Umanistica Digitale – ISSN: 2532-8816 – n. 23, 2026: Special Issue
P. Vervek – *Collatinus*
DOI: http://doi.org/10.60923/issn.2532-8816/23866

Collatinus

A swiss-knife for Latin

Philippe Verkerk

Université de Lille Bât
Philippe.Verkerk@univ-lille.fr

Abstract

Created by Yves Ouvrard, *Collatinus* is a lemmatizer and a morphological analyzer for Latin. Its initial aim was to help teachers prepare texts together with their vocabulary list, and to assist beginners in reading authentic Latin texts independently. *Collatinus* relies on a lexical base of more than 80,000 lemmata and on tables of word-endings. The lexical base includes a short translation of the lemmata, mainly in English and in French and the software allows users to consult dictionaries, both digital and image-based, in order to access full lexical information. *Collatinus*' ability to split an inflected form in its stem and ending enables it to provide its full morphological analysis. As the quantities of each syllable are given in its databases, *Collatinus* is also able to scan texts metrically or to indicate accentuation. Several external tools have been developed to extend these core possibilities. More recently, *Collatinus* has been coupled to AI techniques (Latin-BERT) to choose the more appropriate lemmatization and analysis for each word in context. The AI model was trained on the annotated LASLA corpus, and the resulting tagger produces output files in the same APN format used by LASLA.

Keywords: Latin; Morphological Analysis; Lemmatization; Artificial Intelligence

Collatinus *è un lemmatizzatore e un analizzatore morfologico per il latino sviluppato da Yves Ouvrard. In origine era pensato per aiutare gli insegnanti a preparare testi con elenchi di vocaboli e per permettere ai principianti di leggere autonomamente testi latini.* Collatinus *si fonda su una base lessicale di oltre 80.000 lemmi e su tabelle di flessione. La base lessicale include una traduzione base dei lemmi, principalmente in inglese e in francese, e il programma offre la possibilità di consultare dizionari, sia in formato digitale che di immagine. La capacità di dividere una parola flessa nella radice e nella desinenza consente a* Collatinus *di fornire l'analisi morfologica completa. Poiché la quantità di ciascuna sillaba è indicata nelle sue basi di dati,* Collatinus *è in grado di fare la scansione di un testo o di indicarne l'accentuazione. Sono stati sviluppati alcuni strumenti esterni aggiuntivi per estendere queste possibilità di base.* Collatinus *è stato inoltre potenziato con l'intelligenza artificiale (Latin-BERT) affinché scelga la migliore lemmatizzazione ed analisi per ogni parola di un testo, tenendo conto del contesto. L'intelligenza artificiale è stata addestrata sul* corpus *annotato dal LASLA e il* tagger *produce una file nello stesso formato APN utilizzato nel* corpus *LASLA.*

Parole chiave: Latino; Analisi morfologica; Intelligenza artificiale

1. Introduction

Collatinus is a free and open-source software initially developed by Yves Ouvrard in the 90's. It is a lemmatizer and morphological analyzer for Latin [5]. For any inflected form, it tries to split it into its stem and its ending, as any student does. The system relies on a lexical database containing more than 82,000 lemmata, each associated with a paradigm. Paradigms are linked to tables of word-endings, which are connected to the morphosyntactic features. This information can be used to find all the possible inflected forms of a lemma or, conversely, to identify all the possible lemmata that may account for a given form.

The lexical database also includes short translations of the lemmata. To provide fuller lexical information, *Collatinus* allows users to consult integrated dictionaries. These may be digital ones –such as the *Lewis & Short*, converted in XML by the *Perseus* project, or the *Gaffiot*, digitalized by Gérard Gréco and his team– or dictionaries displayed as images. This last possibility is particularly useful for target languages that are less widely spread than English, French or German.

Collatinus was originally designed for teaching purposes. It helps teachers prepare a text with its vocabulary list to be distributed to pupils or students. On the other hand, it assists beginners in reading authentic Latin texts on their own. Around 2010, syllable quantities have been added to both stems and word-endings. This breakthrough enables *Collatinus* to scan individual word and, consequently, entire texts. For medieval Latin, where rhythmic accentuation is derived from the syllable quantities, *Collatinus* can identify the stressed and unstressed syllables within a text. This feature opens the way to studies of metrical poetry or rhythmic prose, as discussed below.

Although most of *Collatinus*' features are available online,[1] I will focus on the offline version, which offers more options, particularly for research. This offline version can be freely downloaded as easy-to-install packages for Mac OS or Windows.[2] Unfortunately, it was not possible to support all Linux distributions: Linux users must therefore compile the software from source.[3] The present article does not describe the installation process nor the main window displayed at startup. Instead, it provides an overview of basic uses and a more detailed discussion of advanced functionalities. By default, the interface is in French, but it can be set to English in the 'View' menu[4].

[1] https://outils.biblissima.fr/fr/collatinus-web/.

[2] https://outils.biblissima.fr/fr/collatinus/.

[3] https://github.com/biblissima/collatinus/tree/Medieval.

[4] To apply the changes you may need to restart the program.

2. Simple uses

2.1 Basic usage

The upper part of the window has a large editing area where a text can be inserted, either by typing it directly, copying it from elsewhere, or opening a file containing a plain text (TXT-format). The lower part of the window displays the results. It is divided into tabs depending on the type of output required. The main tabs are 'Lexicon and morphology', 'Dictionaries' and 'Scansion'.

For simple queries as the lemmatization of a single form (in the 'Lexicon and morphology' tab), the user can type the form into the small input line and press 'Enter': the possible lemmata corresponding to the form are then displayed, together with a short translation in the chosen target language (which can be changed in the 'Lexicon' menu), when available. Depending on the active options, the corresponding analyses (case and number for nouns, TAME for verbs etc.) are also provided.

It should be noted that, when several solutions are possible, the order in which lemmata are displayed reflects their frequency rather than alphabetical order. It does not imply that the first solution is the correct one. This is mainly done to avoid that the lemmatization of *suis* gives the solution *sus* before *suus*. Similarly, in the 'Dictionaries' tab, the user can enter a lemma to retrieve the corresponding entry in the selected dictionary.[5] In the same way, a form can be scanned or accented in the 'Scansion' tab.

The last two tabs, 'Inflection' and 'Tagger', will not be described in detail here. The former is self-explanatory and provides tables of inflected forms for any lemma: it may be useful for beginners who wish to become familiar with inflectional patterns. The latter was originally intended as a context-based lemmatizer where the solutions are ranked using contextual information. It relies on a third order Hidden Markov Model (HMM-3) to select the most likely tag sequence for a sentence. The statistical parameters of the HMM-3 were derived from the LASLA annotated corpus [4]. This approach has since been superseded by the *Collatinus*-LASLA tagger based on AI techniques, which is described below.

2.2 Reading help and vocabulary list

We have seen how to retrieve information about a single form or lemma, but in most cases the user works with an entire text, either for reading or for preparation. As already mentioned, the text can be inserted into the main editing area. We now describe how it can be exploited.

When the mouse cursor is placed on a word in the text window, a tooltip appears displaying the possible lemmatizations of the word together with the short translations (and, if the corresponding option was selected in the 'Lexicon and morphology' tab, with the associated morphological analyses). This feature is particularly useful for students who are beginning to study Latin and need assistance with unfamiliar vocabulary.

[5] A selection wheel allows to change the dictionary in the list of available ones. An extra window can be displayed to consult simultaneously two different dictionaries.

Clicking on a word produces the same effect as copying it into the input field of the active tab and pressing 'Enter'. In the lemmatization tab, this action displays the possible lemmata with the translations in the results window. Similarly, the scanned or accented form is shown in the 'Scansion' tab, and the dictionary entries are displayed in the 'Dictionary' tab.

In both the lemmatization and scansion tabs, the entire text can be processed by clicking the button represented by gears in the toolbar. This button preserves the order of words in the text, allowing the user to scroll through the results window while reading the text sequentially. It is also possible to process a part of the text by selecting it beforehand, although care must be taken to include complete words (the program does not extend the selection to have whole words). These operations are not meaningful in the dictionary tab.

In the same way, teachers or lecturers can prepare the vocabulary list associated with a text by using the gear-button. The content of the results window can then be copied and pasted into a preferred text editor (LibreOffice Writer preserves character formatting) and adapted to the needs of the pupils or students. To increase the level of difficulty, the alpha-button can be used: this lemmatizes the text, but presents the lemmata is alphabetical order. Another approach consists in clicking successively on selected words in the order in which they appear in the text: their lemmatizations are appended to the existing results in the window. As this method can be tedious, an additional tool has been introduced: *TextiColor*.

2.3 TextiColor

TextiColor takes into account the supposed knowledge of the pupils or students. Before generating a vocabulary list, the teacher gives the computer a list of known lemmata selecting the 'Read a list of known words' item in the Files menu. A pop-up window then opens, allowing the user to select the file.[6] The file is a plain text file with one lemma on each line. The *TextiColor* mode is then active until you click on the same item in the Files menu a second time.

In *TextiColor* mode, when a text is lemmatized using the gear-button, all words are processed, but not all results are necessarily displayed. If, among the possible lemmatizations of a form, a known lemma is found, the solutions are not displayed: since students are assumed to know the lemma, they are expected to be able to analyze the form themselves. This assumption must nevertheless be treated with caution. It clearly applies to frequent invariable words (e.g., *et*, *cum*, etc.) and, reasonably, to nouns and adjectives once their declension has been thoroughly studied. For verbs, the situation is more delicate because conjugation tables are seldom taught at once. If the form is a subjunctive perfect of a known verb, *Collatinus* in the *TextiColor* mode will consider it as known, which may not be the case.

A collateral effect of lemmatizing a text in *TextiColor* mode is that the text displayed in the editing window is color-coded: the forms that are assumed to be known are displayed in blue, the forms that *Collatinus* knows but that the student is allowed to ignore are shown in black and the forms that *Collatinus* did not recognize appear in red. This side effect explains the name *TextiColor* and provides a convenient way to assess at a glance whether the level of a text is

[6] Different lists can be used for the different levels of the classes, and these lists will change during the year.

appropriate for a given class.[7] If the text appears entirely in blue, it should be theoretically readable without difficulty. If it contains 10% of black words, the reading is interrupted by occasional consultations of the vocabulary list, but remains reasonably fluent. Although no strict threshold can be defined, experience helps teachers judge the suitability of a text.

TextiColor mode can also be used without any list of known words.[8] Then, only two colors will be displayed: red and black. This configuration can be employed to check OCR-processed texts. A word in red may well be correct as *Collatinus* does not cover the entirety of Latin vocabulary, but it nevertheless merits verification, as it may indicate a typographical or OCR error. *Collatinus* includes almost all the words that are in *Lewis & Short* and *Gaffiot*, which is sufficient for most classical texts. Conversely, a word in black may result from an OCR error if the erroneous form happens to coincide with an existing Latin word. In any case, lemmatization in *TextiColor* mode can be a useful aid in detecting misprints.

3. Use in research

In this section, I present several examples of research-oriented uses of *Collatinus*, together with a number of dedicated auxiliary programs. These examples should be understood as case studies rather than as fully standardizes workflows. In many instances, they may be regarded as work in progress, and user feedback remains essential for further development.

3.1 Export to CSV

Since it is impossible to anticipate all possible research needs, a generic export function was introduced in *Collatinus*, allowing users to treat the exported data as input for external programs. The aim is to enable researchers to develop their own analyses without having to deal again with lemmatization or scansion. The key function, called `txt2csv`, processes any plain text file and produces a CSV-file containing either the lemmatization or the scansion of the text, depending on the active tab. Options specific to the active tab are not taken into account. The CSV-file uses the tab-character as a field separator.

In the case of lemmatization, the CSV-file contains seven columns. The first column gives the position of the word in the text. If a word has several possible lemmatizations, it appears on several consecutive lines. The second provides an indication of the frequency of the associated lemma: 1 for very frequent lemmata, 2 for common ones and 3 for less common ones.[9] The third column contains the word as it appears in the text. The fourth column gives the lemma without quantities (facilitating lookup) while the fifth one provides the lemma with quantities together

[7] This tool has been asked to me by a Dutch lecturer, Jan Bart, for this reason.

[8] When the file selection window appears, the user simply clicks the *Cancel* button, which activates TextiColor mode without any list.

[9] The frequency is simply the number of occurrences of the lemma in the corpus of the LASLA which has about 1.8 million of words. The given figures are associated with a frequency which is respectively larger than 5000, in the range 500-4999 and smaller than 499.

with the morphological information. The sixth column contains a short definition of the lemma and the final one gives the frequency of the lemma.

In the case of scansion, the CSV-file contains ten columns. As scansion often deals with verses, end-of-line characters are considered significant and are encoded using a pseudo-token (¶). The first three columns contain numerical indices locating the word within the text.[10] The next two columns give the word as it appears in the text and any following punctuation or whitespace). This makes it possible to reconstruct the original text exactly from the CSV-file. The sixth column provides the normalized form, which is mainly relevant when dealing with medieval variants of classical spelling. If several normalizations are possible, the most frequent one is listed first, with alternatives given in parentheses. The same principle applies to the remaining four columns, which provide, respectively, the form with quantities, a compact encoding of these quantities, the accented form, and its encoding.[11] In these two pairs of columns, multiple variants appear in the same order in the corresponding fields (for instance, between columns 7 and 8). However, in the seventh and ninth columns, the order might be different and the number of objects can differ too.[12]

3.1.1 Towards augmented books

The introduction of the `txt2csv` function for lemmatization has been triggered by Anthony, a British lecturer who wished to create augmented books (in .pdf format, intended for print-on demand). The structure of such a book is as follows. It begins with a vocabulary list of the common lemmata accompanied by short definitions; this list includes words that occur more than five times in the text.[13] It is followed by the text, and on each page the reader finds the necessary vocabulary –excluding the most frequent words already listed at the beginning – together with comments or explanations added by the author. The book concludes with the list of all the lemmata occurring in the text, indicating their number of occurrences and a short definition. The CSV-file containing the lemmatization of the text constitute an appropriate starting point for this workflow.

In practice, however, a significant amount of manual work remains necessary. First, spurious lemmatizations must be removed: when a form corresponds to several possible lemmata, irrelevant lines must be deleted. At this stage, it is not essential to revise the translations, as this can be done later when the vocabulary lists are generated automatically. At the same time, or subsequently, authors may add comments associated with specific words by inserting a new column at the end of the corresponding

[10] Rank in the text, verse number and rank in the verse. If the text is prose, the verse number is just the paragraph number. If some sign comes before the first word of a verse (or paragraph), it is placed on the line of the special token ¶ that announces the beginning of the verse. For that purpose, the CSV-file begins with a virtual ¶ with rank 0.

[11] To encode the quantity of the syllables, a + sign denotes a long syllable, a - sign is a short one, a * symbol represents a common one (or of unknown quantity) and, in case of elision, a ` character (grave accent) stands for the elided syllable. For the rhythm, the encoding consists in the number of syllables followed by p (paroxyton) or pp (proparoxyton) if the number of syllables is strictly larger than 2.

[12] For instance, *populus* has two different scanned forms, *pŏpŭlŭs (pōpŭlŭs),* but only one accented form, *pópulus*.

[13] This threshold of 5 has been chosen by the author, but can be tuned in the program.

lines.[14] Basic HTML-tags may be used to format these comments.[15] A comment may begin with the word or expression being discussed, displayed in bold, and may address grammatical, lexical, semantic, or contextual issues. In addition, the text itself must be prepared so as to match the desired final layout of the book. An optional intermediate step consists in generating the vocabulary list (in a CSV format): the author can then revise the short definitions provided by *Collatinus* in order to supply more precise or context-specific translations.

An ad hoc program subsequently processes the available files to generate an HTML document in which each page is divided into three sections: the text, the vocabulary, and the comments (when present).[16] The program estimates the space required for definitions and comments related to each line in order to decide whether this line can fit on the current page or must be moved to the next one. Definitions and comments always appear on the same page as the words to which they refer. The program attempts to fill pages as efficiently as possible but does not split a comment across pages. The resulting HTML file can then be opened in a suitable text editor for final adjustments and minor corrections, although page boundaries must be respected.

The same CSV-file containing comments could also be used to produce the book in .epub format. In such a version, clicking on a word would display its translation, while comments would appear as footnotes. Some attempts were made in this direction, but difficulties arose from differences in how various EPUB-readers handle footnotes and navigation back to the main text. This approach was therefore abandoned, pending further interest or technical improvements.

3.1.2 Dactylic verses

Dactylic verses are quite common in Latin poetry. They are composed of metrical units that are either spondees (two successive long syllables, often noted — —) or dactyls (one long syllable followed by two short ones, — ∪ ∪). According to additional rules, these units are combined to form hexameters or pentameters. Two types of problems may be addressed in this context. On the one hand, dactylic verses may be embedded in prose texts, and the scholar may wish to identify them [6]. On the other hand, a scholar may seek to compare two dactylic poems in order to determine whether specific words occur in similar metrical positions. In both cases, the starting point is the CSV-file produced by *Collatinus* containing the scansion of the text.

3.1.3 Hidden verses

When searching for hidden verses in prose, a relatively simple algorithm can be applied. Assuming that any word beginning with a long syllable may serve as the first word of a verse, the algorithm tests each possibility by successively adding words and checking whether their scansion is compatible with the metrical scheme of a hexameter or a pentameter. An additional constraint is imposed: the verse must end at a word boundary and not in the middle of a word.

[14] If the comment refers to an expression with more than one word, the author has to choose one of them as the anchor for the comment.

[15] For instance, <b> for bold, <i> for italics, <sup> and <sub> for superscripts and indices. Always with the corresponding closing tag to form a consistent HTML line.

[16] Another template is possible: the text is on one page and the short definitions and the comments are on the facing page.

The procedure is nevertheless not entirely straightforward, as some words admit multiple scansions (for example *pŏpŭlŭs* and *pōpŭlŭs*), and some words may not be recognized by *Collatinus*, in which case all syllables are treated as potentially long or short. From a computational perspective, however, this does not pose a serious difficulty: all possible paths can be explored in a tree structure, branches that do not fit any valid metrical pattern are pruned, and the exploration continues until a hexameter or pentameter is found.

If, at some point, all branches are eliminated, or if a candidate verse ends in the middle of a word, the algorithm restarts the process with the next word as a potential beginning of a verse. The program described here does not apply criteria for identifying a "good" verse; for example, it does not enforce the rule that the penultimate meter of a hexameter should be a dactyl.

The main difficulty lies not so much in detecting potential dactylic verses as in presenting the results in a meaningful way. Since no objective criterion exists for excluding certain verses, all detected candidates are retained, and it is left to the scholar to determine whether a given verse is significant or merely coincidental.

3.1.4 Comparing poems

The problem addressed here was proposed by Elena Ghiringhelli, who sought to compare Ovid's Fasti with a *Continuatio* written by Morisot in 1649 [2]. The aim was to determine which words were used by each author and the metrical positions in which they occurred. Since *PedeCerto*[17] is the world reference in the field, its encoding scheme was adopted here, with minor modifications. *PedeCerto* is able to scan dactylic verses,[18] but it does not, to our knowledge, provide a list of words together with their prosodic structure and encoding.

Although this task may appear similar to the previous one, it differs in a crucial respect: the boundaries of each verse are known in advance. While the construction and pruning of the tree of possible scansions are similar, the output and its encoding are different. The program produces, simultaneously, an analysis of each verse as a sequence of dactyls and spondees, and a list of words together with their positions within the verse. However, too little room is left for scholar's experience. For example, a computational approach would not identify Virgil's *Laviniaque* as *Lāvĭ́niăquĕ*, with the *i* of the penultimate syllable functioning as a consonant.[19] Elsewhere (23 occurrences in *PedeCerto*), the second i in Lavinia is always scanned short, as it should be.

Constructing all possible sequences of quantities for a verse without pruning the tree, and computing a Levenshtein distance between these sequences and the canonical patterns of hexameters or pentameters, could make it possible to detect poetic licenses when a verse does not scan correctly. Another situation requiring scholarly intervention arises when unknown words (for instance proper names) form a sequence of five syllables with undetermined quantities. Such a sequence may correspond either to a dactyl – spondee or to a spondee – dactyl pattern. While

[17] https://www.pedecerto.eu.

[18] To access the feature either select 'Utilities/Free scansion' or 'Utilities/Prosodic structures' or click on https://www.pedecerto.eu/scansioni/scansioni and https://www.pedecerto.eu/lessico/lessico.

[19] PedeCerto gives to VERG. *Aen.* 1,2 Ítălĭám fātó prŏfŭgŭs Lāuĭ́niăquĕ uĕnit DSDS, but all the verses have been checked by a scholar.

the verse is recognized by the program, the assignment of metrical positions to words remains ambiguous.[20] For this reason, the code would probably benefit from being rewritten as a two-step procedure: in a first step, verses would be scanned automatically; in a second step, the scholar would validate or correct the scansion before the extraction of detailed metrical information.

3.1.5 Rhythmic prose

Over time, the modulation of vowel length gave way to tonic accentuation, and classical metrical clausulae gradually evolved into rhythmic ones, also known as *cursus*. The study of the use of *cursus* in texts can provide valuable evidence for dating or attributing works [3]. The program described here is essentially an update, based on *Collatinus*, of works previously carried out with Anne-Marie Turcan-Verkerk [8].

The program processes a CSV-file produced by *Collatinus* and compiles statistics about the rhythmic structure of the sequences of two words, considering the entire text as well as clause endings and sentence endings (*cola* and *commata*). The results are presented in large tables intended for detailed scholarly analysis, while a chi-square (χ^2) analysis highlights statistically significant tendencies. More compact and readable tables summarize the distribution of the four main rhythmic patterns (*planus*, *tardus*, *velox*, and *trispondaicus*) and their variants at the ends of clauses and sentences.

The table below shows the rhythm at the end of the sentences for a very short text. The first column with white background contains the number of occurrences and the frequency of the standard form of the cursus (which is given in the first line of each cell). The other columns are about different ways to divide up words within each rhythm, while keeping the same positions for the accents.

Planus	p 3p % 4 27	p 1+2 % 1 7		pp 2 % 2 13	Total % 7 47
Tardus	p 4pp % 0 0	p 1+3pp % 2 13		pp 3pp % 0 0	2 13
Trispondaicus	p 4p % 0 0	p 1+3p % 0 0	p 2+2 % 1 7	pp 3p % 0 0	1 7
Velox	pp 4p % 0 0	pp 1+3p % 1 7	pp 2+2 % 0 0	p 5p % 0 0	1 7
Other	4 27				

Tab. 1 - statistics about the use of cursus at the end of sentences. The lines correspond to the main cursus and their variants. In each cell, we give the associated rhythm (first line) and the number of occurrences and the corresponding percentage (frequency).

[20] *PedeCerto* fails to scan HILD. CEN. (?) *reg.* 1 Ēlcănă dē *Ramatha Phenēnnāe* spōnsŭs ĕt Ānnāe, where *Ramatha* and *Phenennae* are unknown words. My program proposes two solutions DSDSDX and DDSSDX. The third verse of this poem suggests that the first syllable of *Phenenna* is long, which validates the second option, in contrast with the onomasticon Forcellini-Perin which indicates all the syllables of *Ramatha* as long (which cannot be to fit a dactylic hexameter in the present case).

These tables make it possible to compare different texts by collecting the percentages associated with each rhythmic type and plotting the results. Valérie Thon applied this method to a sample of letters by Petrus Damianus and showed that rhythmic preferences differ between sentence endings and clause endings (Fig. 1 and Fig. 2) [7]. In her study, *velox* and *planus* predominate at the end of sentences, whereas *tardus* and *planus* are more frequent at the end of clauses.

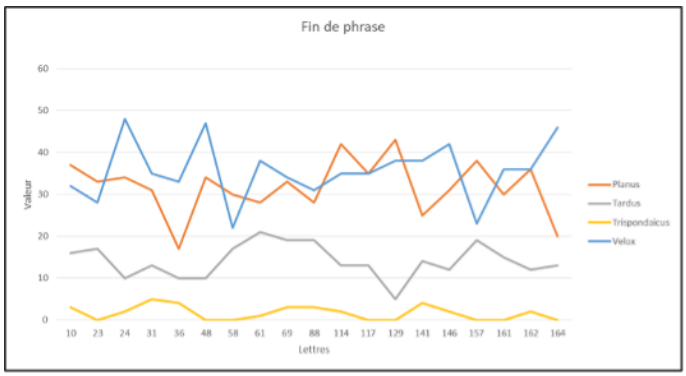

Fig. 1 - Use in % of the main cursus at the end of sentences. The horizontal axis is the identifier of the letters

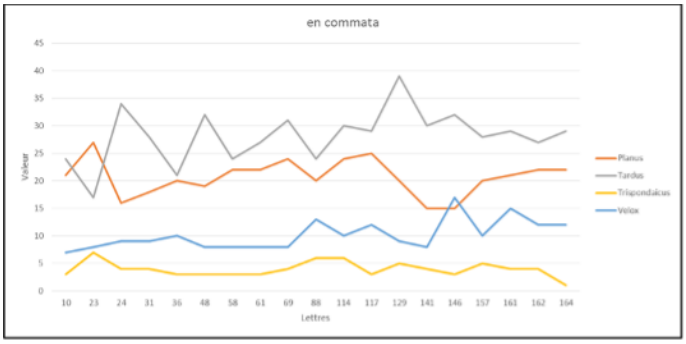

Fig. 2 - Use in % of the main cursus at the end of clauses. The horizontal axis is the identifier of the letters.

3.2 AI-assisted tagger

As discussed above, *Collatinus* is a lemmatizer with an extensive lexical coverage. However, when a form admits several possible analyses or derives from different lemmata, *Collatinus* alone cannot determine, on the basis of context, which solution is the most appropriate for automatic text annotation. An initial version of the *Collatinus*–LASLA tagger relied on a third-order Hidden Markov Model [9]. The statistical parameters of this model were derived from the APN-annotated LASLA corpus.

More recently, with the rapid development of AI techniques, an additional predictive layer has been introduced to improve tagging accuracy [10]. This AI component is based on Latin-BERT

[1],[21] which was available at the outset of the project. In their case studies, the authors of Latin-BERT provided an example of part-of-speech tagging that was adapted to the present needs. The model was trained on the LASLA annotated corpus, producing distinct models for the tag, the homonymy index, and the subordination code.

Despite this integration of AI, the underlying principles of the previous system have been retained. Lemmatization itself remains rule-based and is not entrusted to AI. If a form is not recognized, the system does not attempt to guess a lemma, as purely AI-based approaches might do; instead, the form is simply labeled as unknown. An optional preprocessing step allows unrecognized forms to be collected and submitted to the philologist, who may either create a new lemma for *Collatinus* or provide the lemma and morphological analysis in LASLA APN format.

AI predictions are systematically checked: if a prediction does not correspond to any of the analyses proposed by *Collatinus*, it is rejected, and the philologist is informed of this rejection. After the automatic annotation phase, the resulting file may be edited manually. In particular, the philologist can first inspect conflicting predictions and then revise the file as a whole. In a more expedient workflow, however, the annotated file can be used directly and imported into Hyperbase for further analysis. User feedback indicates an accuracy exceeding 90%, in line with the evaluation obtained on test data during training [10]. The possibility of exporting annotated texts to additional formats is currently under investigation.

Conclusion

Collatinus is a free tool for the study of Latin whose capabilities extend well beyond its original pedagogical aims. Although it was initially designed for teaching purposes, its functionality makes it a valuable resource for a wide range of scholarly applications. One of its major strengths lies in its lexical database, which has been compiled from available digital dictionaries.

The lexical base of *Collatinus* is currently being expanded, in particular through the inclusion of medieval Latin vocabulary and proper names. The modular architecture of the software also makes it possible to integrate its components into other specialized programs, such as the AI-assisted tagger described above.

References

[1] Bamman, David, and Patrick J. Burns, "Latin-BERT: A contextual language model for classical philology". arXiv Preprint arXiv:2009.10053. https://doi.org/10.48550/arXiv.2009.10053.

[2] Ghiringhelli, Elena. 2024. "La continuation des *Fastes* d'Ovide par le Dijonnais Claude-Barthélemy Morisot (1649): introduction et traduction annotée du mois de juillet du calendrier romain". PhD thesis. Université Bourgogne Franche-Comté.

[21] https://github.com/dbamman/latin-bert.

https://theses.hal.science/tel-05019439

[3] Jansson, Tore. 1975. *Prose Rhythm in Medieval Latin from the 9th to the 13th Century*, Almqvist & Wiksell International.

[4] Longree, Dominique, and Fantoli, Margherita. 2023. "LASLAfiles_Latin_APNformat", V1. ULiège Open Data Repository. https://doi.org/10.58119/ULG/QJJ0SA.

[5] Ouvrard, Yves, and Philippe Verkerk. 2014. "*Collatinus*, un outil polymorphe pour l'étude du latin". *Archivum Latinitatis Medii Aevi* 72: 305-311. https://www.persee.fr/doc/alma_0994-8090_2014_num_72_1_1156.

[6] Roelli, Philipp, and Jan Ctibor. 2022-2023. "A new version of *Corpus Corporum*, the Latin full-text database and tool". *ALMA* 80: 251-266.

[7] Thon, Valérie. forthcoming. *Quand écrire, c'est agir. Étude linguistique des lettres de Pierre Damien (XI^e siècle)*. PhD thesis (Liège).

[8] Turcan-Verkerk, Anne-Marie, and Philippe Verkerk. 1996. "Un programme informatique pour l'étude de la prose rimée et rythmée". *Le médiéviste et l'ordinateur* 33: 41–48. https://hal.science/hal-04394988v1.

[9] Verkerk, Philippe, Yves Ouvrard, Margherita Fantoli, and Dominique Longrée. 2020. "L.A.S.L.A. and *Collatinus*: A convergence in Lexica". *Studi e Saggi Linguistici*, 58 (1): 95-120. https://hal.science/hal-02399878v1.

[10] Verkerk, Philippe. 2022-2023. "Elaboration of a practical lemmatizer for Latin using Artificial Intelligence". *ALMA* 80: 267-294. https://hal.science/hal-04721577v1.