then automatically be converted to the International Phonetic Alphabet in plain Unicode (IPA, 1999) and automatically *segmented* into individual sounds, following the standards proposed by the Cross-Linguistic Data Formats initiative for the handling of phonetic transcriptions (Forkel et al., 2018). In EDICTOR 3, these editing functionalities were streamlined and extended by adding additional possibilities to segment words into morphemes and to group individual sounds into evolving units.

The extended sequence editing features are illustrated in Figure 3, where some German words are provided as sample sequences along with their morpheme structure, annotated with the help of morpheme glosses. While the conversion from input in SAMPA/X-SAMPA is automatically triggered when selected by the user (conversion can also be turned of if phonetic transcriptions are provided from the original data or if users prefer to user their own IPA keyboard), the segmentation of individual characters into speech sounds in phonetic transcription is carried out when inserting a sequence with a preceding space. Since trailing spaces are disallowed in the standard format of the column storing sound sequence data in EDICTOR (typically called *TOKENS*), this does not conflict with alternative annotations or other forms of user input. Once sound sequences are inserted into the text fields, EDICTOR automatically colors them, using a color schema that distinguishes 10 different sound classes, as originally proposed by Dolgopolsky (1964). These sequences can then be edited in consecutive steps. First, by right-mouseclicking

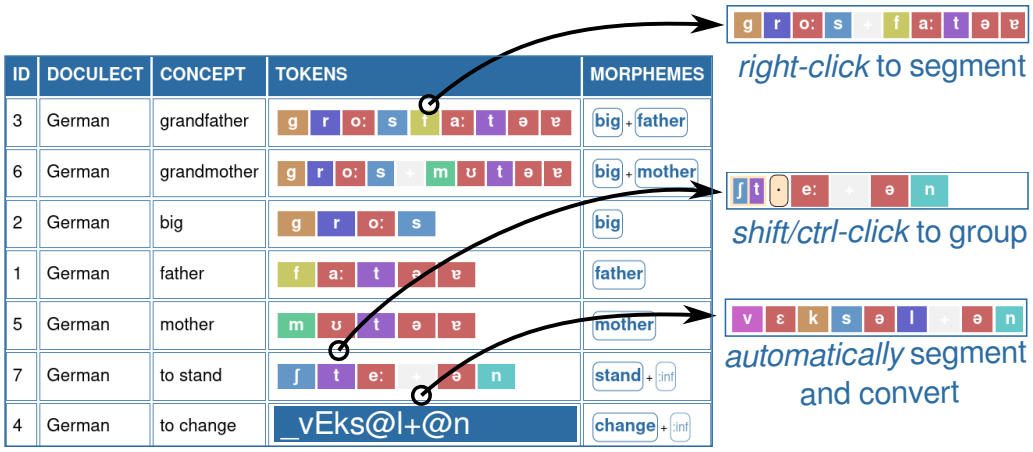


Figure 3: Sequence editing in EDICTOR 3. In addition to the conversion from SAMPA to IPA, EDICTOR 3 also supports the convenient segmentation of sound sequences into morphemes and the grouping of sounds into evolving units.

on individual sound segments, a morpheme boundary marker (the +) is inserted *before* the segment that was clicked, thus allowing users to quickly segment their words into morphemes. Second, by pressing the SHIFT or the CTRL button *and* right-mouse-clicking a segment, it will be grouped with the segment *following* it, using the specific annotation for grouped sounds developed in List et al. (2024).

In combination with additional panels, such as the dedicated panel for the handling of *Morpheme Glosses* that was introduced with an earlier version of EDICTOR, the tool now equips users with multiple possibilities to efficiently annotate language-internal cognates by segmenting words into morphemes and handling co-evolving sounds as single units. Our hope is that these additional methods will soon allow us to create a larger collection of morpheme-segmented wordlists that could later be used to test automatic approaches to the task of morpheme segmentation in computational historical linguistics, for which by now no satisfying solution exists (List, 2024).

5 Outlook

With EDICTOR 3, we hope to enter a new stage of computer-assisted language comparison, by providing a tool that is increasingly robust, allowing for multiple ways of access, and offering sophisticated methods for data annotation and analysis that are more and more fine-grained and adapted to the complex task of etymological analysis in historical linguistics. For the future, we plan not only to improve the integration with existing tools (for example by providing enhanced export functionalities to major phylogenetic software packages), but also to consolidate the current code base. While unit tests for the Python code running the local server application have now been set up, with the tool being tested on all major operation systems, the JavaScript code base was written over a long time frame, containing numerous lines of code that should be refactored. In order to improve the accessibility of the tool further, we also plan to conduct more explicit trainings by offering webinars and by sharing tutorials in video form where we run users through major annotation stages and workflows.

Although not perfect yet, however, we think that EDICTOR 3 already now provides a greatly improved user experience with new functionalities, and we hope that the tool will prove useful for those who want to work with computer-assisted workflows instead of conducting purely quantitative or purely qualitative analyses. The tool is intended to help linguists in their etymological work, not to replace them by switching to exclusively automatic approaches that discard 200 years of scholarship. This general spirit of computer-assisted language comparison has not changed with EDICTOR 3, and we hope that the tool will prove *actually* useful for comparative work in historical linguistics.

Supplementary Material

EDICTOR 3 has been archived with PyPi at https://pypi.org/project/edictor (Version 3.0), is curated on GitHub at https://github.com/digling/edictor, and can be accessed online from https://edictor.org.

Limitations

Computer-assisted approaches to historical language comparison still face many limitations that cannot be overcome by one single tool. The majority of the limitations we face in building tools that assist linguists conducting computer-assisted as opposed to purely classical studies consist in the modeling of etymological relations between words (both when comparing words inside one and the same language and across multiple languages). Regarding EDICTOR 3, three very urgent limitations can be found in the lack of a principled handling of complex paradigms in multilingual wordlists (1), the limitation of the models used to handle partial cognates to account for non-concatenative morphology (2), and the absence of general procedures to check or annotate conditioning context that would explain multiple sound reflexes in individual languages for the same proto sound (3). We do not have any concrete ideas to solve any of these three problems at the moment, but we discuss them often and hope to be able to improve our work on these open problems at some point in the future.

Acknowledgments

This project was supported by the ERC Consolidator Grant ProduSemy (PI Johann-Mattis List, Grant No. 101044282, see https://doi.org/10.3030/101044282). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency (nor any other funding agencies involved). Neither the European Union nor the granting authority can be held responsible for them.

We thank all those who have been supporting the development of EDICTOR in the past by testing the tool, suggesting modifications, and discussing new features and annotations. Special thanks in this context go to Carlos Barrientos, Frederic Blum, Nicolás Brid, Fabrício Gerardi, Abbie Hantgan, Nathan W. Hill, Guillaume Jacques, John Miller, Laurent Sagart, and Roberto Zariquiey. We also thank the doctoral students in the ProduSemy project – Katja Bocklage, Alžbęta Kuçerová, Arne Rubehn, and David Snee – for testing and discussing development versions of EDICTOR 3.

References

Raimo Anttila. 1972. *An introduction to historical and comparative linguistics*. Macmillan, New York.

Robert Atkinson. 1875. Comparative grammar of the Dravidian languages. *Hermathena*, 2(3):60–106.

J. Bhasker and Tariq Samad. 1991. The clique-partitioning problem. *Computers & Mathematics with Applications*, 22(6):1–11.

Johannes Dellert. 2018. Combining information-weighted sequence alignment and sound correspondence models for improved cognate detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3123–3133.

Aron B. Dolgopolsky. 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.

Robert Forkel. 2023. *CommonNexus. A nexus (phylogenetics) file reader and writer [Software, Version 1.9.1]*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Robert Forkel, 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.

Alexandre François. 2008. Semantic maps and the typology of colexification: intertwining polysemous networks across languages. In Martine Vanhove, editor, *From polysemy to semantic change*, pages 163–215. Benjamins, Amsterdam.

Hans J. Geisler and Johann-Mattis List. 2022. Of word families and language trees: New and old metaphors in studies on language history. *Moderna*, 24(1-2):134–148.

Dafydd Gibbon, Roger Moore, and Richard Winski, editors. 1997. *Spoken Language Reference Materials*. De Gruyter Mouton, Berlin and Boston.

Simon J. Greenhill, Hannah J. Haynie, Robert M. Ross, Angela Chira, Johann-Mattis List, Lyle Campbell, Carlos A. Botero, and Russell D. Gray. 2023. A recent northern origin for the uto-aztecan family. *Language*, 0(0).

Jacob Grimm. 1822. *Deutsche Grammatik*, 2 edition, volume 1. Dieterichsche Buchhandlung, Göttingen.

Nathan W. Hill and Johann-Mattis List. 2017. Challenges of annotation and analysis in computer-assisted language comparison: A case study on Burmish languages. *Yearbook of the Poznań Linguistic Meeting*, 3(1):47–76.

Charles A. R. Hoare. 1962. Quicksort. *The Computer Journal*, 5(1):10–16.

IPA. 1999. *Handbook of the International Phonetic Association*. Cambridge University Press, Cambridge.

Gerhard Jäger, Johann-Mattis List, and Pavel Sofroniev. 2017. Using support vector machines and state-of-the-art algorithms for phonetic alignment to identify cognates in multi-lingual wordlists. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Long Papers*, pages 1204–1215, Valencia. Association for Computational Linguistics.

Marwan Kilani. 2020. FAAL: a feature-based aligning ALgorithm. *Language Dynamics and Change*, 11(1):30–76.

Johann-Mattis List. 2012a. LexStat. Automatic detection of cognates in multilingual wordlists. In *Proceedings of the EACL 2012 Joint Workshop of Visualization of Linguistic Patterns and Uncovering Language History from Multilingual Resources*, pages 117–125, Stroudsburg.

Johann-Mattis List. 2012b. SCA: Phonetic alignment based on sound classes. In Marija Slavkovik and Dan Lassiter, editors, *New directions in logic, language, and computation*, pages 32–51. Springer, Berlin and Heidelberg.

Johann-Mattis List. 2014. *Sequence comparison in historical linguistics*. Düsseldorf University Press, Düsseldorf.

Johann-Mattis List. 2017a. 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*, pages 9–12, Valencia. Association for Computational Linguistics.

Johann-Mattis List. 2017b. *Computer-Assisted Language Comparison. Reconciling computational and classical approaches in historical linguistics [Research Project, 2017–2022]*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Johann-Mattis List. 2019. Automatic inference of sound correspondence patterns across multiple languages. *Computational Linguistics*, 45(1):137–161.

Johann-Mattis List. 2021a. *EDICTOR. A web-based tool for creating, editing, and publishing etymological datasets*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Johann-Mattis List. 2021b. Using EDICTOR 2.0 to annotate language-internal cognates in a German wordlist. *Computer-Assisted Language Comparison in Practice*, 4(4).

Johann-Mattis List. 2023. Inference of partial colexifications from multilingual wordlists. *Frontiers in Psychology*, 14(1156540):1–10.

Johann-Mattis List. 2024. Open problems in computational historical linguistics [version 2; peer review: 3 approved, 1 approved with reservations]. *Open Research Europe*, 3(201):1–27.

Johann-Mattis List. forthcoming(a). Computational approaches to historical language comparison. In Claire Bowern and Bethwyn Evans, editors, *Routledge Handbook of Historical Linguistics*, 2 edition, pages 1–20. Routledge, London and New York.

Johann-Mattis List. forthcoming(b). Productive Signs: A computer-assisted analysis of evolutionary, typological, and cognitive dimensions of word families. In *International Conference of Linguists*, 0, pages 1–12. De Gruyter.

Johann-Mattis List and Robert Forkel. 2023a. *LingPy. A Python library for quantitative tasks in historical linguistics [Software Library, Version 2.6.13]*. MCL Chair at the University of Passau, Passau.

Johann-Mattis List and Robert Forkel. 2023b. *LingRex: Linguistic reconstruction with LingPy*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Johann-Mattis List, 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.

Johann-Mattis List, Simon J. Greenhill, and Russell D. Gray. 2017. The potential of automatic word comparison for historical linguistics. *PLOS ONE*, 12(1):1–18.

Johann-Mattis List, Nathan W. Hill, Frederic Blum, and Cristian Juárez. 2024. Grouping sounds into evolving units for the purpose of historical language comparison [version 1; peer review: 2 approved]. *Open Research Europe*, 4(34):1–8.

Johann-Mattis List, Philippe Lopez, and Eric Bapteste. 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)*, pages 599–605, Berlin. Association of Computational Linguistics.

Johann-Mattis List and Steven Moran. 2013. An open source toolkit for quantitative historical linguistics. In *Proceedings of the ACL 2013 System Demonstrations*, pages 13–18, Stroudsburg. Association for Computational Linguistics.

Johann-Mattis List, Mary Walworth, Simon J. Greenhill, Tiago Tresoldi, and Robert Forkel. 2018. Sequence comparison in computational historical linguistics. *Journal of Language Evolution*, 3(2):130–144.

David R. Maddison, David L. Swofford, and Wayne P. Maddison. 1997. NEXUS: an extensible file format for systematic information. *Syst. Biol.*, 46(4):590–621.

Cédric Notredame, Desmond G. Higgins, and Jaap Heringa. 2000. T-Coffee. *Journal of Molecular Biology*, 302:205–217.

Jelena Prokić, Martijn Wieling, and John Nerbonne. 2009. Multiple sequence alignments in linguistics. In *Proceedings of the EACL 2009 Workshop on Language Technology and Resources for Cultural Heritage, Social Sciences, Humanities, and Education*, pages 18–25.

Michele Pulini and Johann-Mattis List. 2024. First steps towards the integration of resources on historical glossing traditions in the history of Chinese: A collection of standardized fǎnqiè spellings from the Guǎngyùn. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7343–7348, Torino, Italy. ELRA and ICCL.

Taraka Rama, Johann-Mattis List, Johannes Wahle, and Gerhard Jäger. 2018. Are automatic methods for cognate detection good enough for phylogenetic reconstruction in historical linguistics? In *Proceedings of the North American Chapter of the Association of Computational Linguistics*, pages 393–400.

Malcom Ross and Mark Durie. 1996. Introduction. In Mark Durie, editor, *The comparative method reviewed. Regularity and irregularity in language change*, pages 3–38. Oxford University Press, New York.

Laurent Sagart, Guillaume Jacques, Yunfan Lai, Robin Ryder, Valentin Thouzeau, Simon J. Greenhill, and Johann-Mattis List. 2019. Dated language phylogenies shed light on the ancestry of Sino-Tibetan. *Proceedings of the National Academy of Science of the United States of America*, 116:10317–10322.

Nathanael E. Schweikhard and Johann-Mattis List. 2020. Developing an annotation framework for word formation processes in comparative linguistics. *SKASE Journal of Theoretical Linguistics*, 17(1):2–26.

Guillaume Segerer and S. Flavier. 2015. *RefLex: Reference Lexicon of Africa*. CNRS, Paris and Lyon.

Sergej A. Starostin. 2005. *Germanic 100 wordlists*. The Tower of Babel, Moscow.

Sergej Anatolévič Starostin. 2000a. Comparative-historical linguistics and lexicostatistics. In *Time depth in historical linguistics*, volume 1 of *Papers in the prehistory of languages*, pages 223–265. McDonald Institute for Archaeological Research, Cambridge.

Sergej Anatolévič Starostin. 2000b. *The STARLING database program*. RGGU, Moscow.

Peter Turchin, Ilja Peiros, and Murray Gell-Mann. 2010. Analyzing genetic connections between languages by matching consonant classes. *Journal of Language Relationship*, 3:117–126.