ScanLat:
Grammatical Prosody Scansion for Classical Latin Poetry

Matthew A. Penza*
Department of Computer Science
Princeton University
Princeton, NJ
mpenza@cs.princeton.edu

Christopher M. Ferri
Department of Computer Science
Princeton University
Princeton, NJ
cferri@princeton.edu

***Abstract*—This paper describes ScanLat, a new tool for automatic prosody scansion of classical Latin poetry. Using a combination of dependency parsing and morphological analysis, ScanLat is able to determine the syllabic quantities of a poetic text and output the same text annotated with long syllables marked with macrons. No quantitative metric exists to assess the accuracy of ScanLat, but qualitatively we conclude that it is a success, albeit fallible. We recommend ScanLat as a baseline tool, whose results a precision-conscious reader should verify and fine-tune.**

Keywords—poetry, prosody scansion, meter, syllable quantity, dependency parsing, morphological analysis, digital humanities

INTRODUCTION

An authentic reading of classical Latin poetry depends on proper scansion of its prosody: the determination of the quantities, or lengths, of each syllable. Historically, scansion was and still is done primarily with pen and paper, with the results reliant upon the reader's knowledge of Latin grammar, the rules of prosody scansion, and the peculiarities of individual poets and texts. To make this process less tedious, we have developed ScanLat, a tool that automatically scans Latin poetry, marking the long syllables with macrons as is customary.

Other tools for this task already exist, most notably the Latin prosody module of the Classical Languages Toolkit (CLTK). However, ScanLat leverages grammatical information from the text to resolve ambiguous syllable quantities, whereas CLTK ignores this information in favor of a strictly probabilistic approach to determining syllable quantities.

In this paper, we will: (1) explain the fundamentals of prosody scansion for classical Latin poetry; (2) detail the methods used in ScanLat to accomplish this task; (3) explain the intended usage of the tool; (4) present and qualitatively evaluate ScanLat's results on several sample texts; (5) propose improvements and extensions for future versions; and (6) detail our conclusions regarding the success of the tool and our recommendations for its use.

I. BACKGROUND

A. Prosody

The meter of classical Latin poetry is based primarily on syllable quantities, quite unlike English and even later Latin poetry, which are based on stress accents.

Each syllable is said to be either one or two *morae* long: intuitively, a short syllable is one *mora*, while a long syllable is two *morae*. A verse of poetry is broken into *feet*, each of which is either three or four *morae*. The word *carō*, for example, is an iamb: three *morae* long, consisting of a short syllable followed by a long one. The word *fātō*, on the other hand, is a spondee: four morae long, consisting of two long syllables. These examples are whole words, but feet can span across parts of multiple words: consider this phrase from the first line of the *Aeneid* of Virgil*: prīmus ab ōrīs*. It is a dactyl (*prīmus ab*: one long, two shorts) followed by a spondee (*ōrīs*).

In order to properly read a poetic text—classical Latin poetry was historically always recited aloud, never read silently—it is essential to determine the syllable quantities. Fortunately for classicists, there are rules for this; it is not arbitrary. A syllable is long *by nature* if:

- It always has a long vowel: e.g. the *i* in *sīcut* is always long; or
- It has a diphthong, i.e. any of the combinations *ae*, *au*, *ei*, *eu*, or *oe*, or *ui* in the words *huius*, *cuius*, *huic*, *cui*, and *hui* only.

A syllable is long *by position* if:

- It ends in two consonants, or either of the compound consonants *x* and *z*: e.g. the *a* in *canto* and the *u* in *lux* are both long by position; or
- It ends in one or more consonants and is followed by a syllable that begins with a consonant: e.g. the first *u* in *virumque* is long by position (*qu* is counted as a single consonant); or
- It is *brevis in longo*, or the final syllable of a line of verse, which is always long by position.

If a syllable is neither long by nature nor long by position, then by default it is short. There are no other criteria that coerce a syllable to be short.

B. Syllabification

Before syllable quantities can be determined, the syllables themselves must be extracted from the text. Fortunately, again, there are rules for this:

- Consecutive single vowels are separated: e.g. *be-a-tus*;
- Single vowels are separated from diphthongs and vice versa; e.g. *me-ae* or *Car-thae-am*;
- When one or more consonant separates vowels or diphthongs, generally the last consonant belongs to the second syllable, and the consonants preceding it belong to the first: e.g. *mul-tum*; and
- Exceptions to the above: combinations of plosives (*p*, *b*, *t*, *d*, *c*, and *g*) and liquids (*l* and *r*), as well as *qu* and the Greek-type aspirates *ch*, *ph*, and *th*, each count as a single consonant and should not be separated; e.g. *vi-rum-que*, *pa-tris*, and *ca-the-dra*.

II. Specifications

A. Dependencies

ScanLat depends on the following:

- Python 2.7+ (***not*** Python 3)
- Stuttgart Finite State Transducer (SFST) 1.4.7d+
- LatMor
- UDPipe 1.2+
- Universal Dependencies (UD) 2.0

LatMor is a finite-state morphological analysis tool that can analyze and generate Latin wordforms, both macronized and unmacronized. It is the latter function that is of primary concern for ScanLat's purposes. LatMor's finite-state transducers must be read using SFST.

The UD models are trained dependency parsers for over one hundred languages. ScanLat is built to use Latin PROIEL model. It may be compatible with other models, but this has not been tested. UDPipe is a Python wrapper library that allows access to the UD models.

B. Limitations

ScanLat is primarily limited by the accuracy of the UD 2.0 models with UDPipe, and that of LatMor.

In some tests, even with syntactically valid input, LatMor has encountered words that it could not analyze at all. Such cases are generally confined to proper names, borrowed words (such as from Hebrew in medieval and ecclesiastical texts), and rare words. If LatMor cannot analyze a word, ScanLat outputs an unmacronized copy of the word, prefixed with the flag "ERROR_" to alert the reader that the word has not been properly scanned. E.g., *rudentum*, a variant form of the more usual genitive plural *rudentium* of the noun *rudens, -entis* (meaning *rope* in English) has no entry in LatMor and is therefore output as "ERROR_Rudentum".

A final note of caution: be sure that there are no blank lines at the beginning of the input text document, or else ScanLat will crash for reasons yet to be determined.

C. Usage

To use ScanLat, first install Python 2, UDPipe, and SFST, and decompress `ScanLat.tar.bz2`. (NB: the requisite files for LatMor and the UD 2.0 models are included therein; no further installations are required.) Then place into the resulting directory `ScanLat` as many `.txt` files as desired, each containing text to be scanned. Use `cd` to navigate in the Terminal to the `ScanLat` directory.

Now suppose `input.txt` is to be scanned. There are three possible ways to handle it:

1. If it contains text *without macrons* (other diacritics are acceptable), then run `bash scanlat.sh input.txt` in the Terminal. This will supply first the natural vowel lengths, and then the positional vowel lengths.

2. If it contains text *with macrons* whose reliability as natural vowel quantities is *in doubt*, then run `bash scanlat.sh input.txt` in the Terminal. This will clean the input text of its macrons, resupply the natural vowel lengths, and then supply the positional vowel lengths, as if the original input had no macrons.

3. If it contains text *with macrons* that are known to be *reliably annotated* natural vowel quantities, then run `bash scanlat.sh input.txt m` in the Terminal, where `m` stands for *macronized*. (NB: any second argument will work; the use of `m` in particular is merely a suggestion. E.g., `bash scanlat.sh input.txt yes` functions identically.) This will rely on the macrons already in the text to serve as the natural vowel lengths and will only supply the positional vowel lengths.

The scanned text will be found in `ScanLat/output/scanned_input.txt`. A log of status updates and error messages, primarily intended for developer use, will be produced as well at `ScanLat/output/log_input.txt`.

III. Methods

A. Text Cleaning

ScanLat's first step is to clean up the input text, removing extraneous punctuation, numerical characters, and extra white space. ScanLat can handle input text that has stress accent marks such as *ú*, diaereses such as *ë*, and the diphthong ligatures *æ*, *œ*, *Æ*, and *Œ*, replacing each with its plaintext equivalent. It also replaces all *j* and *J* characters with *i* and *I*, respectively, for the sake of consistency. All told, it makes for a robust tool.

B. Determining Natural Syllable Quantities

1) Dependency Parsing

In order to determine the vowels in a word that are long by nature, ScanLat uses the UD 2.0 models in UDPipe to create a dependency parse of the text, along with corresponding feature tagging. These features include both basic parts of speech (noun, verb, etc.) and more detailed information specific to each part of speech. E.g. finite verbs' tags include mood and tense, nouns have gender, case, and number, adjectives and adverbs have degree, etc.

These features help resolve ambiguities between homographs that differ only by vowel length. E.g. the nominative and vocative of the word for *girl* are both *puella*, whereas the ablative of the same word is *puellā*: the only difference is in the length of the final *a*. In this instance, knowing the case is key to macronizing the word properly.

2) Morphological Analysis

After the features of a word are acquired from UDPipe, they are transformed into LatMor-readable form and fed in. If this initial attempt is successful, then LatMor's output becomes the macronized form of the word.

While UDPipe generally produces an accurate, or at least reasonable, set of features for a given word, it does regularly output spurious results. These cannot be ported directly to LatMor either because they are incomplete or nonsensical. E.g., consider the word *scuta*, which is a second-declension neuter plural **noun** in the nominative, accusative, or vocative case,

meaning *shield* in English. In one run of ScanLat, UDPipe correctly produced the lemma *scutum* but labelled it a second-person singular present active imperative **verb**. In cases such as these, when LatMor fails to provide a valid result for the invalid input, ScanLat attempts to find a "default" macronization, feeding LatMor the word alone (rather than with syntactic information) to generate a list of possible outputs. Then it narrows this list down to only the plausible choices and chooses from among those arbitrarily. In particular, it chooses the first plausible choice in the list of possible outputs. In this context, a "plausible" choice is a macronized form whose demacronization matches the original form of the word. E.g. for an original form *spiritus*, there are plausible choices *spīritus* and *spīritūs*.

C. Determining Positional Syllable Quantities

With the natural syllable quantities accounted for, ScanLat then finds the syllables that are long by position. To do this, it traverses the text line by line, word by word, syllable by syllable, applying the rules described in I.A. (Background: Prosody) and I.B. (Background: Syllabification).

Once the positional quantities are determined, ScanLat outputs the final scanned text, with line numbers every fifth line for the reader's convenience.

IV. Results

ScanLat works well overall, qualitatively speaking. Due to the lack of a reliably scanned corpus of classical Latin poetry against which to compare, there is unfortunately no quantitative measure of success available. However, we will present ScanLat's output from a few inputs, with run times. While we are generally pleased with the output, the tool's real-time performance is rather slow.

Table 1: ScanLat Performance Times

Text	Real time	User time	Sys. time
Aeneid 1-101 (plain input)	2m53.116s	2m29.705s	0m17.966s
Aeneid 1-101 (naturally long vowels marked in input)	0m0.096s	0m0.063s	0m0.026s
Introit (plain input, 52 lines)	1m3.568s	0m55.074s	0m6.504s

The first 22 lines of Virgil's *Aeneid*, using plain input and UDPipe and LatMor to determine naturally long vowels:

```
    ārmā virūmque cānō , Trōiae quī prīmus ab ōrīs
    Ītaliām fātō profugūs Lāvīniaque vēnīt
    lītora , mūltum īlle ēt tērrīs iāctātus et āltō
    vī superūm , saevae memorēm Iūnōnis ob īrām ,
5   mūlta quoque ēt bēllō pāssūs , dūm cōnderet ūrbēm
    īnfĕrrētque deōs lātiō ; genus ūnde Latīnūm
    Ālbānīque patrēs ātque āltae moenia Rōmae .
    Mūsa , mihī causās memorā , quō nūmine laesō
    quīdve dolēns rēgīna deum tōt vōlvere cāsūs
10  īnsīgnēm pietāte virūm , tot adīre labōrēs
    īmpulerīt . tāntaene animīs caelēstibus īrae ?
    urbs āntīquā fuīt Tyriī tenuēre colōnī
    ERROR_Karthago , Ītaliām cōntrā Tiberīnaque lōngē
    ōstia , dīves opūm studiīsque āspērrima bēllī ,
15  quām Iūnō fĕrtūr tērrīs magīs ōmnibus ūnām
    pōsthabita coluīsse Samō . hīc īllīus ārmā ,
    hīc cūrrūs fuīt ; hōc rēgnūm dea gēntibus ēssē ,
    sī quā fāta sinānt , iām tūm tēndītque fovētquē .
    prōgeniēm sed enīm Trōiānō ā sānguine dūcī
20  audierāt Tyriās ōlīm quae vērteret ārcēs ;
    hīnc populūm lātē rēgēm bēllōque supērbūm
    vēntūrum ēxcidiō Libyae ; sīc vōlvere pārcās .
```

The same excerpt, but using input with the natural vowels already included, and not relying on UDPipe or LatMor to determine naturally long vowels. Note the similarities and differences, especially the lack of "ERROR_"-tagged words:

```
    Ārma virūmque canō , Trōiae quī prīmus ab ōrīs
    Ītaliām fātō profugūs Lāvīniaque vēnīt
    lītora , mūltum īlle ēt tērrīs iāctātus et āltō
    vī superūm , saevae memorēm Iūnōnis ob īrām ,
5   mūlta quoque ēt bēllō pāssūs , dūm cōnderet ūrbēm
    īnfĕrrētque deōs Latiō ; genus ūnde Latīnūm
    Ālbānīque patrēs ātque āltae moenia Rōmae .
    Mūsa , mihī causās memorā , quō nūmine laesō
    quīdve dolēns rēgīna deum tōt vōlvere cāsūs
10  īnsīgnēm pietāte virūm , tot adīre labōrēs
    īmpulerīt . Tāntaene animīs caelēstibus īrae ?
    Urbs āntīqua fuīt Tyriī tenuēre colōnī
    Kārthāgō , Ītaliām cōntrā Tiberīnaque lōngē
    ōstia , dīves opūm studiīsque āspērrima bēllī ,
15  quām Iūnō fĕrtūr tērrīs magis ōmnibus ūnām
    pōsthabitā coluīsse Samō . hīc īllius ārmā ,
    hīc cūrrūs fuīt ; hōc rēgnūm dea gēntibus ēssē ,
    sī quā Fāta sinānt , iām tūm tēndītque fovētquē .
    Prōgeniēm sed enīm Trōiānō ā sānguine dūcī
20  audierāt Tyriās ōlīm quae vērteret ārcēs ;
    hīnc populūm lātē rēgēm bēllōque supērbūm
    vēntūrum ēxcidiō Libyae ; sīc vōlvere Pārcās .
```

Excerpts from Introit of the 1962 Roman Missal (serving as arbitrary Latin text; this is not classical poetry by any means):

```
    īn nōmine patris , ēt fīliī , ēt spīritūs sānctī . āmēn .
    īntroībō ad āltāre deī .
    ād deum quī laetificāt iuvēntūtēm meām .
```

```
      iūdicā mē , deus , ēt dīscērne causām meām dē gēnte
nōn sāncta : āb homine inīquō ēt dolōsō ērue mē .
5     quia tū ēs , deus , fortitūdō meā : quārē mē rēppulīstī , ēt
quārē trīstis īncēdō , dum āfflīgīt mē inimīcūs ?
      ēmītte lūcēm tuam ēt vēritātēm tuam : īpsa mē
dēdūxērūnt et āddūxērūnt īn mōntēm sānctūm tuum , et īn
tabērnācula tuā .
      et īntroībō ad āltāre deī : ād deum quī laetificāt
iuvēntūtēm meām .
      cōnfitēbōr tibī īn cithara , deus , deus meus : quārē
trīstis es anima meā , ēt quārē cōntūrbās mē ?
      spērā īn deō , quoniam ādhūc cōnfitēbor īllī : salūtāre
vūltūs mēī , ēt deus meus .
10    glōria patrī , ēt fīliō , ēt spīrituī sānctō .
      sīcut erat īn prīncipiō , ēt nūnc , ēt sēmper : et īn saecula
saeculōrum . āmēn .
      īntroībō ad āltāre deī .
      ād deum quī laetificāt iuvēntūtēm meām .
      adiūtōriūm nōstrum īn nōmine dominī .
15    quī fēcīt caelum ēt tērrām .
      cōnfiteōr deō ōmnipotēntī ,
      beātae Mariae sēmpēr vīrginī ,
      beātō Michāēlī ārchāngelō ,
      beātō Iōānnī bāptīstae ,
20    sānctīs apōstolīs petrō ēt paulō ,
      ōmnibūs sānctīs ,
      ēt vōbīs , frātrēs :
      quia pēccāvī nimīs cōgitātiōne , vērbō , et operē :
      meā cūlpā , meā cūlpā , meā māximā cūlpā .
25    ideō precōr beātām Mariām sēmpēr vīrginēm ,
      beātūm Michāēlem ārchāngelūm , beātūm Iōānnēm
bāptīstām ,
      sānctōs apōstolōs ERROR_Petrum ēt paulūm ,
      ōmnēs sānctōs ,
      ēt vōs , frātrēs ,
30    ōrāre prō mē ād dominūm deum nōstrūm .
      misereātūr tuī ōmnipotēns deus , ēt dīmīssīs pēccātīs
tuīs , pērdūcāt tē ād vītam aetērnām .
      āmēn
      glōria in ēxcēlsīs deō
      et īn tērrā pāx hominibūs bonae volūntātīs .
35    laudāmūs tē . benedīcimūs tē .
      adōrāmūs tē . glōrificāmūs tē .
      grātiās agimūs tibī prōptēr māgnām glōriām tuām .
      domine deus , rēx ERROR_Coelestis , deus pater
ōmnipotēns .
      domine fīlī ūnigenitē , iēsū chrīstē .
40    domine deus , āgnūs deī , fīliūs patrīs .
      quī tōllīs pēccāta mūndī , miserēre nōbīs .
      quī tōllīs pēccāta mūndī , sūscipe dēprecātiōnēm
nōstrām .
      quī sēdēs ād dēxterām patrīs , miserēre nōbīs .
      quoniām tū sōlūs sānctūs .
45    tū sōlūs dominūs .
      tū sōlus āltīssimūs , iēsū chrīstē .
      cūm sānctō spīritū īn glōriā deī patris . āmēn .
```

By manual inspection, these results are mostly accurate. Ideally a direct comparison between these ScanLat results and manually scanned text would be the best assessment of quality, but due the lack of availability of a scanned corpus and time constraints preventing us from generating our own scans, this is not possible at this time. We do know, however, that ScanLat scans *perfectly* the first four lines of the *Aeneid* ("*Arma virumque cano … Iunonis ob iram*").

V. Future Work

Initially we had hoped to produce output visually similar to that pictured below in Figure 1, which macronizes only the naturally long vowels and denotes syllable length with positional quantities accounted for in a separate line above each line of text. Due to time constraints, we had to settle for marking the long vowels directly with macrons. In a future version, we would like to implement this more standard notation.

Figure 1: A Typical Scansion Markup

```
-  u  u| -   u  u| -||  -| -    -|  - u  u |- -
Ārmă vĭ-rŭmquĕ că-nō, Trō-iae quī prīmŭs ăb ōrīs
- u u|-   -| - ||  u u| -   -| -  u  u| - -
Ĭtălĭ-ăm fā-tō   prŏfŭ-gŭs Lā-vīniăquĕ vēnĭt
 - u u | -       - |     -   - | - || - | - u  u |-  -
lĭtōră, mūlt(um) īl-l(e) ēt tĕr-rīs  iăc-tātŭs ēt ăltō
 -  u u| - ||  - | -   u u| -   -| - u  u |- -
vī sŭpĕ-rūm,  sae-vae mĕmŏ-rĕm Iū-nōnĭs ōb īrăm;
```

Under certain circumstances, adjacent syllables are elided. Namely, if a word-final vowel is followed by a word-initial vowel or *h*, then the former is elided into the latter and does not count as a syllable. Word-final combinations of a vowel with *m* followed by a word-initial vowel or *h* behave similarly. E.g. *multum ille et* sounds as *mult'ill'et*. ScanLat does not currently account for elision, though it would be possible to account for it in a future version.

ScanLat currently does not interact with poetic meter beyond the simple syllable quantities. A future version would do well to expand its functionality to check whether a text fits a given meter. This would amount to fairly simple pattern-matching. Once implemented, an easy corollary to such an extension would be to have ScanLat to attempt to identify the meter of a text by simply iterating through the checks.

Finally, when running with natural vowel lengths dependent on UDPipe and LatMor, ScanLat is noticeably slow, with a nearly three-minute real run time on a 100-line input. This is at least in part because each time ScanLat invokes LatMor, LatMor has to re-load the transducer, which takes about a second each time. LatMor does have a batch mode that avoids re-reading the transducer for every input. A high priority for a

future version of ScanLat would be to exploit that feature.

VI. Conclusion

At this time, we unfortunately have no reliable quantitative metric to evaluate ScanLat's results, owing to the lack of corpora of reliably scanned poetic texts against which to compare. We do, however, conclude qualitatively that ScanLat is a success, albeit fallible. We recommend ScanLat as a baseline tool, whose results an attentive reader should verify and fine-tune.

Acknowledgements

We extend our sincerest thanks to:

Prof. Helmut Schmid of the Center for Information and Language Processing at the Ludwig Maximilian University Munich and ***Prof. Uwe Springmann*** of Universität Würzburg, for their enthusiastic help in navigating and resolving issues that arose with **LatMor** during development;

Prof. Francis Tyers of the Russian National Research University Higher School of Economics, for recommending **UDPipe** to us;

Prof. Gregory Crane, Editor-in-Chief of the Perseus Project at Tufts University, for recommending **LatMor** to us; and

Prof. Laura Janda of the University of Tromsø—The Arctic University of Norway, currently a Visiting Research Scholar at Princeton University, for putting us in contact with Prof. Tyers.

References

[1] Wheelock, Martha, Deborah Wheelock Taylor, and Richard A. LaFleur. *Wheelock's Latin.* Introduction. Syllables and Syllable Quantity. Harper Collins. http://wheelockslatin.com/chapters/introduction/introduction_syllables.html

[2] Wheelock, Martha, Deborah Wheelock Taylor, and Richard A. LaFleur. *Wheelock's Latin.* Introduction. Diphthongs. Harper Collins. http://wheelockslatin.com/chapters/introduction/introduction_diphthongs.html

[3] Same as (1).

[4] SFST 1.4.7d. Schmid, Helmut. Available for download and installation at http://www.cis.uni-muenchen.de/~schmid/tools/SFST/.

[5] LatMor. Springmann, Uwe, Dietmar Najock, and Helmut Schmid. Available for download at http://www.cis.uni-muenchen.de/~schmid/tools/LatMor/.

[6] Straka, Milan, and Jana Straková. UDPipe 1.2.0.1. Install from PyPi with `pip install ufal.udpipe`.

[7] Straka, Milan and Jana Straková, 2017, Universal Dependencies 2.0 Models for UDPipe (2017-08-01), LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University, http://hdl.handle.net/11234/1-2364.

[8] Virgil. *The Aeneid*. Ed. Christopher Francese and Meghan Reedy, *Vergil: Aeneid Selections*. Carlisle, Pennsylvania: Dickinson College Commentaries, 2016. ISBN: 978-1-947822-08-5. http://dcc.dickinson.edu/vergil-aeneid/vergil-aeneid-i-1-11.