helayo: Reconstructing Sanskrit texts from manuscript witnesses

Charles Li[1, 2]

1 Centre nationale de la recherche scientifique **2** École des hautes études en sciences sociales

DOI: 10.21105/joss.03971

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Submitted: 01 December 2021
Published: 01 December 2021

Summary

For the vast majority of ancient and medieval texts, the original text itself is no longer extant in a material form. Instead, what we have are manuscripts which are copies of copies of copies, made over the course of hundreds or thousands of years, which accumulate errors and other changes each time they are copied by hand. In order to reconstruct the original text from these imperfect copies, scholars create a stemma — analogous to an evolutionary tree — in order to determine the relationships between manuscripts and to trace those textual changes over time.

Statement of need

Due to the similarities in the methods used in the fields of textual reconstruction and evolutionary biology, textual scholars have begun to employ software created for biologists in order to analyze texts. Specifically, textual scholars are now using sequence alignment algorithms and phylogenetic tree-building packages to help with the reconstruction of ancient texts (Maas, 2013; Phillips-Rodriguez, 2007; Salemans, 2000). However, as bioinformatics becomes increasingly sophisticated, their models and algorithms have become more specific, and less applicable to non-biological sequences.

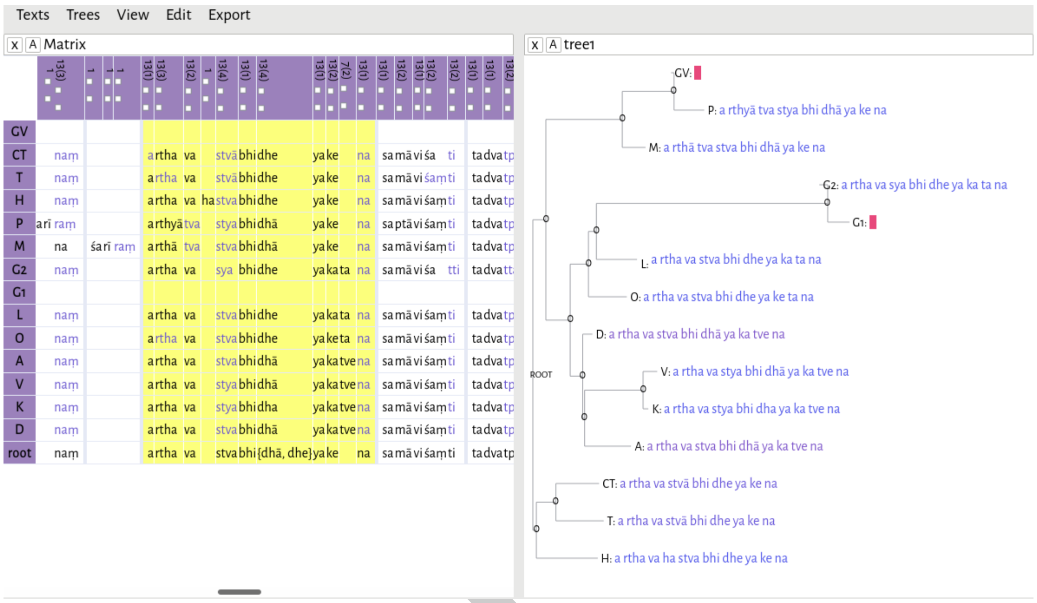
`helayo` has been designed from the ground up to perform multiple sequence alignment specifically for Sanskrit texts. Since Sanskrit has been written in over a dozen different scripts, each with their own orthographic peculiarities depending on their time and place, `helayo` performs a crucial pre-processing step in which the texts are normalized so that they can be compared meaningfully (Li, 2017). `helayo` can also tokenize texts either as individual characters or as *akṣaras*, since the Brahmic scripts used to write Sanskrit are abugidas, in which consonant and vowel pairs are written as a single unit.

In addition, a web-based `matrix editor` can be used to edit an alignment. It is also capable of automatically reconstructing a text based on an alignment and a phylogenetic tree, using the Fitch algorithm (Fitch, 1971). A full tutorial, with example files, is available at https://chchch.github.io/sanskrit-alignment/docs.

Implementation

`helayo` is written in Haskell, and implements the Center Star multiple sequence alignment algorithm (Gusfield, 1997, pp. 347–350) with an affine gap penalty model (Li, 2021). It can be run in three different tokenization modes (character, akṣara, or whitespace-delimited word) and outputs a TEI XML file which can then be edited using the `matrix editor`.

The `matrix editor` is written in Javascript and can be used either online or offline. It loads both TEI XML alignments produced by `helayo` as well as phylogenetic trees in NeXML format, which can be used together to reconstruct a text.



GV					
CT	naṃ		artha va stvā bhidhe yake na	samāviśa ti	tadvatp
T	naṃ		artha va stvā bhidhe yake na	samāviśaṃti	tadvatp
H	naṃ		artha va hastva bhidhe yake na	samāviśaṃti	tadvatp
P	arī raṃ		arthyātva stya bhidhā yake na	saptāviśaṃti	tadvatp
M	na	śarī raṃ	arthā tva stva bhidhā yake na	samāviśaṃti	tadvatp
G2	naṃ		artha va sya bhidhe yakata na	samāviśa tti	tadvatt
G1					
L	naṃ		artha va stva bhidhe yakata na	samāviśaṃti	tadvatp
O	naṃ		artha va stva bhidhe yaketa na	samāviśaṃti	tadvatp
A	naṃ		artha va stva bhidhā yakatvena	samāviśaṃti	tadvatp
V	naṃ		artha va stya bhidhā yakatvena	samāviśaṃti	tadvatp
K	naṃ		artha va stya bhidha yakatvena	samāviśaṃti	tadvatp
D	naṃ		artha va stva bhidhā yakatvena	samāviśaṃti	tadvatp
root	naṃ		artha va stva bhi{dhā, dhe}yake na	samāviśaṃti	tadvatp

GV: 🟥
P: a rthyā tva stya bhi dhā ya ke na
M: a rthā tva stva bhi dhā ya ke na
G2: a rtha va sya bhi dhe ya ka ta na
G1: 🟥
L: a rtha va stva bhi dhe ya ka ta na
O: a rtha va stva bhi dhe ya ke ta na
D: a rtha va stva bhi dhā ya ka tve na
V: a rtha va stya bhi dhā ya ka tve na
K: a rtha va stya bhi dha ya ka tve na
A: a rtha va stva bhi dhā ya ka tve na
CT: a rtha va stvā bhi dhe ya ke na
T: a rtha va stvā bhi dhe ya ke na
H: a rtha va ha stva bhi dhe ya ke na

Figure 1: The matrix editor.

References

Fitch, W. M. (1971). Defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, *20*, 406–316.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences*. Cambridge University Press.

Li, C. (2017). Critical diplomatic editing: Applying text-critical principles as algorithms. In P. Boot, A. Cappellotto, W. Dillen, F. Fischer, A. Kelly, A. Mertgens, A.-M. Sichani, E. Spadini, & D. van Hulle (Eds.), *Advances in digital scholarly editing. Papers presented at the DiXiT conferences in The Hague, Cologne, and Antwerp* (pp. 305–310). Sidestone Press.

Li, C. (2021). Align-affine: Sequence alignment with an affine gap penalty model. In *GitHub repository*. GitHub. https://github.com/chchch/align-affine

Maas, P. A. (2013). On what to do with a stemma – towards a critical edition of the Carakasaṃhitā Vimānasthāna 8.29. In D. Wujastyk, A. Cerulli, & K. Preisendanz (Eds.), *Medical texts and manuscripts in indian cultural history*. Manohar.

Phillips-Rodriguez, W. J. (2007). *Electronic techniques of textual analysis and edition for ancient texts: An exploration of the phylogeny of the Dyūtaparvan* [PhD thesis]. University of Cambridge.

Salemans, B. J. P. (2000). *Building stemmas with the computer in a cladistic, Neo-Lachmannian, way: The case of fourteen text versions of Lanseloet van Denemerken* [PhD thesis]. Katholieke Universiteit Nijmegen.