





Potnia: A Python library for the conversion of transliterated ancient texts to Unicode

Emily Tour ^{1*}, Kabir Manandhar Shrestha ^{2*}, and Robert Turnbull ^{2*}

1 University of Melbourne, Australia 2 Melbourne Data Analytics Platform, University of Melbourne, Australia  Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.07725](https://doi.org/10.21105/joss.07725)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Arfon Smith](#)  

Reviewers:

- [@afrubin](#)
- [@claresloggett](#)

Submitted: 27 October 2024

Published: 03 April 2025

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

- Linear A
- Linear B
- Hittite cuneiform
- Arabic

Summary

Potnia is an open-source Python library designed to convert Romanized transliterations of ancient texts into their respective Unicode representations. Significant progress has been made in the digitization of ancient language corpora. However, many of these datasets are solely presented in transliterated form, even though the necessary Unicode blocks exist to render them using their native script. This restriction to using transliterated datasets for certain ancient scripts has the potential to limit the precision of linguistic analysis via machine learning.

Potnia bridges this gap by providing a flexible framework for converting transliterations into Unicode. By enabling tokenization and processing in the original script, Potnia can optimize tasks such as textual restoration and machine learning-based analysis. The library currently supports:

While Linear B has the most comprehensive test cases and is the most robust, the tool can also be used effectively for the other scripts. The architectural flexibility of Potnia makes it easy to accommodate additional scripts, offering significant value to both computational linguistics and digital humanities by enabling researchers to work with ancient texts in their native scripts.




Statement of Need

While machine learning has increasingly been applied to the study of ancient texts ([Sommer-schild et al., 2023](#)), much of this progress has involved working with transliterated texts, rather than native script formats ([Fetaya et al., 2020](#); [Luo et al., 2019](#); [Papavassileiou et al., 2023](#); [Perono Cacciafoco & Loh, 2021](#)). Although Unicode standards exist for many ancient scripts, transliterated texts remain prevalent due to historical digitization practices.

Transliteration is the process of converting text from its original script into a different script, using systematic processes. It allows those who can understand the secondary script to comprehend the orthography and the approximate pronunciation of the original text. Prior to the gradual introduction of relevant Unicode blocks since the 1990s ([Hossain, 2024](#)), it was also usually necessary for representing non-Latin scripts on Western computational systems, which were largely confined to letters of the Latin alphabet and a small number of special characters.

Transliteration has an important place in aiding new learners of an ancient script to understand the pronunciation and orthography of the underlying language it represents (particularly for

non-alphabetic scripts, where beginners need to grasp a vast repertoire of unfamiliar signs). However, it is well recognized that this process can only provide an approximate, and often unsatisfactory or disputed, representation of the original text (Martinet, 1953; Odisho, 1992; Weinberg, 1974). In particular, a lack of standardized approaches to transliteration can introduce considerable ambiguity and noise into the dataset in a variety of ways, including:

- the use of various notation systems, with different transliterations representing the exact same sign in distinct ways (e.g. where  in Akkadian cuneiform can be represented as either 'mè' or 'me₃');
- changing opinions on particular sign values over time, introducing possible differences between older and newer transliterations (e.g.  in the Linear B script changing from the previously suggested value of 'pa₂' to 'qa')(Chadwick, 1973, pp. 389, 391);
- and the way in which transliteration obscures polyvalency in scripts, where a single sign can represent multiple different values (e.g.  in Hittite cuneiform can represent three different syllables, transliterated as 'har', 'hur' and 'mur', as well as acting as a logogram for three different words, 'ring', 'thick' and 'lung').

For language modelling tasks, we therefore suggest that representations of texts in their native form are preferable to achieve the most accurate results. A number of digitized corpora for well-resourced and widely studied ancient languages are now available in Unicode representations of their native script, including a corpus of ancient Greek (Cerrato et al., 2021), classical Hebrew (Sefaria, 2024), Syriac (Walters, 2020) and Arabic (Nigst et al., 2023). However, many other online text corpora remain restricted to Romanized transliterations (despite the availability of relevant Unicode standards), presumably due to considerations around ease, system limitations and accessibility, e.g. Linear B (Aurora, 2015), Ugaritic (Prosser & Pardee, 2019) and Sumero-Akkadian cuneiform (CDLI contributors, 2024). For this latter group of scripts, current tools capable of converting transliterated ancient texts to the corresponding Unicode appear limited to a handful of individual scripts, such as the various implementations of 'Cuneify' (Tinney, 2019) that handle Sumero-Akkadian cuneiform, the PHP script 'UnicodeConverter', for Egyptian hieroglyphs (Ilin-Tomich, 2019), or 'Anatolian Hieroglyphics (Luwian) generation' tool for Luwian hieroglyphs (Senior, 2023), the latter only available to use through a basic online graphical user interface. While the PyArabic package (Zerrouki, 2023) is able to convert Arabic text to and from the popular Timothy Buckwalter transliteration system, Potnia provides a complementary functionality for the DIN 31635 transliteration system (Deutsches Institut für Normung, 2011) which is widely used in academic literature.

In addition, such transliterations of ancient texts are often heavily annotated, with special characters used to denote a range of features including uncertain readings, missing or damaged elements, erasures, non-textual marks, and annotations by modern transliterators pertaining to structural or physical elements of the document. If not removed or handled appropriately, these have the potential to introduce further noise into language models.

These are the primary gaps we have aimed to address through the development of Potnia. The library's focus on ancient scripts and its extensible architecture make it a valuable asset for researchers working with digitized ancient corpora. It is also equipped to provide specific handling of these elements, with tailored tokenization and regularization rules pertaining to both script-specific and corpus-specific conventions. Potnia therefore enables a key pre-processing step in the language modelling pipeline, with the resulting Unicode outputs of ancient texts enabling more accurate and nuanced computational analysis of these texts in downstream modelling tasks.

Implementation

Potnia is implemented in Python with an extensible architecture centered around the `Script` class, which converts transliterated texts into Unicode representations. It is designed to handle the complexities of ancient scripts through a flexible and customizable framework.

Key Features

1. **YAML-Based Mapping and Rule Specification:** Each script in Potnia (e.g. Linear A, Linear B, Arabic, Hittite cuneiform) is configured via a single YAML file that contains syllabograms, logograms, and rules for transliteration and regularization. This unified structure simplifies updates, scales easily for new scripts, and eliminates the need for hardcoded source files (fig. 1).

```
mappings:
#####
##### Syllabograms Common to Linear A and Linear B #####
#####
# LB SOUND VALUES KNOWN
a: 𐎠
e: 𐎡
i: 𐎢
o: 𐎣
u: 𐎤
```

Figure 1: Example of YAML mapping specification.

2. **Tokenization:** The `tokenize_transliteration` method applies complex symbol replacements and regular expressions to transliterated text based on the rules specified in the YAML file. This tokenization process ensures that the text is split accurately into its meaningful components, handling special symbols and spacing using placeholders, and preparing the text for Unicode conversion.
3. **Transliteration to Unicode:** Potnia uses the `__call__` method to convert the transliterated text to its Unicode representation (fig. 2).

```
1 from potnia import linear_b
2 text = "po-ti-ni-ja"
3
4 tokens = linear_b.tokenize_transliteration(text)
5 print(tokens) # Output: ['po', 'ti', 'ni', 'ja']
6
7 unicode_text = linear_b(text)
8 print(unicode_text) # Output: 𐎠𐎡𐎢𐎣
```

Figure 2: Example of using Potnia.

4. **Regularization of Text:** The `regularize` method applies a series of regular expression rules to clean and normalize the Unicode output. It removes unnecessary tags, ignores patterns specified in the YAML file (e.g. annotations or uncertain characters), and ensures that only the essential characters are retained. This step ensures the output is refined and ready for downstream tasks.
5. **Comprehensive Testing:** Pytest fixtures allow us to define test cases as lines in YAML files which allowed us to consisely add over 360 test examples, covering a broad range of edge cases. The code coverage of the tests is 100%.
6. **Versatile Interface Options** Users can interact with Potnia as a Python library, or through the command line interface (CLI), or through the graphical user interface (GUI) (fig. 3)



Figure 3: Example of using the Potnia GUI.

Research Application

Potnia's design and functionality address the following challenges in the analysis of ancient texts:

1. **Extensibility:** Potnia is designed to be highly extensible, allowing researchers to integrate new scripts by defining script-specific rules for tokenization and conversion. This flexibility makes the library suitable for a wide range of ancient scripts that are not yet represented in Unicode, providing a valuable tool for researchers across various fields of ancient studies.
2. **Integration with Research Workflows:** Researchers can easily incorporate Potnia into their existing workflows. For example, in a typical research scenario, Potnia could be used to preprocess a corpus of Linear B texts before feeding them into a machine learning model for further analysis.

As part of a broader initiative to develop language models for ancient language research, Potnia serves as a foundational component by converting Romanized transliterations of Linear B texts into Unicode datasets for computational analysis. These datasets enable the development of language-specific models supporting tasks such as text generation, restoration and vector embedding analysis. The library's modular design facilitates its application to additional ancient scripts, contributing to broader research initiatives in computational philology.

Availability

Potnia is open-source software released under the Apache 2.0 license. It is available through PyPI <https://pypi.org/project/potnia/> and GitHub <https://github.com/AncientNLP/potnia>. We welcome contributions from the community and adhere to the Contributor Covenant Code of Conduct. Documentation is available at <https://ancientnlp.github.io/potnia/>.

Acknowledgements

We acknowledge support from Wytamma Wirth, Brent Davis, Kim Doyle, Man-Hua (Kate) Chu, Anhui (Ellie) Situ, Ekaterina Vylomova, Chris Guest and Stavroula (Stephie) Nikoloudis. This research was supported by The University of Melbourne's Research Computing Services. Robert Turnbull completed part of this work through the BICROSS project, which has received funding from the European Research Council (ERC) under the European Union's Horizon Europe research and innovation programme (grant agreement no. 101043730 – BICROSS – ERC-2021-COG).

References

- Aurora, F. (2015). DĀMOS (Database of Mycenaean at Oslo). Annotating a fragmentarily attested language. *Procedia - Social and Behavioral Sciences*, 198, 21–31. <https://doi.org/10.1016/j.sbspro.2015.07.415>
- CDLI contributors. (2024). *Cuneiform Digital Library Initiative*. <https://cdli.mpiwg-berlin.mpg.de/>.
- Cerrato, L., Almas, B., TDBuck, ahanhardt, srdee, Babeu, A., Clérice, T., Fleischman, S., gregorycrane, Munson, M., Berra, A., Mittmann, A., Palladino, C., KATEBHN, Sowell, E., Kalvesmaki, J., Scott, S., Hellingman, J., Andrei, & Drymon, C. (2021). *Canonical greek literature* (Version 0.0.2867). Zenodo. <https://doi.org/10.5281/zenodo.5090923>
- Chadwick, J. (1973). *Documents in Mycenaean Greek* (2nd ed.). Cambridge University Press.
- Deutsches Institut für Normung. (2011). *DIN 31635: Transliteration of the Arabic alphabet*. Standard published by Deutsches Institut für Normung.
- Fetaya, E., Lifshitz, Y., Aaron, E., & Gordin, S. (2020). Restoration of fragmentary Babylonian texts using recurrent neural networks. *Proceedings of the National Academy of Sciences*, 117(37), 22743–22751. <https://doi.org/10.1073/pnas.2003794117>
- Hossain, A. (2024). Text Standards for the “Rest of World”: The Making of the Unicode Standard and the OpenType Format. *IEEE Annals of the History of Computing*, 46(1), 20–33. <https://doi.org/10.1109/MAHC.2024.3351948>
- Ilin-Tomich, A. (2019). *UnicodeConverter*. <https://github.com/ailintom/UnicodeConverter/>.
- Luo, J., Cao, Y., & Barzilay, R. (2019). Neural decipherment via minimum-cost flow: From Ugaritic to Linear B. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3146–3155. <https://doi.org/10.18653/v1/P19-1303>
- Martinet, A. (1953). A Project of Transliteration of Classical Greek. *WORD*, 9(2), 152–161. <https://doi.org/10.1080/00437956.1953.11659466>
- Nigst, L., Romanov, M., Savant, S. B., Seydi, M., & Verkinderen, P. (2023). *OpenITI: a Machine-Readable Corpus of Islamicate Texts* (Version 2023.1.8) [Data set]. Zenodo. <https://doi.org/10.5281/zenodo.10021513>
- Odisho, E. Y. (1992). Transliterating English in Arabic. *Zeitschrift Für Arabische Linguistik*, 24, 21–34. <http://www.jstor.org/stable/43525603>
- Papavassileiou, K., Kosmopoulos, D. I., & Owens, G. (2023). A generative model for the Mycenaean Linear B script and its application in infilling text from ancient tablets. *Journal on Computing and Cultural Heritage*, 16(3), 1–25. <https://doi.org/10.1145/3593431>
- Perono Cacciafoco, F., & Loh, C. J. S. (2021). A New Approach to the Decipherment of Linear A, Stage 2 - Cryptanalysis and Language Deciphering: A “Brute Force Attack” on an Undeciphered Writing System. *Grapholinguistics in the 21st Century 2020. Proceedings*, 927–943. <https://doi.org/10.36824/2020-graf-cacc>
- Prosser, M. C., & Pardee, D. G. (2019). *The Ras Shamra Tablet Inventory*. <https://onlinepublications.uchicago.edu/RSTI/>.
- Sefaria. (2024). *Sefaria: A living library of jewish texts online*. <https://www.sefaria.org>.
- Senior, A. (2023). *Anatolian hieroglyphics (luwian) generation*. <https://andrewsenior.com/luwian/>.
- Sommerschild, T., Assael, Y., Pavlopoulos, J., Stefanak, V., Senior, A., Dyer, C., Bodel, J., Prag, J., Androutsopoulos, I., & Freitas, N. D. (2023). Machine learning for ancient languages: A survey. *Computational Linguistics*, 49(3), 1–45. <https://doi.org/10.1162/>

[coli_a_00481](#)

- Tinney, S. (2019). *Cuneify*. Oracc: The Open Richly Annotated Cuneiform Corpus. <http://oracc.museum.upenn.edu/doc/tools/cuneify/>
- Walters, J. E. (2020). The Digital Syriac Corpus: A Digital Repository for Syriac Texts. *Zeitschrift Für Antikes Christentum / Journal of Ancient Christianity*, 24(1), 109–122. <https://doi.org/10.1515/zac-2020-0018>
- Weinberg, B. (1974). Transliteration in documentation. *Journal of Documentation*, 30(1), 18–31. <https://doi.org/10.1108/eb026567>
- Zerrouki, T. (2023). PyArabic: A Python package for Arabic text. *Journal of Open Source Software*, 8(84), 4886. <https://doi.org/10.21105/joss.04886>