EvaLatin 2026

-

Dependency Parsing Shared Task Guidelines

Version 1.1
February 17, 2026

Federica Iurescia[1], Marco Passarotti[1], Rachele Sprugnoli[1]
1. CIRCSE Research Centre, Università Cattolica del Sacro Cuore
Largo Agostino Gemelli 1, 20123 Milan, Italy
federica.iurescia[AT]unicatt.it
marco.passarotti[AT]unicatt.it
rachele.sprugnoli[AT]unicatt.it

Contents

Chapter 1

Introduction

EvaLatin 2026 is the fourth edition of the evaluation campaign of Natural Language Processing (NLP) tools for the Latin language. The campaign is designed with the aim of answering two main questions:

- How can we promote the development of resources and language technologies for the Latin language?
- How can we foster collaboration among scholars working on Latin and attract researchers from different disciplines?

EvaLatin 2020 [7] was organized around 2 tasks (i. e. Lemmatization and PoS Tagging); EvaLatin 2022 [8] was organized around 3 tasks (i. e. Lemmatization, PoS Tagging, and Features Identification); EvaLatin 2024 [6] was organized around 2 tasks (i. e. Dependency Parsing and Emotion Polarity Detection). For EvaLatin 2026, we propose 2 tasks (i. e. Dependency Parsing and Named Entity Recognition). Shared data and a scorer are provided to the participants. Participants can choose to participate in either one or both tasks. The organizers rely on the honesty of all participants who might have some prior knowledge of part of the data used for evaluation not to unfairly use such knowledge.

EvaLatin 2026 is organized within the "Fourth Workshop of Language Technologies for Historical and Ancient Languages" (LT4HALA 2026), colocated at LREC 2026.[1] The workshop will be held in Palma de Mallorca, Spain, on 11th May, 2026. EvaLatin 2026 is organized by the CIRCSE research centre at the Università Cattolica del Sacro Cuore in Milan, Italy.

For any update, please check the LT4HALA 2026 website: `https://circse.github.io/LT4HALA/2026/`.

[1] `https://lrec2026.info/`

Chapter 2

Data

Dependency parsing is based on the Universal Dependencies (UD) framework.[1] No specific training data are released but participants are free to make use of any (kind of) resource they consider useful for the task, including the Latin treebanks already available in the UD collection. In this regard, one of the challenges of this task is to understand which treebank (or combination of treebanks) is the most suitable to deal with new test data.[2] Test data are both prose and poetic texts from different time periods. Some of the datasets include punctuation, some do not, as this is the actual state of the art for the Latin treebanks. The task aims at improving a state of the art that is not optimal. UD treebanks currently show different degrees of harmonization, and Latin is not an exception in this respect. The diversity of the data currently available for this task is an issue we are aware of, and that needs to be addressed. This evaluation campaign aims at addressing this issue, and among the desired outcomes there are strategies to deal with it successfully. Given the current widespread use of Large Language Models (LLMs) in all areas of language processing, we expect that they will also be used for this task. Information about the models and prompts used must be detailed in the final report.

2.1 Format

This section gives details on the format of test data for 2026 Dependency Parsing shared task.

Test data for the task of Dependency Parsing are distributed in the CoNLL-U format.[3] Following such format, the annotations are plain text files having the `.conllu` extension and encoded in UTF-8 containing:

- 2 comment lines starting with a hashtag (`#`): one line specifies the sentence unique code (`# sent_id`), while the other line reports the sentence string (`# text`).

[1] `https://universaldependencies.org/`

[2] See [3] for further details.

[3] `https://universaldependencies.org/format.html`

- Lines composed of 10 tab-separated fields and containing either the UD-style annotation of a single-word token, or denoting a multi-word token. When values for a given field are not determined, an underscore (_) is used instead (there are no empty fields). The 10 fields of the CoNLL-U format are as follows:

  1. `ID`: numerical index of the word in the sentence. For single-word tokens, a progressive integer starting from 1 for each new sentence; for multi-word tokens, it is a range in the form of $i-(i+n)$, meaning that the token in the sentence string is split into $n+1$ words, with indices from i to $i+n$, each with its own analysis;

  2. `FORM`: word form as occurring in the sentence string. For multi-word tokens, the form of a single word is normalized, whereas it might be contracted, or otherwise subject to changes, in the univerbation (e. g. *imprimis* would be decomposed as *in* and *primis*);

  3. `LEMMA`: form conventionally representing the lexeme related to the word form;

  4. `UPOS`: universal PoS tag of the word;[4]

  5. `XPOS`: language-specific PoS of the word (as used e. g. in the same treebank prior to conversion to the UD annotation style);

  6. `FEATS`: list of morpholexical features of the word, either "universal" or "language-specific";[5]

  7. `HEAD`: index of the syntactic head of the word (possibly `0` if the word is the sentence root);

  8. `DEPREL`: universal syntactic dependency relation with respect to the `HEAD`;[6]

  9. `DEPS`: syntactic relations in the enhanced dependency graph;

  10. `MISC`: any other annotation.

- Blank lines marking boundaries between sentences and the end of the document.

[4] `https://universaldependencies.org/u/pos/index.html`
[5] `https://universaldependencies.org/u/feat/index.html`
[6] `https://universaldependencies.org/u/dep/index.html`

An example of the CoNLL-U format is given in Figure 1. In our test dataset, the fields HEAD and DEPREL are filled in with underscores. This format is used for the test data and participants are expected to produce the same format for the final evaluation. An example of the test data format is given in Figure 2.

```
# sent_id = CaesBG4-A-01-607
# text = neque multum frumento sed maximam partem lacte atque pecore uiuunt multumque sunt in uenationibus
1   neque   neque   CCONJ   S   Polarity=Neg    2   cc  _   LiLaflcat=i
2   multum  multum  ADV M   Degree=Pos  10  advmod  _   LASLAVariant=2|LiLaflcat=i
3   frumento    frumentum   NOUN    A2  Case=Abl|Gender=Neut|InflClass=IndEur0|Number=Sing  10  obl _   LiLaflcat=n2
4   sed sed CCONJ   S   _   6   cc  _   LiLaflcat=i
5   maximam magnus  ADJ C1  Case=Acc|Degree=Abs|Gender=Fem|InflClass=IndEurA|Number=Sing    6   amod    _   LiLaflcat=n6
6   partem  pars    NOUN    A3  Case=Acc|Gender=Fem|InflClass=IndEurI|Number=Sing   10  obj _   LiLaflcat=n3
7   lacte   lac NOUN    A3  Case=Abl|Gender=Masc|InflClass=IndEurI|Number=Sing  10  obl _   LiLaflcat=n3
8   atque   atque   CCONJ   S   _   9   cc  _   LASLAVariant=1|LiLaflcat=i
9   pecore  pecus   NOUN    A3  Case=Abl|Gender=Neut|InflClass=IndEurX|Number=Sing  7   conj    _   LASLAVariant=1|LiLaflcat=n3
10  uiuunt  uiuo    VERB    B3  Aspect=Imp|InflClass=LatX|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act   0   root    _   LiLaflcat=v3
11-12   multumque   _   _   _   _   _   _   _   _
11  multum  multum  ADV M   Degree=Pos  15  advmod  _   LASLAVariant=2|LiLaflcat=i
12  que que CCONJ   S   _   15  cc  _   LiLaflcat=i
13  sunt    sum AUX B6  Aspect=Imp|InflClass=LatAnom|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin  15  cop _   LASLAVariant=1|LiLaflcat=v6
14  in  in  ADP R   AdpType=Prep    15  case    _   LiLaflcat=i
15  uenationibus    uenatio NOUN    A3  Case=Abl|Gender=Fem|InflClass=IndEurX|Number=Plur   10  conj    _   LiLaflcat=n3
```

Figure 1: Example of CoNLL-U file.

```
# sent_id = CaesBG4-A-01-607
# text = neque multum frumento sed maximam partem lacte atque pecore uiuunt multumque sunt in uenationibus
1   neque   neque   CCONJ   S   Polarity=Neg    _   _   _   LiLaflcat=i
2   multum  multum  ADV M   Degree=Pos  _   _   _   LASLAVariant=2|LiLaflcat=i
3   frumento    frumentum   NOUN    A2  Case=Abl|Gender=Neut|InflClass=IndEur0|Number=Sing  _   _   _   LiLaflcat=n2
4   sed sed CCONJ   S   _   _   _   _   LiLaflcat=i
5   maximam magnus  ADJ C1  Case=Acc|Degree=Abs|Gender=Fem|InflClass=IndEurA|Number=Sing    _   _   _   LiLaflcat=n6
6   partem  pars    NOUN    A3  Case=Acc|Gender=Fem|InflClass=IndEurI|Number=Sing   _   _   _   LiLaflcat=n3
7   lacte   lac NOUN    A3  Case=Abl|Gender=Masc|InflClass=IndEurI|Number=Sing  _   _   _   LiLaflcat=n3
8   atque   atque   CCONJ   S   _   _   _   _   LASLAVariant=1|LiLaflcat=i
9   pecore  pecus   NOUN    A3  Case=Abl|Gender=Neut|InflClass=IndEurX|Number=Sing  _   _   _   LASLAVariant=1|LiLaflcat=n3
10  uiuunt  uiuo    VERB    B3  Aspect=Imp|InflClass=LatX|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin|Voice=Act   _   _   _   LiLaflcat=v3
11-12   multumque   _   _   _   _   _   _   _   _
11  multum  multum  ADV M   Degree=Pos  _   _   _   LASLAVariant=2|LiLaflcat=i
12  que que CCONJ   S   _   _   _   _   LiLaflcat=i
13  sunt    sum AUX B6  Aspect=Imp|InflClass=LatAnom|Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin  _   _   _   LASLAVariant=1|LiLaflcat=v6
14  in  in  ADP R   AdpType=Prep    _   _   _   LiLaflcat=i
15  uenationibus    uenatio NOUN    A3  Case=Abl|Gender=Fem|InflClass=IndEurX|Number=Plur   _   _   _   LiLaflcat=n3
```

Figure 2: Example of the test data format.

2.2 Test Data

Texts provided as test data for the Dependency Parsing task are by one Classical author (Seneca the Younger) and one Medieval author (Thomas Aquinas) for a total of more than 8,000 tokens. Each author is taken as a representative of a specific text genre: Seneca the Younger for poetry, more specifically tragedy, with the *Phoenissae* (more than 4,000 tokens), composed in the 1st century CE; and Thomas Aquinas for prose, more specifically philosophical-theological treatise, with a portion of the *Summa Theologiae* (more than 4,000 tokens), written in the 13th century.[7]

Data for Seneca the Younger (see Table 1) are taken from the *Opera Latina* corpus, [2] a linguistic resource manually annotated since 1961 by the Laboratoire d'Analyse Statistique des Langues Anciennes (LASLA) at the University of Liège,[8] Belgium. Original data

[7] To create a balanced corpus, we included only a portion of the *Summa Theologiae* as test data.

[8] https://www.lasla.uliege.be/cms/c_8508894/fr/lasla

were converted into the annotation formalism of the UD project, automatically parsed for dependency relations with an NLP pipeline and manually revised by an expert of Latin language and literature.

Data for Thomas Aquinas (see Table 2) are taken from *Index Thomisticum.*[9] [5]. The data are manually annotated with regard to tokenization, sentence splitting, lemmatization, PoS tagging, and syntactic annotation originally according to Prague Dependency Treebank formalism. They were then automatically converted into UD formalism and revised manually by a team of experts in Latin language.

AUTHOR	**TEXT**	**TOKENS**
Seneca	Phoenissae	4,155

Table 1: Test data for poetry.

AUTHOR	**TEXT**	**TOKENS**
Thomas Aquinas	Summa Theologiae (portion)	4,151

Table 2: Test data for prose.

[9] `https://itreebank.marginalia.it`.

Chapter 3

Task

This Section provides details on the Dependency Parsing task included in EvaLatin 2026.

Dependency Parsing is based on the UD framework. The aim of the task is to provide syntactic analysis of Latin texts. The output .conllu file provided by the participants shall have the indications of the syntactic head and of the dependency relations in the fields 7 (`HEAD`) and 8 (`DEPREL`) respectively. The set of labels for dependency relations adopted by the UD framework is as follows (relations are listed in alphabetical order):

- `acl`: a clausal modifier of a noun (`acl`) is a finite or non-finite clause that modifies a nominal.[1]
- `advcl`: an adverbial clause modifier (`advcl`) is a clause that modifies a verb or other predicate (e. g. an adjective) not introducing a core argument.[2]
- `advmod`: an adverbial modifier (`advmod`) is a non-clausal adverb or adverbial phrase that modifies a predicate or a modifier word.[3]
- `amod`: an adjectival modifier of a nominal is a phrase that serves to modify a nominal.[4]
- `appos`: an appositional modifier (`appos`) is a nominal immediately following the first noun that serves to define, modify, name, or describe that noun.[5]
- `aux`: an auxiliary of a clause (`aux`) is a function word associated with a verbal predicate that expresses categories such as tense, mood, aspect, voice or evidentiality.[6]
- `case`: the relation of case marking (`case`) is used for any case-marking element treated as a separate syntactic word (including prepositions, postpositions, and clitic case markers).[7]

[1] `https://universaldependencies.org/u/dep/acl.html`
[2] `https://universaldependencies.org/u/dep/advcl.html`
[3] `https://universaldependencies.org/u/dep/advmod.html`
[4] `https://universaldependencies.org/u/dep/amod.html`
[5] `https://universaldependencies.org/u/dep/appos.html`
[6] `https://universaldependencies.org/u/dep/aux_.html`
[7] `https://universaldependencies.org/u/dep/case.html`

- `cc`: the relation of coordinating conjunction (`cc`) holds between a conjunct and a word that links words or larger constituents without syntactically subordinating one to the other (that is, a word with PoS `CCONJ`).[8]

- `ccomp`: a clausal component (`ccomp`) is a dependent clause that functions as the object of a clause.[9]

- `clf`: a classifier (`clf`) is a function word that reflects some kind of conceptual classification of nouns, based principally on features of their referents.[10]

- `compound`: this relation is used to analyze combinations of lexemes that morphosyntactically behave as single words (that is, compounds).[11]

- `conj`: this relation holds between elements that lie syntactically on the same hierarchical level. The conjunct(s) may be connected via coordinating conjunctions or punctuation symbols.[12]

- `cop`: a copula (`cop`) is a function word used to link a subject to a nonverbal predicate.[13]

- `csubj`: a clausal subject (`csubj`) is a clausal syntactic subject of a clause.[14]

- `dep`: this relation is used when a more precise relation is impossible to determine, because of (e.g.) a weird grammatical construction, or a limitation in conversion or parsing software.[15]

- `det`: a determiner (`det`) is a function word that links a nominal head to a word that modifies nouns or noun phrases and expresses the reference of the noun phrase in context (that is, words with PoS `DET`).[16]

- `discourse`: a discourse element is an interjection or any other discourse element (intended as elements that serve to convey expressive functions).[17]

- `expl`: an expletive (`expl`) is a nominal that appears in an argument position of a predicate but does not satisfy any of the semantic roles of the predicate.[18]

[8] `https://universaldependencies.org/u/dep/cc.html`
[9] `https://universaldependencies.org/u/dep/ccomp.html`
[10] `https://universaldependencies.org/u/dep/clf.html`
[11] `https://universaldependencies.org/u/dep/compound.html`
[12] `https://universaldependencies.org/u/dep/conj.html`
[13] `https://universaldependencies.org/u/dep/cop.html`
[14] `https://universaldependencies.org/u/dep/csubj.html`
[15] `https://universaldependencies.org/u/dep/dep.html`
[16] `https://universaldependencies.org/u/dep/det.html`
[17] `https://universaldependencies.org/u/dep/discourse.html`
[18] `https://universaldependencies.org/u/dep/expl.html`

- `fixed`: a fixed multiword expression (`fixed`) is used for fixed grammaticized expressions.[19]

- `flat`: a flat expression (`flat`) combines the elements of an expression where none of the immediate components can be identified as the sole head using standard substitution tests.[20]

- `dislocated`: a dislocated element is a fronted or postposed element that does not fulfill the usual core grammatical relations of a sentence.[21]

- `goeswith`: this relation links two or more parts of a word that are separated in texts that are not well edited.[22]

- `iobj`: an indirect object (`iobj`) is a nominal that is a core argument of the verb but is not its subject or (direct) object, typically the recipient of a ditransitive verb of exchange.[23]

- `list`: this relation is used for chains of comparable items.[24]

- `mark`: a marker (`mark`) is a function word marking a clause as subordinate to another clause.[25]

- `nmod`: a nominal modifier (`nmod`) is a nominal dependent of another noun or noun phrase.[26]

- `nummod`: a numeric modifier (`nummod`) of a noun is a number phrase that serves to modify the meaning of the noun with a quantity.[27]

- `nsubj`: a nominal subject (`nsubj`) is the first core argument of a verb. It is a nominal (noun, pronoun, noun phrase) that is the syntactic subject of a clause.[28]

- `obj`: an object (`obj`) is the second most core argument of a verb after the subject. It is a nominal that denotes the entity acted upon or which undergoes a change of state or motion.[29]

[19] `https://universaldependencies.org/u/dep/fixed.html`
[20] `https://universaldependencies.org/u/dep/flat.html`
[21] `https://universaldependencies.org/u/dep/dislocated.html`
[22] `https://universaldependencies.org/u/dep/goeswith.html`
[23] `https://universaldependencies.org/u/dep/iobj.html`
[24] `https://universaldependencies.org/u/dep/list.html`
[25] `https://universaldependencies.org/u/dep/mark.html`
[26] `https://universaldependencies.org/u/dep/nmod.html`
[27] `https://universaldependencies.org/u/dep/nummod.html`
[28] `https://universaldependencies.org/u/dep/nsubj.html`
[29] `https://universaldependencies.org/u/dep/obj.html`

- `orphan`: this relation is used to mark cases of head ellipsis where simple promotion (standard treatment of ellipsis) would result in an unnatural and misleading dependency relation.[30]
- `parataxis`: this relation is used to mark a clause placed side by side without any explicit relation with the head word.[31]
- `punct`: this relation is used for any piece of punctuation in a clause.[32]
- `reparandum`: this relation indicates disfluencies overridden in a speech repair.[33]
- `root`: this relation is used to mark the root node of a sentence.[34]
- `vocative`: a vocative is the term of address used to address the interlocutor.[35]
- `xcomp`: an open clausal component (`xcomp`) is a dependent clause that a) functions as a core argument of the verb; b) does not have an own subject; c) has an external argument that determines its subject.[36]

Please refer to UD guidelines for further details on the relations subtypes.[37]

[30] `https://universaldependencies.org/u/dep/orphan.html`
[31] `https://universaldependencies.org/u/dep/parataxis.html`
[32] `https://universaldependencies.org/u/dep/punct.html`
[33] `https://universaldependencies.org/u/dep/reparandum.html`
[34] `https://universaldependencies.org/u/dep/root.html`
[35] `https://universaldependencies.org/it/dep/vocative.html`
[36] `https://universaldependencies.org/u/dep/xcomp.html`
[37] `https://universaldependencies.org/survey-deprel.html`

Chapter 4

Evaluation

Unlabeled test data are released according to the deadlines provided in the EvaLatin 2026 web page: `https://circse.github.io/LT4HALA/2026/EvaLatin`. After the assessment, the labels for the test data will be released to the participants.

4.1 Scorer

The scorer to be employed for the evaluation of the Dependency Parsing task is a revised version of one developed for the *CoNLL18 Shared Task on Multilingual Parsing from Raw Text to Universal Dependencies*.[1] The script allows for the evaluation of parsing either with or without taking deprel subtypes into account, and it is available on the EvaLatin 2026 web page: `https://github.com/CIRCSE/LT4HALA/blob/master/2026/data_and_doc/conll18_ud_eval_with_subtypes-EvaLatin2026.py`.

The command to run the script WITHOUT evaluating subtypes is the following (Python 3 required):

- `conll18_ud_eval_with_subtypes-EvaLatin2026.py -v gold_conllu_file system_conllu_file`

The command to run the script INCLUDING the evaluation of subtypes is the following (Python 3 required):

- `conll18_ud_eval_with_subtypes-EvaLatin2026.py -s -v gold_conllu_file system_conllu_file`

An example of the scorer's output is given in Figure 3. The evaluation starts by aligning the system-produced tokens to the gold standard one; given that we provide test data already sentence-splitted and annotated with morpho-grammatical information, the alignment for tokens, sentences, words, UPOS, UFeats and lemmas should be perfect

[1] `https://universaldependencies.org/conll18/evaluation.html`

(i. e. 100.00). Then, CLAS[2] and LAS[3] are evaluated.[4] Each metric is evaluated in terms of Precision, Recall, F1 and Aligned Accuracy.[5]

We will evaluate dependency relations with and without subtypes separately, e.g., `advcl:abs` (ablativus absolutus/ablative absolute) and `advcl` (adverbial clause modifier), providing two separate rankings. However, the use of subtypes is not mandatory: participants who do not use subtypes will not be penalized but they will still be evaluated for dependency relations without subtypes.

Metric	Precision	Recall	F1 Score	AligndAcc
Tokens	100.00	100.00	100.00	
Sentences	100.00	100.00	100.00	
Words	100.00	100.00	100.00	
UPOS	100.00	100.00	100.00	100.00
XPOS	100.00	100.00	100.00	100.00
UFeats	100.00	100.00	100.00	100.00
AllTags	100.00	100.00	100.00	100.00
Lemmas	100.00	100.00	100.00	100.00
UAS	96.67	96.67	96.67	96.67
LAS	80.00	80.00	80.00	80.00
CLAS	80.00	70.59	75.00	70.59
MLAS	66.67	58.82	62.50	58.82
BLEX	80.00	70.59	75.00	70.59

Figure 3: Example of the scorer's output for the Dependency Parsing task.

4.2 Baselines

As a baseline, we provide the scores obtained on the test data using UDPipe 2 [9] with the models trained on the Universal Dependencies Latin CIRCSE Treebank,[6] v. 2.17,

[2]Content-Word Labeled Attachment Score (CLAS) is the labeled F1- score over all relations except relations for function words (`aux`, `case`, `cc`, `clf`, `cop`, `det`, `mark`) and for punctuation (`punct`). For greater details, see [4].

[3]Labeled Attachment Score (LAS) is the percentage of tokens assigned both the correct DEPREL and HEAD. For further details, see [1].

[4]As shown in Figure 3, the scorer computes also the Unlabeled Attachment Score (UAS), that is the percentage of tokens assigned the correct HEAD; the Morphology-aware Labeled Attachment Score (MLAS), that is CLAS extended with evaluation of POS tags and morphological features; the Bi-Lexical dependency score (BLEX) that combines content-word relations with lemmatization, but not with POS tags and features. These 3 metrics are not taken into account for this shared task.

[5]Precision (P) is the number of the correct values divided by the number of the values produced by the system. Recall (R) is the number of correct values divided by the number of the values of the gold standard. F1 score is the harmonic mean of P and R. Aligned Accuracy is the Accuracy computed on the data aligned with the values of the gold standard; for greater details on this, see `http://universaldependencies.org/conll17/evaluation.html`

[6]`https://github.com/UniversalDependencies/UD_Latin-CIRCSE`

and the Universal Dependencies Latin *Index Thomisticus* Treebank (IT-TB), [7] v. 2.17, respectively, as they are available from the tool's web interface.[8] Baseline results are given in Tables 3 and 4 for poetry, and 5 and 6 for prose.

Metric	**Precision**	**Recall**	**F1 Score**	**AligndAcc**
CLAS	56.14	57.76	56.94	57.76
LAS	57.22	57.22	57.22	57.22

Table 3: Baseline results on poetry, with subtypes.

Metric	**Precision**	**Recall**	**F1 Score**	**AligndAcc**
CLAS	57.29	57.19	57.24	57.19
LAS	59.74	59.74	59.74	59.74

Table 4: Baseline results on poetry, no subtypes.

Metric	**Precision**	**Recall**	**F1 Score**	**AligndAcc**
CLAS	80.33	78.51	79.41	78.51
LAS	82.17	82.17	82.17	82.17

Table 5: Baseline results on prose, with subtypes.

Metric	**Precision**	**Recall**	**F1 Score**	**AligndAcc**
CLAS	83.00	83.14	83.07	83.14
LAS	85.21	85.21	85.21	85.21

Table 6: Baseline results on prose, no subtypes.

[7] `https://universaldependencies.org/treebanks/la_ittb`
[8] `http://lindat.mff.cuni.cz/services/udpipe/`

Chapter 5

How to Participate

Participants are required to submit their runs and to provide a technical report describing their systems.

5.1 Submitting Runs

Each participant can submit a maximum of 2 runs for each task. Participants are allowed to use any approach (e.g. from traditional machine learning algorithms to LLMs) and any resource (annotated and non-annotated data, embeddings): all approaches and resources are expected to be described in the system's report.

Once the system has produced the results for the task over the test set, participants have to follow these instructions to complete their submissions:

- name the runs with the following filename format: `task_docName_teamName_runID.conllu`.
  For example: `parsing_XXX_unicatt_1.conllu` would be the first run of a team called *unicatt* for the dependency parsing task on the `XXX.conllu` document.

- send the file to the following email address: `federica.iurescia[AT]unicatt.it`, using the subject "EvaLatin Submission: task - teamName", where the "task" is *Parsing*.

5.2 Writing the Technical Report

Technical reports will be included in the proceedings of the Fourth Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA 2026) as short papers and will be published along the LREC 2026 proceedings. Reports must be submitted through the START platform (`https://softconf.com/lrec2026/LT4HALA2026/`). All reports must meet the following requirements:

- they must be written in English;

- they must be formatted according to the LREC 2026 conference style[1];
- the maximum length is 4 pages (excluding references);
- they should contain (at least) the following sections: description of the system, results, discussion, references.

Reports will receive a light review: we will check for the correctness of the format, the exactness of results and ranking, and overall exposition. If needed, we will contact the authors asking for corrections.

[1] https://lrec2026.info/authors-kit/

Appendix A

Selection of Resources for Latin

- Lemma embeddings: https://embeddings.lila-erc.eu/
- Latin texts and embeddings: http://www.cs.cmu.edu/~dbamman/latin.html
- Latin BERT: https://github.com/dbamman/latin-bert
- Word embeddings: https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1989
- CLTK: http://cltk.org/
- UD Latin PROIEL: https://github.com/UniversalDependencies/UD_Latin-PROIEL
- UD Latin ITTB: https://github.com/UniversalDependencies/UD_Latin-ITTB
- UD Latin Perseus: https://github.com/UniversalDependencies/UD_Latin-Perseus
- UD Latin LLCT: https://github.com/UniversalDependencies/UD_Latin-LLCT
- UD Latin UDante: https://github.com/UniversalDependencies/UD_Latin-UDante
- UD Latin CIRCSE: https://github.com/UniversalDependencies/UD_Latin-CIRCSE
- Latin texts: https://github.com/PerseusDL
- Collatinus: https://outils.biblissima.fr/en/collatinus/index.php
- LEMLAT v.3: https://github.com/CIRCSE/LEMLAT3
- Treetagger: https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
- Glossaria: https://glossaria.eu/outils/lemmatisation/#page-content

- Word Formation Latin (WFL) lexicon: http://wfl.marginalia.it/
- Lemmatized corpus with morphological analysis "Corpus Latin antiquité et antiquité tardive lemmatisé": https://doi.org/10.5281/zenodo.4337145
- Latin Lasla Model (Thibault Clérice) for lemmatization and morphological analysis: https://doi.org/10.5281/zenodo.3773327
- LatinCy: https://spacy.io/universe/project/latincy
- LatinAffectus v4: https://github.com/CIRCSE/LT4HALA/blob/master/2024/LatinAffectusv4.tsv
- Deucalion: https://dh.chartes.psl.eu/deucalion/latin
- UDPipe 1 Models: https://ufal.mff.cuni.cz/udpipe/1/models
- UDPipe 2 Models: https://ufal.mff.cuni.cz/udpipe/2/models
- Latin Macronizer: https://alatius.com/macronizer/
- LatinISE: https://lindat.mff.cuni.cz/repository/items/6078be62-56b9-4a42-99e5-766c3b2e37df
- LiLa resources: https://lila-erc.eu/data-page/

Bibliography

[1] Sabine Buchholz and Erwin Marsi. CoNLL-X shared task on multilingual dependency parsing. In Lluís Màrquez and Dan Klein, editors, *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June 2006. Association for Computational Linguistics.

[2] Joseph Denooz. Opera Latina : une base de données sur internet. *Euphrosyne*, 32:79–88, 2004. Retrievable at `https://www.brepolsonline.net/doi/10.1484/J.EUPHR.5.125535`.

[3] Federica Gamba and Daniel Zeman. Universalising Latin Universal Dependencies: a harmonisation of Latin treebanks in UD. In Loïc Grobol and Francis Tyers, editors, *Proceedings of the Sixth Workshop on Universal Dependencies (UDW, GURT/SyntaxFest 2023)*, pages 7–16, Washington, D.C., March 2023. Association for Computational Linguistics.

[4] Joakim Nivre and Chiao-Ting Fang. Universal Dependency evaluation. In Marie-Catherine de Marneffe, Joakim Nivre, and Sebastian Schuster, editors, *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden, May 2017. Association for Computational Linguistics.

[5] Marco Passarotti. *The Project of the Index Thomisticus Treebank*, pages 299–320. Number 10 in Age of Access? Grundfragen der Informationsgesellschaft. De Gruyter Saur, Berlin, Germany; Boston, MA, USA, 2019.

[6] Rachele Sprugnoli, Federica Iurescia, and Marco Passarotti. Overview of the EvaLatin 2024 evaluation campaign. In Rachele Sprugnoli and Marco Passarotti, editors, *Proceedings of the Third Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA) @ LREC-COLING-2024*, pages 190–197, Torino, Italia, May 2024. ELRA and ICCL.

[7] Rachele Sprugnoli, Marco Passarotti, Flavio Massimiliano Cecchini, and Matteo Pellegrini. Overview of the EvaLatin 2020 evaluation campaign. In *Proceedings of LT4HALA 2020 - 1st Workshop on Language Technologies for Historical and Ancient Languages*, pages 105–110, Marseille, France, May 2020. European Language Resources Association (ELRA). Retrievable at `https://www.aclweb.org/anthology/2020.lt4hala-1.16`.

[8] Rachele Sprugnoli, Marco Passarotti, Cecchini Flavio Massimiliano, Margherita Fantoli, and Giovanni Moretti. Overview of the evalatin 2022 evaluation campaign. In *Proceedings of the Second Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA 2022), Language Resources and Evaluation Conference (LREC 2022)*, pages 183–188, 2022.

[9] Milan Straka, Jan Hajič, and Jana Straková. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). Retrievable at `https://aclanthology.org/L16-1680`.