The Classical Language Toolkit: An NLP Framework for Pre-Modern Languages

Kyle P. Johnson, Clément Besnier, and Todd Cook

Digital Classicist Seminars
Zentrum Grundlagenforschung Alte Welt
Berlin-Brandenburg Academy of Sciences and Humanities
December 14, 2021

Presentation

1. Kyle P. Johnson: CLTK Overview
2. Clément Besnier: Demo via Old Norse Example
3. Todd Cook: BERT & MLOps

CLTK Outline

1. Pre-Modern NLP
2. System Design
3. CLTK Architecture

Abstract

- **Problem:**
  - Most NLP for living languages, neglects non-spoken historical languages
  - Scholars of pre-modern languages often have different goals than those of living-language researchers
- **Solution:** An NLP framework for pre-modern languages with a modular processing pipeline that balances the competing demands of algorithmic diversity with pre-configured defaults.

1. Pre-Modern NLP

NLP for Pre-Modern Languages

Pre-modern languages have traits distinguishing them from living languages, including:

- **A finite corpus**: Since native speakers no longer generate new texts, corpora may be too small for some machine learning algorithms, thus requiring rules-based or hybrid approaches.
- **Variation**: Corpora of pre-modern languages are likely to demonstrate greater variation than living languages.
- **Limited resources**: Interest in pre-modern languages is largely scholarly or religious, meaning less funding from government and industry.

NLP for Pre-Modern Researchers

Researchers of pre-modern languages have concerns that are likely *philological*, *linguistic*, or *pedagogical*.

- **Philology**: Philology is an approach to pre-modern writing that focuses on the historical origins of texts; it is comparative as well as genealogical in nature.
- **Linguistics**: Historical linguists study diachronic change in a language itself, as opposed to philologists' focus upon written language.
- **Pedagogy**: Students do not learn by speaking but reading original texts.

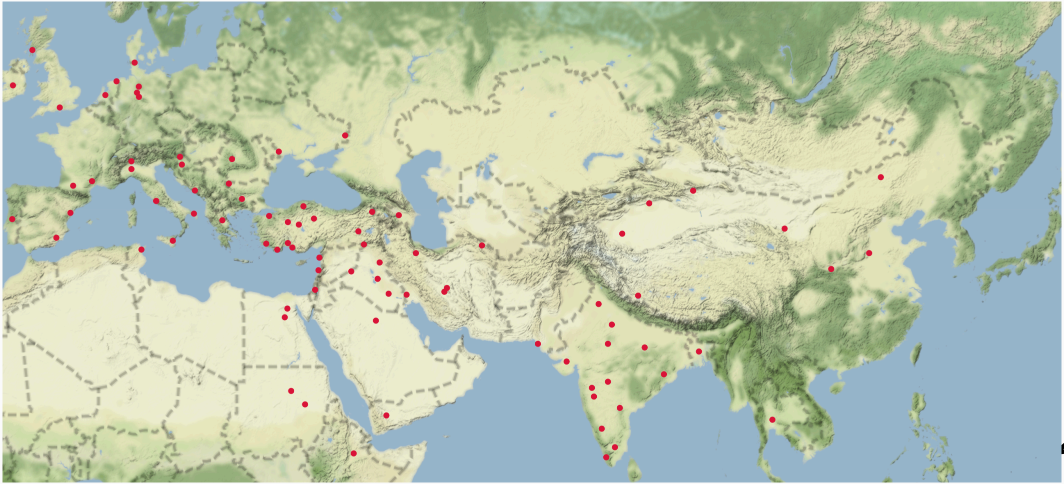
Definitions

- **Pre-modern language:** encompasses the ISO 639-3 definitions of:
    - *ancient*, *extinct*, and *historic* (SIL)
    - 219 languages between the 33rd century B.C. (Sumerian) up until the start of the A.D. 19th century
- **Framework & pipeline:**
    - Frameworks make the technology easier for non-specialists to use (e.g., NLTK)
    - Pipelines have default algorithms are run in series upon input text (e.g., Stanza, spaCy)

219 Pre-Modern Languages

Aequian, Aghwan, **Akkadian**, Alanic, **Ancient Greek**, Ancient Hebrew, Ancient Ligurian, Ancient Macedonian, Ancient North Arabian, Ancient Zapotec, Andalusian Arabic, Anglo-Norman, Aquitanian, Ardhamāgadhī Prākrit, Armazic, Avestan, Bactrian, Bengali, Bolgarian, Burma Pyu, Camunic, Carian, Celtiberian, **Church Slavic**, Cisalpine Gaulish, Classical Armenian, Classical Mandaic, Classical Mongolian, Classical Nahuatl, Classical Newari, Classical Quechua, Classical Syriac, Classical Tibetan, **Coptic**, Cumbric, Cuneiform Luwian, Curonian, Dacian, Early Irish, Early Tripuri, **Eastern Panjabi**, Eblaite, Edomite, Egyptian (Ancient), Elamite, Elymian, Epi-Olmec, Epigraphic Mayan, Eteocretan, Eteocypriot, Etruscan, Faliscan, Galatian, Galindan, Geez, **Gothic**, Gujarati, Gāndhārī, Hadrami, Harami, Harappan, Hattic, Hernican, Hiberno-Scottish Gaelic, Hieroglyphic Luwian, **Hindi**, Hittite, Hunnic, Hurrian, Iberian, Illyrian, Jutish, Kajkavian, Kannada, Kara (Korea), Karakhanid, Kaskean, Kawi, Khazar, Khorezmian, Khotanese, Khwarezmian, Kitan, Koguryo, Langobardic, **Latin**, Lemnian, Lepontic, Liburnian, Linear A, **Literary Chinese**, Lusitanian, Lycian A, Lydian, Maek, Maharastri Prakrit, Malayalam, Manichaean Middle Persian, Marrucinian, Marsian, Median, Meroitic, Messapic, Middle Armenian, Middle Breton, Middle Chinese, Middle Cornish, Middle Dutch, **Middle English**, **Middle French**, **Middle High German**, Middle Hittite, Middle Irish (10-12th century), Middle Korean (10th-16th cent.), Middle Low German, Middle Mongol, Middle Newar, Middle Welsh, Milyan, Minaean, Minoan, Moabite, Mozarabic, Mycenaean Greek, Mysian, Nadruvian, Neo-Hittite, Noric, North Picene, Numidian, Odia, **Official Aramaic (700-300 BCE)**, Old Aramaic (up to 700 BCE), Old Avar, Old Breton, Old Burmese, Old Chinese, Old Cornish, Old Dutch-Old Frankish, **Old English (ca. 450-1100)**, Old Frankish, **Old French (842-ca. 1400)**, Old Frisian, Old Georgian, Old High German (ca. 750-1050), Old Hittite, Old Hungarian, Old Japanese, Old Korean (3rd-9th cent.), Old Lithuanian, Old Manipuri, Old Marathi, Old Mon, **Old Norse**, Old Nubian, Old Ossetic, Old Persian (ca. 600-400 B.C.), Old Provençal, Old Russian, Old Saxon, Old Spanish, Old Tamil, Old Tibetan, Old Turkic, Old Turkish, Old-Middle Welsh, Oscan, Ottoman Turkish (1500-1928), Paekche, Paelignian, Pahlavi, Palaic, Palestinian Jewish Aramaic, **Pali**, Parthian, Pecheneg, Phoenician, Phrygian, Pictish, Pisidian, Primitive Irish, Punic, Puyo, Puyo-Paekche, Qatabanian, Raetic, Sabaic, Sabine, **Sanskrit**, Sauraseni Prakrit, Scythian, Sicana, Sicula, Siculo Arabic, Sidetic, Skalvian, Sogdian, Sorothaptic, South Picene, **Standard Arabic**, Sumerian, Tangut, Tartessian, Telugu, Thracian, Tokharian A, Tokharian B, Transalpine Gaulish, Tumshuqese, Ugaritic, Umbrian, Urartian, Urdu, Vandalic, Venetic, Vestinian, Volscian, Western Farsi, Zhangzhung

Map of Pre-Modern Languages



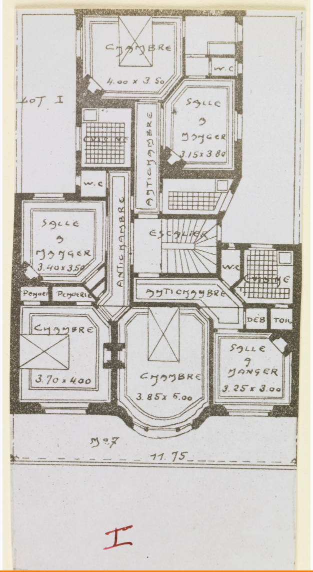
2. System Design

Requirements

An NLP pipeline within a framework architecture standardizes I/O while preserving algorithmic diversity. The CLTK should provide:

- **Modular processing pipelines**: Each language should come with a pre-configured pipeline set to defaults expected by most users.
- **Diversity of algorithms**: When there are several popular ways researchers perform a particular process
- **Standard I/O**: an API should accept standard input for all human languages
- **Model management**: The project must provide models for every pipeline.

Multilingual NLP Framework: An Analogy

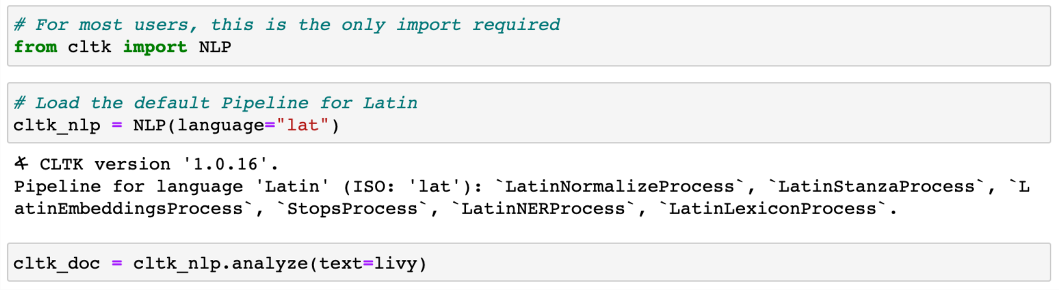


3. CLTK Architecture

Processes

- `NormalizeProcess`
- `TokenizationProcess`
- `SentenceProcess`
- `StopsProcess`
- `LemmatizationProcess`
- `MorphologyProcess`
- `PhonologyProcess`
- `StemmingProcess`
- `WordNetProcess`
- `LexiconProcess`
- `NERProcess`
- `DependencyProcess`
- `ProsodyProcess`
- `EmbeddingsProcess`
- `StanzaProcess`

Run Pipeline with `NLP()`



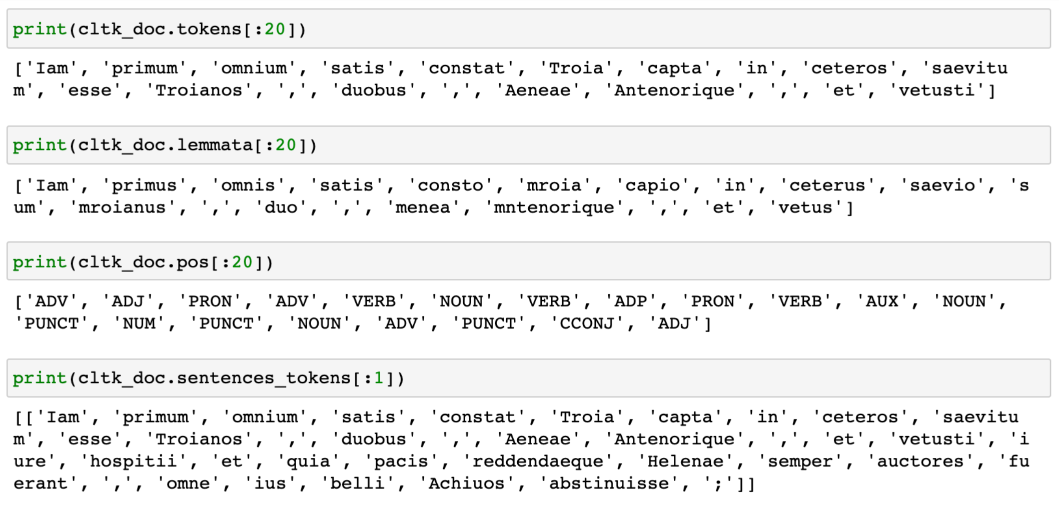
```python
# For most users, this is the only import required
from cltk import NLP
```

```python
# Load the default Pipeline for Latin
cltk_nlp = NLP(language="lat")
```

```
𐤀 CLTK version '1.0.16'.
Pipeline for language 'Latin' (ISO: 'lat'): `LatinNormalizeProcess`, `LatinStanzaProcess`, `L
atinEmbeddingsProcess`, `StopsProcess`, `LatinNERProcess`, `LatinLexiconProcess`.
```

```python
cltk_doc = cltk_nlp.analyze(text=livy)
```

Inspect Doc



```
print(cltk_doc.tokens[:20])
```

```
['Iam', 'primum', 'omnium', 'satis', 'constat', 'Troia', 'capta', 'in', 'ceteros', 'saevitu
m', 'esse', 'Troianos', ',', 'duobus', ',', 'Aeneae', 'Antenorique', ',', 'et', 'vetusti']
```

```
print(cltk_doc.lemmata[:20])
```

```
['Iam', 'primus', 'omnis', 'satis', 'consto', 'mroia', 'capio', 'in', 'ceterus', 'saevio', 's
um', 'mroianus', ',', 'duo', ',', 'menea', 'mntenorique', ',', 'et', 'vetus']
```

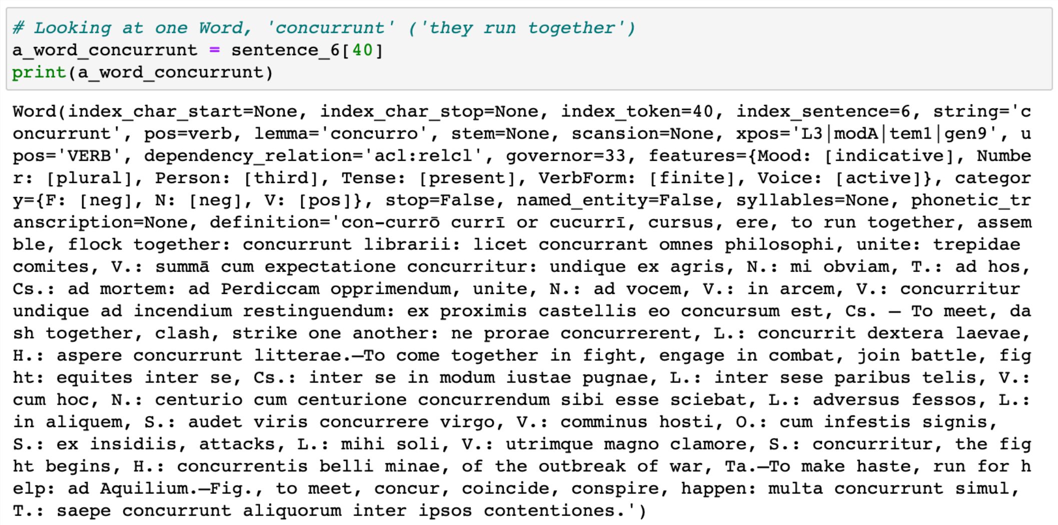
```
print(cltk_doc.pos[:20])
```

```
['ADV', 'ADJ', 'PRON', 'ADV', 'VERB', 'NOUN', 'VERB', 'ADP', 'PRON', 'VERB', 'AUX', 'NOUN',
'PUNCT', 'NUM', 'PUNCT', 'NOUN', 'ADV', 'PUNCT', 'CCONJ', 'ADJ']
```

```
print(cltk_doc.sentences_tokens[:1])
```

```
[['Iam', 'primum', 'omnium', 'satis', 'constat', 'Troia', 'capta', 'in', 'ceteros', 'saevitu
m', 'esse', 'Troianos', ',', 'duobus', ',', 'Aeneae', 'Antenorique', ',', 'et', 'vetusti', 'i
ure', 'hospitii', 'et', 'quia', 'pacis', 'reddendaeque', 'Helenae', 'semper', 'auctores', 'fu
erant', ',', 'omne', 'ius', 'belli', 'Achiuos', 'abstinuisse', ';']]
```

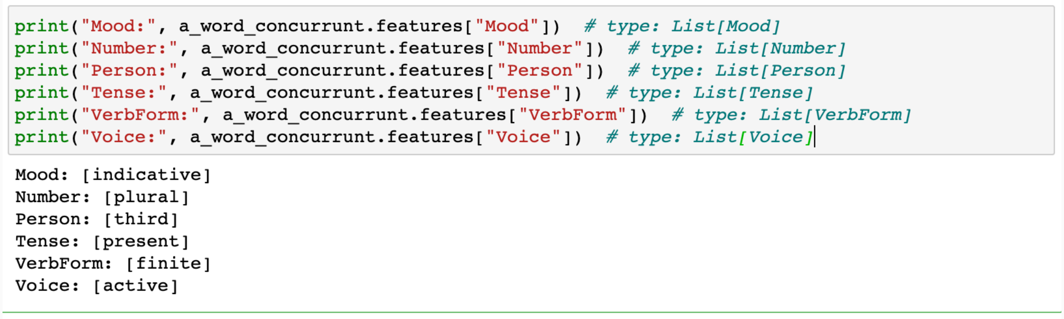
Inspect Word



```python
# Looking at one Word, 'concurrunt' ('they run together')
a_word_concurrunt = sentence_6[40]
print(a_word_concurrunt)
```

```
Word(index_char_start=None, index_char_stop=None, index_token=40, index_sentence=6, string='c
oncurrunt', pos=verb, lemma='concurro', stem=None, scansion=None, xpos='L3|modA|tem1|gen9', u
pos='VERB', dependency_relation='acl:relcl', governor=33, features={Mood: [indicative], Numbe
r: [plural], Person: [third], Tense: [present], VerbForm: [finite], Voice: [active]}, categor
y={F: [neg], N: [neg], V: [pos]}, stop=False, named_entity=False, syllables=None, phonetic_tr
anscription=None, definition='con-currō currī or cucurrī, cursus, ere, to run together, assem
ble, flock together: concurrunt librarii: licet concurrant omnes philosophi, unite: trepidae
comites, V.: summā cum expectatione concurritur: undique ex agris, N.: mi obviam, T.: ad hos,
Cs.: ad mortem: ad Perdiccam opprimendum, unite, N.: ad vocem, V.: in arcem, V.: concurritur
undique ad incendium restinguendum: ex proximis castellis eo concursum est, Cs. — To meet, da
sh together, clash, strike one another: ne prorae concurrerent, L.: concurrit dextera laevae,
H.: aspere concurrunt litterae.—To come together in fight, engage in combat, join battle, fig
ht: equites inter se, Cs.: inter se in modum iustae pugnae, L.: inter sese paribus telis, V.:
cum hoc, N.: centurio cum centurione concurrendum sibi esse sciebat, L.: adversus fessos, L.:
in aliquem, S.: audet viris concurrere virgo, V.: comminus hosti, O.: cum infestis signis,
S.: ex insidiis, attacks, L.: mihi soli, V.: utrimque magno clamore, S.: concurritur, the fig
ht begins, H.: concurrentis belli minae, of the outbreak of war, Ta.—To make haste, run for h
elp: ad Aquilium.—Fig., to meet, concur, coincide, conspire, happen: multa concurrunt simul,
T.: saepe concurrunt aliquorum inter ipsos contentiones.')
```

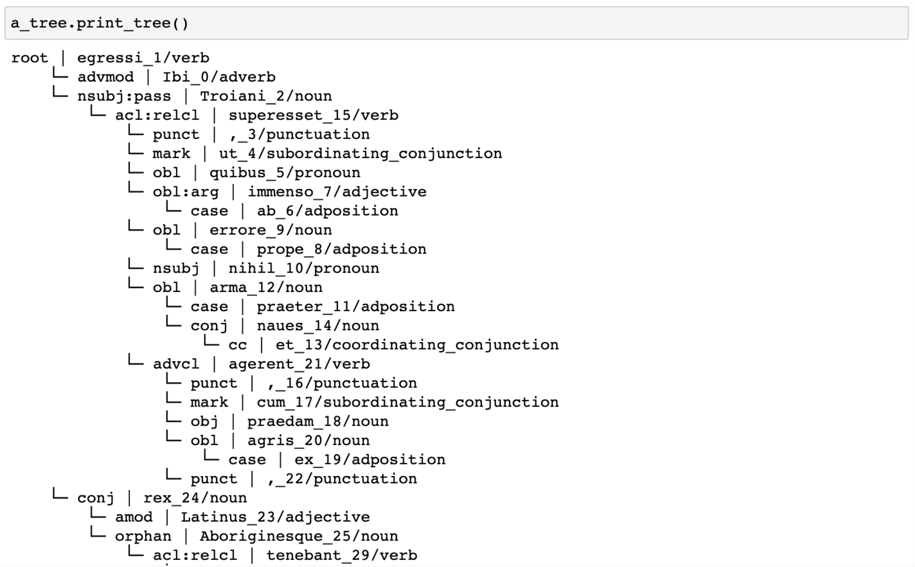
Modeling Morphology with `MorphosyntacticFeature`



```
print("Mood:", a_word_concurrunt.features["Mood"])  # type: List[Mood]
print("Number:", a_word_concurrunt.features["Number"])  # type: List[Number]
print("Person:", a_word_concurrunt.features["Person"])  # type: List[Person]
print("Tense:", a_word_concurrunt.features["Tense"])  # type: List[Tense]
print("VerbForm:", a_word_concurrunt.features["VerbForm"])  # type: List[VerbForm]
print("Voice:", a_word_concurrunt.features["Voice"])  # type: List[Voice]
```

```
Mood: [indicative]
Number: [plural]
Person: [third]
Tense: [present]
VerbForm: [finite]
Voice: [active]
```

Modeling Syntax with `DependencyTree`



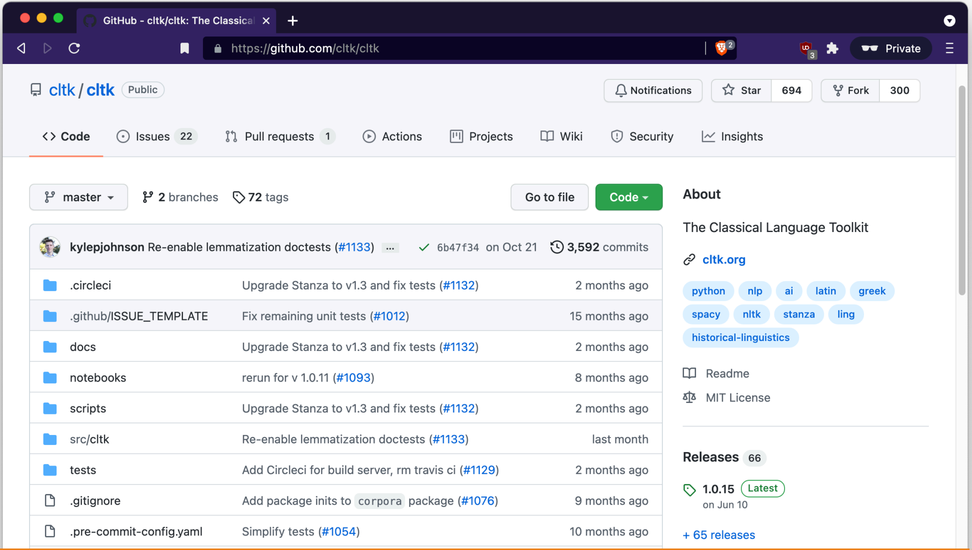
```
a_tree.print_tree()
```

```
root | egressi_1/verb
    └─ advmod | Ibi_0/adverb
    └─ nsubj:pass | Troiani_2/noun
        └─ acl:relcl | superesset_15/verb
            └─ punct | ,_3/punctuation
            └─ mark | ut_4/subordinating_conjunction
            └─ obl | quibus_5/pronoun
            └─ obl:arg | immenso_7/adjective
                └─ case | ab_6/adposition
            └─ obl | errore_9/noun
                └─ case | prope_8/adposition
            └─ nsubj | nihil_10/pronoun
            └─ obl | arma_12/noun
                └─ case | praeter_11/adposition
                └─ conj | naues_14/noun
                    └─ cc | et_13/coordinating_conjunction
            └─ advcl | agerent_21/verb
                └─ punct | ,_16/punctuation
                └─ mark | cum_17/subordinating_conjunction
                └─ obj | praedam_18/noun
                └─ obl | agris_20/noun
                    └─ case | ex_19/adposition
                └─ punct | ,_22/punctuation
    └─ conj | rex_24/noun
        └─ amod | Latinus_23/adjective
        └─ orphan | Aboriginesque_25/noun
            └─ acl:relcl | tenebant_29/verb
```

19 Pre-Modern Languages

- Akkadian
- Ancient Greek
- Church Slavic
- Coptic
- Eastern Panjabi
- Gothic
- Hindi
- Latin
- Literary Chinese
- Middle English
- Middle French
- Middle High German
- Official Aramaic (700-300 BCE)
- Old English (ca. 450-1100)
- Old French (842-ca. 1400)
- Old Norse
- Pali
- Sanskrit
- Standard Arabic

FOSS as Org


GitHub - cltk/cltk: The Classical
https://github.com/cltk/cltk
Private

cltk / cltk Public

Notifications | Star 694 | Fork 300

Code | Issues 22 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

master | 2 branches | 72 tags | Go to file | Code

kylepjohnson Re-enable lemmatization doctests (#1133) ... ✓ 6b47f34 on Oct 21 | 3,592 commits

.circleci	Upgrade Stanza to v1.3 and fix tests (#1132)	2 months ago
.github/ISSUE_TEMPLATE	Fix remaining unit tests (#1012)	15 months ago
docs	Upgrade Stanza to v1.3 and fix tests (#1132)	2 months ago
notebooks	rerun for v 1.0.11 (#1093)	8 months ago
scripts	Upgrade Stanza to v1.3 and fix tests (#1132)	2 months ago
src/cltk	Re-enable lemmatization doctests (#1133)	last month
tests	Add Circleci for build server, rm travis ci (#1129)	2 months ago
.gitignore	Add package inits to `corpora` package (#1076)	9 months ago
.pre-commit-config.yaml	Simplify tests (#1054)	10 months ago

About

The Classical Language Toolkit

cltk.org

python nlp ai latin greek spacy nltk stanza ling historical-linguistics

Readme

MIT License

Releases 66

1.0.15 Latest
on Jun 10

+ 65 releases

People

Maintainers

- Kyle P. Johnson
- Patrick J. Burns
- John Stewart
- Todd G. Cook
- Clément Besnier
- William J. B. Mattingly

Academic Advisors

- Neil Coffee, University at Buffalo
- Gregory Crane, Tufts University
- Peter Meineck, New York University
- Leonard Muellner, Brandeis University

Also 90+ contributors over the past 6 years.

Ongoing Work

- To create evaluation benchmarks for each NLP task, for each language
- To make a `TrainingPipeline`, similar to the inference `Pipeline`, that would standardize the training of new models
- to develop Internet infrastructure for training and hosting models

- Home: http://cltk.org/
- Code: https://github.com/cltk/cltk
- Docs: https://docs.cltk.org/
- Tutorial: https://github.com/cltk/cltk/blob/master/notebooks/CLTK%20Demonstration.ipynb