SSL

L.A.S.L.A. and Collatinus: A convergence in lexica

Philippe Verkerk, Yves Ouvrard,
Margherita Fantoli, Dominique Longrée

Abstract

The research group *L.A.S.L.A.* (Laboratoire d'Analyse Statistique des Langues Anciennes, University of Liège, Belgium) began in 1961 a project of lemmatization and morphosyntactic tagging of Latin texts. This project continues with new texts lemmatized each year (see *http://web.philo.ulg.ac.be/lasla/*). The resulting files, which contain approximatively 2,500,000 words, whose lemmatization and tagging have been verified by a philologist, have recently been made available to interested scholars. In the early 2000's, Collatinus was developed by Yves Ouvrard for teaching. Its goal was to generate a complete lexical aid, with a short translation and the morphological analyses of the forms, for any text that can be given to the students (see *https://outils.biblissima.fr/fr/collatinus/*). Although these two projects look very different, they met a few years ago in the conception of a new tool to speed up the lemmatization process of Latin texts at *L.A.S.L.A.* This tool is based on a concurrent lemmatization of each word by looking for the form in those already analyzed in the *L.A.S.L.A.* files and by Collatinus. This lemmatization is followed by a disambiguation process with a second-order hidden Markov model and the result is presented in a text-editor to be corrected by the philologist.

Keywords: lemmatization, morphosyntactic analysis, disambiguation, probabilistic tagger.

1. L.A.S.L.A.

The Laboratory for Statistic Analysis of Classical Languages (*L.A.S.L.A.* in the following) was founded in November 1961 at the University of Liège, by L. Delatte and E. Évrard. Its original aim is to lemmatize and analyze (tag) literary classical texts, both in Greek and in Latin, in order to produce indexes and to allow the study of classical languages with statistical and quantitative methods. This project, which is still on going, has already produced a large digitalized, lemmatized and annotated Latin corpus. This corpus covers the classical period, from Plautus to Ausonius, with some other Late-Latin texts. The *L.A.S.L.A.* Encoding Initiative interface allows the addition of new texts to the corpora. *L.A.S.L.A.* also released Textual

Received: January 2020
Accepted: May 2020

Data Analysis tools to access the information contained in its files (amongst which, for instance, the software Hyperbase; see *http://hyperbase.unice.fr/hyperbase/*). Through a specific agreement, access to these files is now free and open for every scholar who requests it.

1.1. *The structure of the files*

The *L.A.S.L.A.* Latin files contain fully lemmatized texts with a complete morphosyntactic analysis and some syntactic information. They have been systematically verified by a confirmed Latinist (either M.A. or Ph.D.). The annotation is not related to any specific grammar or to any specific linguistic description. In short, the available files are put in a text format where each line contains all the information related to a single token. As a reminiscence of the old punched cards, the fields have a fixed length, the blank character filling the empty spaces.

For each token of the text, the line begins with a unique alphanumeric code that identifies the text and a number that indicates the sentence count. All punctuation, which has been added by modern editors, is removed, except for the period that separates the sentences. The line then contains the lemma – as it appears in the dictionary of reference[1] – associated with an index if there are different homographs or to mark proper names or their derived adjectives. Then comes the form as it appears in the text, the reference – according to the *ars citandi* – and the complete morphologic analysis in an alphanumeric format[2]. For the verbs, an extra field, which remains empty for the other Parts-of-Speech (PoS in the following), gives some syntactic information: the verb of the main clause is identified and a subordinate code – depending on the subordination type – is affixed for the other verbs in the sentence.

The lemma always refers to an entry in the Forcellini's dictionary with a systematic disambiguator. For instance, POPVLVS_1 (i.e. *pŏpŭlus*, i, m.) is the people, while POPVLVS_2 (i.e. *pōpŭlus*, i, f.) is the poplar[3]. The PoS is also used to distinguish the homographs as AMICVS_1, the substantive, and AMICVS_2, the adjective. A problem arose for late Latin texts where an adjective can become a substantive. This is the case for SANCTVS,

[1] Cf. Forcellini (1864).

[2] As a matter of fact, two alphanumeric encodings co-exist, one in 5 characters – which is the original one – and the other with 9 – which is simpler. The matching can be done automatically.

[3] The two words are differentiated by vowel length and gender. POPVLVS_1 (*pŏpulus*), masculine, means "people" while POPVLVS_2 (*pōpulus*), feminine, means "poplar".

which is only an adjective in Classical Latin, but became a substantive later, especially in religious texts. To handle this situation an extra tag has been introduced: 'use as a substantive'.

During the tokenization process, the enclitics are separated from the rest of the form, but a special character is inserted in the line as a reminder that those two tokens correspond to a single word. Conversely, the encoding allows the treatment of verbal compound forms and also ellipsis. Crasis is treated in a way quite similar to enclitics: one word leads to two lemmata. Tmesis and compound words are also encoded in a special way.

The 9-character morphologic tag begins with a one letter PoS (A=noun, B=verb, C=adj, etc.), followed by a figure indicating the declension (for a noun), the conjugation (for a verb) or the class (for an adjective). Then come single digits indicating, if relevant, the case, number, degree, mood, tense, voice and person. For the same lemma, the figure indicating the declension can vary. For instance, *Vlixes* belongs, in principle, to the third declension. However, in accusative singular, the two forms *Vlixem* and *Vlixen* exist and are associated to different tags: A331 for the first one, as it is the normal Latin form, and A731 for the second form which is the Greek one. For the genitive, the two forms *Vlixi* and *Vlixei* are characteristic for the second declension, so the tag is now A241, although the lemma is still VLIXES. The gender is an extra piece of information but, due to the original decision made by the founders, it is not given for nouns and is not fully disambiguated. As a matter of fact, there are six possible genders according to the *L.A.S.L.A.* files[4].

1.2. *The* L.A.S.L.A. *Encoding Initiative interface*

The *L.A.S.L.A.* Encoding Initiative interface (see *http://cipl93.philo.ulg.ac.be/LaslaEncodingInitiative/*) is mainly a selection interface. A new text is first given to an operator who proceeds to some preprocessing[5]: tokenization, lemmatization and analysis. Until 2019, this was achieved with an analytic lemmatizer: starting from the ending, the lemmatizer broke the form down into morphemes in order to get all the possible roots and

[4] The three real genders and the three combinations that exist in the declensions (the f. + n. combination does not exist). We have plans (not yet fully implemented) to add the gender of the nouns and to disambiguate, when possible, the gender of adjectives, depending on the associated noun. Part of the task can be done automatically, but the result will have to be checked. Some words, as *canis* or *pereger*, are common (both masculine and feminine) and, even with the context, it may happen that the gender cannot be decided with certainty.

[5] See Denooz (1978) and Philippart de Foy (2014).

then recomposed in order to get all the possible analyses for all the possible lemmata. But the software supporting this lemmatizer was obsolete and *L.A.S.L.A.* decided to build a new lemmatizer based on form recognition. It means that each word of the text is compared to all the forms present in the *L.A.S.L.A.* forms dictionary. This forms dictionary includes all the possible forms for all the lemmata included in the *L.A.S.L.A.* lemmata dictionary. These possible forms are generated with a software based on morphologic rules, which adds all possible endings to each root corresponding to a lemma from the *L.A.S.L.A.* lemmata dictionary. The limitation here is that only lemmata already found in treated texts are included in the dictionary.

After this preprocessing, the text is presented as a list of tokens followed by all the known lemmatizations and analyses, one per line, in the alphanumerical order of the tags (corresponding to a given and fixed order of PoS, PoS-subcategories and morphosyntactic categories). Then, the philologist comes into play by selecting the 'correct one'. He/she is also invited to enter the syntactic information for the verbs, as it cannot be guessed by the computer. If a form is not in the dictionary or if the proper analysis is not given, the philologist has to add the correct analysis. The validation of the annotated text is possible only when the philologist has selected one analysis for each form of the text. At the end, the treated text returns to an operator who puts it in its final form.

Such a procedure ensures that the philologist has checked the lemmatization and the analysis of each token. As the computer does not select a priori a solution (even if there is only one possible lemmatization and analysis), the philologist has to read every line on the screen. But this process has its drawbacks, especially for technical texts with a specific vocabulary. As the dictionary has been built on Classical Latin literary texts, such as historical works, speeches, poetry, etc., a large amount of technical and scientific Latin words are missing from the *L.A.S.L.A.* dictionary and the philologist has to add them one by one[6]. Moreover, when the philologist inserts a new analysis into the lemmatization and tagging interface, there is no way to copy automatically this new analysis to the same form which could appear further in the text. As a matter of fact, to guarantee the coherence of its dictionary, *L.A.S.L.A.* does not update it automatically with the new forms.

[6] Indeed, the lexical specialization of technical and scientific texts, like didactic works and treatises on specific topics, is a feature which has been studied under many points of views, see for example De Meo (2005) and Fögen (2011).

1.3. *Access to the information*

There are several ways to access the information stored in *L.A.S.L.A.* files. The simplest approach is given by the interface Opera Latina[7] which allows for documentary search (indexes) but gives no statistics. A second possibility is to download the package Hyperbase-Latin which allows documentary and statistical exploration. This software has been developed in collaboration with Étienne Brunet of the laboratory Bases, Corpus, Langage (UMR 7032; CNRS-University of Nice)[8].

A more flexible approach is offered by the Hyperbase Web Edition interface[9]. One can choose between various databases or corpora. Beyond the usual documentary search (indexes), one can also ask for pattern detection – for instance all the sequences of two nouns. The Hyperbase Web Edition allows statistical searches such as z-score, factorial analysis or tree analysis. It is also possible to study the co-occurrences and even co-occurrences of pairs. As an extension of the Hyperbase Web Edition, HyperDeep, which is based on a Convolutional Neural Network, allows the identification of what is characteristic of a text or to find influences between authors.

For more specific purposes, *L.A.S.L.A.* files can be converted to XML and treated with TXM[10] or with data-mining tools[11].

2. *Collatinus*

Collatinus[12] was originally developed by Yves Ouvrard for teaching. It allows the generation of a complete lexical aid, with a short translation and the morphological analyses of forms, for any text which can be given to the students. As time went by, the lemmatizer has been augmented with other useful tools[13]. By simply clicking on a word, one can open a digital dictionary, e.g. Lewis and Short (1879) or Gaffiot (2016), to have the complete definition of the lemma. Another possibility is to scan a text to identify its

[7] Cf. *http://web.philo.ulg.ac.be/lasla/opera-latina/*. The list of the available texts is given at *http://web.philo.ulg.ac.be/lasla/textes-latins-traites/*.

[8] Cf. *http://web.philo.ulg.ac.be/lasla/hyperbase/*.

[9] Cf. *http://hyperbase.unice.fr/hyperbase/?edition=lasla*.

[10] Cf. *textometrie.ens-lyon.fr/spip.php?rubrique96*.

[11] Cf. *https://tal.lipn.univ-paris13.fr/sdmc/*.

[12] See Ouvrard and Verkerk (2014).

[13] For more details about these functionalities, see the article Ouvrard and Verkerk (in press), available as preprint at *https://hal.archives-ouvertes.fr/hal-02385036*.

metrical structure. A probabilistic tagger, based on a second order hidden Markov model (shorten as *HMM* in the following), allows the selection of the best lemmatization and analysis for each form by taking into account its context.

The lemmatization of a form is obtained by trying to split it as a root associated with a standard word-ending, which reproduces what the human reader does. The advantage of a program like Collatinus is that it is able to recognize forms not yet seen as soon as the root-word is known[14]. It is also easier to improve its base of knowledge: adding data for a new root-word allows the immediate recognition of ten or more (even a hundred, for verbs) forms[15]. Obviously, a program like Collatinus 'knows' a lot of forms that are not attested in the texts that have survived[16].

2.1. *Principle of operation*

When a student learns Latin, the first thing he/she has to understand is the way forms are constructed. Words are connected to an inflection paradigm. For each paradigm, one has to learn the list of word-endings and the rules to combine these endings with the roots that can be calculated, in some cases, or must be given. Collatinus works exactly in this way: one file provides the word-endings and the construction rules for each paradigm while another file connects the lemmata to the paradigms and provides also the roots which cannot be constructed. With this data, the construction of the inflected forms is immediate.

The lemmatization of a form requires the reverse process. For a given form, we have to split it in all the possible ways and to check that the first part coincides with a known root and the last one with a word-ending associated to the paradigm of the root[17]. The word-endings carry part of the information for the analysis, which is then stored in the file. Instead of an explicit analysis as e.g. 'nominative singular', we made a list of morphosyntactical analyses, which are possible in Latin and coded the analysis with a

[14] For any unknown form coming from an unknown root-word, it should be possible to guess a reasonable root-word in some simple cases.

[15] As it was the case before for the original *L.A.S.L.A.* lemmatizer.

[16] Note that, if the classical corpus is well established, it is not the case for medieval Latin.

[17] Going further, one can imagine to guess the lemma simply by subtracting the common word-endings. However, it would lead to surprising results. For instance, the form *merobibus* could be analyzed as an ablative plural of an hypothetical *merobis*. But such a method could give good results if several forms of the same lemma are found in a text.

simple number. As a matter of fact, the total number of these possible analyses amounts to 416. The number is converted into its human readable form when needed, i.e. for the display. Moreover, this encoding also allows the translation of the analysis into different languages[18].

2.1.1. *First difficulties*

One of the aims of Collatinus is to treat a Latin text as it is, without requiring some preprocessing steps like tokenization. A difficulty appears because of the enclitics *-que*, *-ne* and *-ve*. These words may be appended at the end of any form, and have to be separated before lemmatization. In most of the instances, the enclitics *-que* and *-ve* do not lead to ambiguous forms[19], which is not the case of the enclitic *-ne*. For instance, a form as *mentione* could be analyzed as the ablative singular of *mentio*, *onis*, as well as the nominative followed by the enclitic *-ne*. Enclitics, however, are not so frequent. We therefore assume that, if a form can be lemmatized as it is, then it is not necessary to search for the enclitics. In other words, the form *mentione* is now analyzed only as the ablative of *mentio*.

Collatinus also knows some contraction and assimilation rules. For instance, a double *i* appearing in the flexion of a word[20] is frequently written as a single long *i*. Some forms of the perfect can be contracted, the *-vi-* disappearing in, for instance, *amasse* (for *amavisse*). These forms are recognized by Collatinus, without the necessity of adding new word-endings. For the verbs constructed with a prefix, assimilation can change the spelling in some cases. It is the case, for instance, of *adfero*, *adtuli*, *adlatum* which often becomes *affero*, *attuli*, *allatum*[21]. The main assimilations of the prefix are known by Collatinus and built-in, so that it avoids the proliferation of forms for the same word.

2.1.2. *Distinction between* u *and* v

Very often, Latinists do not distinguish the letters *u* and *v*, and erase the *j* from the alphabet. But for scansion or counting syllables, it is clearly neces-

[18] For the moment, French, English and Spanish. But one can convert it to any other computer-oriented forms.

[19] A noticeable exception is *quo-que* that appears 7 times in the texts lemmatized by the *L.A.S.L.A.* (to be compared to the 2.290 occurrences of the lemma *quoque*).

[20] The first *i* ending the root, often short, and the second one at the beginning of the word-ending combine in a long *i*.

[21] Gaffiot (2016) gives the first forms, while Lewis and Short (1879) prefers the second ones.

sary to make a distinction. Thus, Collatinus keeps, in its lexicon and in the word-endings, the two consonants *v* and *j*, said to be Ramist consonants[22]. By the way, if one wants to use only *u* and *i*, it is easy to replace *v* by *u* and *j* by *i*. The proof, if needed, that preserving the distinction is the best choice is that the reverse process (restoring *v* and *j*) is almost impossible, and at least very difficult, except through a lemmatization method.

On the other hand, several Latin texts use only the *u* and *i*, and Collatinus knows this[23]. The solution to this problem is obtained through two steps. In a first step, all the *v* are replaced by *u* for the lemmatization. Then in a second step, the form is reconstructed from the root and the word-endings that eventually contain the *v* and *j*. As a result, a word as *uoluit* is analyzed as a form of perfect of either *volo* or *volvo*[24]. But if the text contains *voluit*, with a *v*, one can assume that it is not the perfect of *volvo*, otherwise it should have been written *volvit*, with two *v*'s. If the form of the text contains one (or more) *v*, the program eliminates any lemmatization that would lead to a reconstructed form with a different number of *v*'s.

Another class of *u* are not 'real' vowels, e.g. *suavis* or *sanguis*. It is also the case for the group *qu*, but in this group, the *u* is never a vowel. In the groups *sua* or *gui*, there are examples where the *u* is a vowel, for instance the possessive *sŭă* and the adjective *ām*bĭgŭīs[25]. It would have been shocking to write *svavis* or *sangvis* to stress that these words have only two syllables. Instead, we use the punctuated *-u* and write *sụāvĭs* and *sāngụĭs*[26].

2.1.3. *Word-endings and construction rules*

As already said, besides the lexicon which will be discussed later, Collatinus has another important file which gives the word-endings and the construction rules. For each paradigm, it gives the list of analyses and the

[22] Pierre de la Ramée (Petrus Ramus) is known in France to have introduced this distinction *u*/*v* and *i*/*j* in his *Gramere* (1562). But it seems that this idea appeared earlier in Spain (Antonio Nebrija, 1492) or in Italy (Giovanni Trissino, 1529). See Blanco and Bogacki (2014: 160 n. 24, 161).

[23] In the worst case, the editors write the capital *U* as *V*. It is not infrequent to find *Vnde* at the beginning of a sentence or to meet *Vlixes* in some texts.

[24] *Volvit* can also be a form of the present of *volvo*. The meaning of the sentence allows the reader to identify the correct form, but a computer does not understand the text. The case of *uoluit* can be a problem in prosody as it can counts for two or three syllables.

[25] The vowels are marked with a macron '¯' when they are long, as *ā* or *ī*, and with a breve '˘' when they are short, as *ĭ* or *ŭ*.

[26] Once again, if one does not want to use this strange character, it is easy to replace it by the standard *u*.

corresponding word-ending. A noun that follows a usual declension has 12 analyses and word-endings (some of them are identical), while an adjective has 108 possible analyses and word-endings. All the possible combinations of case, number, gender, degree, tense, mood and voice give 416 analyses which are just designated with a number. To avoid a very long enumeration of word-endings, we introduced a mechanism by which a paradigm 'inherits' the endings of its parent[27]. For instance, *miles* and *civis* have most of their endings in common, so we just have to indicate the differences.

Obviously, the word-ending is not the end of the story because one has to know the root to which this ending can be appended. For some declensions or conjugations, the roots can be calculated with just the lemma. For instance, for the first declension, it is sufficient to drop the last character of the lemma to have the root. In other cases, it must be given by the lexicon: one cannot guess the root *mīlĭt-* for the lemma *mīlĕs*. A more subtle example is the case of the first conjugation. In most cases, the roots for the perfect and the supine are obtained by adding *-āv-* and *-āt-* to the main root: the knowledge of the form *ămo* is sufficient to calculate the three roots *ăm-*, *ămāv- and ămāt-*, so it is not necessary to give them in the lexicon. But some verbs of the first conjugation do not follow this simple construction rule. To solve this problem, we have decided that if a root is given in the lexicon, it replaces the one that could be calculated. For instance, for the verb *sŏno*, we give the two roots *sŏnŭ-* and *sŏnĭt-* for the perfect and the supine.

2.1.4. *Ordering of the solutions*

For several forms, the result of the lemmatization is not unique[28]. Different words can lead to the same form, or a form corresponds to different analyses of the same word. Collatinus now gives the different solutions in an order that reflects the frequency of the use of the words. Up to version 10, the order of the solutions was alphabetical. As a result, the lemmatization of *suis*, for instance, gave the genitive of *sus*, *suis* as the first solution, although the ablative or the dative of *suus*, *a*, *um* are more likely.

[27] The construction rules are also transferred.

[28] There is a problem of vocabulary around the lemmatization: for the final user, the aim of a lemmatizer is to give *the* (unique) lemma associated to a given form in a given sentence. However, an operation that gives *all* the lemmata that can be associated with a form is also a lemmatization. We prefer to stick to this last sense and the full process with the association of a single lemma to a form is obtained with two steps: lemmatization and disambiguation.

Thanks to the statistics made from the lemmatized texts[29] of *L.A.S.L.A.*, we are now able to associate to each word of the lexicon a number of occurrences. Obviously, this number of occurrences is limited to the lemmatized corpus, but one can consider it as representative for the frequency of words. To go back to the previous example, *sus* appears 47 times in the texts of the *L.A.S.L.A.*, while *suus* appears 7,120 times. As Collatinus is not a form-lemmatizer[30], it does not know the number of occurrences for *suis* as dative plural of *suus* and for *suis* as ablative plural of the same *suus*. To order these two possible solutions, we make a strong assumption: the usage of the cases and number[31] (for nouns and adjectives; replaced by the mood for verbs) does not depend on the particular word. We still take into account the PoS[32] of the word. This evaluation does not reproduce exactly the observed frequencies, but remains a fair approximation. There are noticeable exceptions: for instance, *patres* is mainly a vocative plural, a case that is only very seldom used in other nouns/adjectives.

This ordering of the solutions is not sensitive to context. Its depends only on the form itself and its analyses. According to the statistics done on the lemmatized text of the *L.A.S.L.A.*, choosing the most frequent analysis gives the correct result in 80% of the cases. To reach a lower error rate, one can develop disambiguation methods based on the tagging of the words. These methods take into account, very crudely, the context of the word. They will be discussed later.

2.2. *Extension of the lexicon*

The lexicon of Collatinus contains the lemmata associated to a known paradigm, the different root-words that cannot be calculated and various pieces of information, such as the number of occurrences of this lemma in the texts lemmatized by the *L.A.S.L.A.* The translations of these lemmata are given in distinct files (one for each language) so that the material necessary to inflect or analyse the forms is independent from the translations. It also allows the addition of more languages for translations without having

[29] We did the statistical work a few years ago, and some new texts have been added to the corpus, which are not taken into account.

[30] We shall come back later on that example through the *L.A.S.L.A.* tagger.

[31] Unfortunately, the lemmatization by the *L.A.S.L.A.* does not give precisely the gender of the adjectives.

[32] Mainly: noun, adjective, verb and pronoun, as categorized by the *L.A.S.L.A.*

to duplicate or to change the basic information for the inflection. The files are just plain text-files, so that they can be edited and modified by the user to give better results.

Up to its version 10.2, the lexicon of Collatinus was set-up manually, the words being typed in when they were found in new texts given to the students. It contained slightly fewer than 11,000 entries, which allowed the lemmatization a significant portion of classical texts. However, we have decided to improve it by working on the dictionaries in a digital form. The two main dictionaries we have used are Lewis and Short (L&S), converted in XML by the Perseus Project[33], and Gaffiot, converted in TeX by a team lead by Gérard Gréco[34]. We have also used Georges[35] and Jeanneau[36] in their HTML forms. All these dictionaries are part of Collatinus. Some extra pieces of information were also used[37].

The first part of this work has been to collate all the lemmata together with the morphological information and the translation in each dictionary. The precise tagging of L&S and of Gaffiot, although very different, allows the compilation of very rich databases. The translations were probably the most difficult part of the job. Sub-entries, such as adjectives that derive from a noun that is the headword, were collected too. Orthographical variants, often indicated in an abbreviated form (e.g. *affĕro*, *better adf-)*, were expanded and added to the base. This has been done automatically but checked afterwards. The internal variants, (e.g. *rĕverto*, *rĕvorto)*, have been especially difficult to treat, although they are rather intuitive for the human reader. Obviously, one has to acknowledge the imperfection of the tagging[38]: some tags are missing or do not include all relevant information.

To deal with this lack of information, we combine the databases drawn from the various dictionaries, on the principle that, if a supine-form is missing in L&S, we can find it in Gaffiot (or vice-versa). This combination requires the alignment of the files, especially for homonyms, and the elimination of redun-

[33] Lewis and Short (1879), encoded in XML by Perseus (*http://www.perseus.tufts.edu/*).

[34] Gaffiot (2016), see *http://gerardgreco.free.fr/spip.php?article47*. Thanks to Gérard Gréco, we had access to the file before its publication.

[35] Georges (1913).

[36] Gérard Jeanneau, *http://www.prima-elementa.fr/Dico*. This Latin-French dictionary is still evolving. For this work, we have used a version of 2013.

[37] The data from Collatinus itself, a short version of Gaffiot, Lewis (1890), and the headwords of the *Pocket Oxford Latin Dictionary*, i.e. Morwood (2012).

[38] Here, we are considering the XML/HTML tags that identify the different entities. Later on, the word 'tag' will have a rather different meaning.

dant doublets. For instance, in L&S, *abscisus* has its own entry with a laconic definition «*P. a., v. abscido*» and is translated in a sub-entry of *abscido*. A supervised program allowed us to do this in a reasonable amount of time. Quantities can be sufficient to distinguish homonyms as *pŏpŭlus* vs *pōpŭlus*, but not always. Sometimes, we have to consider the PoS, as for instance in *a-spergo*, *ersi*, *ersum*, *3*, *v. a.* vs *aspergo*, *ĭnis*, *f.*, or the gender to recognize homonyms, for instance the noun *par*, *paris* which can be masculine or neuter. As a final option, the human reader can use the translations to align the entries.

The last step is to convert the collected information into a file which can be understood by Collatinus. The quantities given by the dictionaries are compared, and if they differ, we choose the form given by the 'majority'[39]. The quantities that can be determined by position are usually not indicated, but the program knows the rules[40] so that it was able to supply the missing quantities to Collatinus. Once again, a difficult step is the reconstruction of the roots: for the verb *a-spergo*, the program builds the form *āspērgo*[41] and the two roots, for the perfect and the supine, *āspērs*[42], while for the noun, it gives *āspērgŏ*[43] and *āspērgĭn*.

This treatment, mostly automated, yields to a lexicon of about 77,000 lemmata, associated with a paradigm and the necessary roots. But some 7,200 additional words were extracted from the dictionaries but not 'understood'. Some of them are useless for Collatinus: for instance, Gaffiot and the elementary Lewis have an entry for *aberam*, which is not a fundamental word. A Latinist should go through this file to determine which words may be useful to complete the lexicon. On the other hand, the process of expanding the variants of the headwords, which was necessary to align the entries of the dictionaries[44], leads to doublets. Most of the doublets caused by the assimilation of a prefix have been tracked down and suppressed. The Latinization of Greek names (e.g. *Ariadna*, *ae* for *Ariadne*, *es*) also caused

[39] In the comparison of quantities, we have to take into account that Georges (1913) and Lewis (1890) indicate only long vowels. The unmarked vowels can be either long by position or short.

[40] A diphthong is usually long (except for the *æ* of *præ* before a vowel, which becomes short). A vowel placed before two or more consonants is long too. A vowel before another vowel is short.

[41] The quantity of the final o is not relevant, because it is given by the word-endings.

[42] In these cases, the two roots are equal, but they usually differ. A difficult example is *ab-sorbĕo*, *bui*, *rarely psi*, *ptum* where we have two different roots for the perfectum, *ābsōrbŭ* and *ābsōrps*.

[43] The rule that says that the final *o* of the nominative is long when the previous vowel is long – see Quicherat (1885: 32), which can be downloaded from Gallica – seems not well followed. We prefer to mark it as common.

[44] For instance, Gaffiot (2016) has *adfero* as a headword, while Lewis and Short (1879) give *affero* with the variant *adf-*. Both are merged in Collatinus to give a single entry.

doublets. But a similarity of *a*/*e* or *us*/*os* is not sufficient to cause a doublet: for instance, *Agylla*, *ae* is an Etrurian city, while *Agylle*, *es* is a nymph. A final group of doublets comes from the singular or plural forms of some words which are chosen as headwords in the different dictionaries. A careful search for all of these doublets is still to be done.

Finally, to avoid long loading times, we split the lexicon into two parts. About one third of it corresponds to the 24,000 words that have been found in the texts lemmatized by *L.A.S.L.A.* It is loaded by default and allows the lemmatization of a large percentage of words in classical texts. The remaining two thirds, 53,000 words, are rarer words and are loaded only on demand. We planned to split the lexicon into more parts, each one specialized in a period of time or a range of semantically similar topics. We are considering this possibility for future versions as it requires that the program is able to load and purge different lexica while running[45].

2.3. *Perspective - Modularity of the data*

The 12th version of Collatinus (C12 here) is still under development. It focuses essentially on lexical and morphological data. Its aim is to handle larger and more precise data to lemmatize specialized corpora. For instance, when having to lemmatize a large medieval corpus, we confronted several difficulties:

- Numerous new words
- Evolution of semantics
- Evolution of graphic uses
- Evolution of paradigms

So, we found that the actual state of Collatinus' data often leads to wrongs results.

2.3.1. *Modules*

Our plan is to collect all the differences between the classical data and those which are required to lemmatize a non-classical corpus, for instance a medieval one. Using a special editor, a new set of data is created, containing all the differences between the classical state of the Latin language and the one in the corpus under study. These differences may appear at various levels: lexicon and translations, inflections, graphic usages, irregular forms.

[45] For the moment, Collatinus loads the data when booting.

This data is zipped into a package with the *.col extension. Once created, this module can be uploaded to the web-site of Collatinus. Then, other users can download it and install it in their C12.

Then, when lemmatizing a medieval text, the C12 user selects the medieval module. First, C12 reads classical data. Then, from this medieval module, new words are added. If a word already exists in the classical data, it is replaced by the medieval one. Often, the medieval word has few differences with the classical one: for instance, a new meaning. Sometimes, a word only needs to change its flexional paradigm, or one of its stems. But it may also be completely different. The same principle is applied for inflexions, irregular forms and graphic variants.

Orthographic variants: C12 adds a new data file, named *vargraph.la* which stores the orthographic particularities:

- Classical orthographical variants, e.g. *cu*/*quu* (*cum*/*quum*)
- Medieval orthographic variants are numerous, e.g.:
  - ligatures *q*;/*que*
  - phonetics *mpn*/*mn* (*dampnum*); *ß*/*ss*
  - tilde *ã* or *ā*/*an*, *am*

For medieval modules, the problem of the lexicon is very acute. Medieval corpora introduce many anthroponyms, toponyms, Latinization of local words: Celtic, Germanic, Spanish, etc. And these new words depends strongly on the considered corpus. For instance, the words derived from the vernacular languages will differ in Spain and in Germany. Thus, specific specialized lexica may be needed for each corpus[46].

A real difficulty is the survival of the anterior states of language. Classical authors could not know words to be created during the following centuries, but subsequent authors did know classical authors, sometimes very well. We need to be very careful when editing a classical word: classical senses may survive in medieval texts.

2.3.2. *The editor:* Ecce

Ecce (Ecce Collatinistarum Communitatis Editor) aims to create modules for C12. *Ecce*'s interface has four tabs: Lexical Modules, Lexicon,

[46] Another possibility would be to use an expandable personal lexicon, but it would remain 'private' and every scholar would have to develop their own lexicon. A third way could be to gather a huge data-base, but at some point a trade-off has to be made between the size of the base and the responsiveness of the program.

(ortho)Graphic variants, Irregulars. When launched, the first tab, Lexical Modules, is selected. On the left side, the user can choose the module to activate, deactivate, delete, generate or install. He can also choose other modules to extract data he will be able to add to the new module. Let us call them 'tank modules'. A very important tank is lem_ext, named 'extension'. When the new module and tank modules are selected, the user clicks the 'Activate' button. If this modular approach is adopted and widely used, the number of tank modules will grow, and building new modules will be easier and easier.

The Lexicon tab then appears. Latin text, and navigation buttons: beginning, backward, forward, previous failure, next failure, end. To feed the lexicon, the user clicks the 'next failure' button. *Ecce* goes on lemmatizing the text word after word, and stops when the lemmatization fails. The word is displayed, solutions, if any, are searched for in tank modules, so that you can check them, edit one of them, and add it. You can also, on the right side, edit a new lemma from scratch. If the lemma exists with another spelling, or another flexion, the two other tabs can be used. When the new data is validated, it is a good practice to go back to the beginning, and restart the lemmatization, to check if the edition is correct.

2.3.3. *Usages*

Collatinus is a lemmatizer, and its main usage is lemmatization. The modular organization of C12 allows a more precise lemmatization of non-classical or special corpora: author, place, topic. Just as Mario Nizzoli, in 1734, released a *Thesaurus Ciceronianus*, a Ciceronian C12 module could be created, uploaded to the web site of Biblissima and then downloaded by any other user who may be interested. It could be interesting to test it for teaching tasks:

- Provide a tiny module for a short Latin text;
- Ask students not to translate a text, but to develop the module which fits to this text, using *Ecce*.

3. L.A.S.L.A. - *Tagger*

As in every language, forms in Latin can be ambiguous. This ambiguity can be found at different levels. On one hand, in a declension, different cases can have the same form for the same word. A familiar example is the first

declension with the word-endings for the nominative and ablative which look the same but are different. On the other hand, some forms of different lemmata may coincide. For instance, *oris* is both a form of *ora*, *ae* and a form of *os*, *oris*. It can be useful to apply the usual techniques of disambiguation to propose the most probable analysis first. Obviously, one also has perfect homographs, as the two *populus* or the two *levis*, that share the same inflected forms and are completely undistinguishable.

3.1. *Statistics on lemmatized texts*

Methods based on 'hidden Markov models', commonly known as probabilistic taggers, are widely used for disambiguation of the modern languages[47]. They assign to each form a tag that reflects its morphosyntactic nature and sometimes its syntactic function. The PoS is often used as a tag, sometimes complemented with some other pieces of information. The method relies on the hypothesis that the sequences of tags are characteristic of the language and do not depend on the text, whatever the subject is and whoever the author. Knowing the frequencies of the pairs (form, tag) and the frequencies of the sequences of three tags (second order Markov process), one can compute the probabilities associated with each of the possible sequences of tags for the sentence. Then one assumes that the most probable sequence is the correct one, or at least the more likely one[48]. Very high accuracies are obtained with modern languages, where the order of the words in the sentence is rather fixed. It is not demonstrated that the same fidelity can be reached with Latin, where the order of the words is free, or at least much freer than in modern languages.

On the other hand, in the last decade, new methods appear which are based on Artificial Intelligence (AI) and, often, Neural Networks. They give better results than *HMM* with modern languages. However, the evaluation of the error rates is sometimes questionable, especially for Latin, as the 'tasks' for lemmatization and PoS tagging are separated. If an AI program analyzes the form *cum* as the accusative of a substantive *cum* (2nd declension neuter, obviously), it could be counted as a correct lemmatization[49]. Anyhow, even if the error rate of AI methods is lower, it is still far from the aim of the phi-

[47] See for instance Rabiner (1989).

[48] For a more detailed description of a tagger, see Schmid (1994), available at *http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/*.

[49] Except that *L.A.S.L.A.* knows several lemmata *cum* that have an index 1, 2, etc.

logist who wants to reach a golden standard result with no errors. Moreover, the AI methods require more computational power than *HMM* and behave as a magical black box. What we need is a practical and robust tool giving understandable results. *HMM* relies on a simple hypothesis and then one needs only Bayesian probabilistic calculations. The principle of *HMM* is easy to understand and to follow.

To begin, one has to choose the tag-set and to do some statistics on a training corpus[50]. A trade-off has to be made for the tag-set. If the tag-set is too small, its disambiguation capabilities will be restricted: for instance, if we just consider the PoS, we will not be able to distinguish the two *oris*, which are both nouns. On the other hand, if the tag-set is too large, the statistics on a finite corpus will be poor. As a training corpus, we used texts lemmatized and analyzed by *L.A.S.L.A.*[51]. The files we used count slightly fewer than two millions words, each form being associated with a lemma and a code that gives the full analysis[52]. This code cannot be used as a tag, because it would lead to an excessively large tag-set with more than 3,000 different tags. We cut from these codes some redundant information: for instance, for verbs, the type of conjugation is associated to the lemma and the different persons have different word-endings. We choose to restrict the tag to the PoS associated with the mood for verbs and with the case and number for the declined forms[53]. For each triplet (form, lemma, tag), we counted the number of occurrences in the corpus. We obtained a file with about 150,000 entries. And we did the same for the sequences of three tags, obtaining a file with 235,000 entries. These numbers are the primary information sources for the implementation of a probabilistic tagger.

3.2. *Double lemmatization*

With the statistical data extracted from the texts lemmatized by *L.A.S.L.A.*, we have developed a lemmatizer-tagger. The first version of the program began with a sequential lemmatization. It first looked if the form

[50] It is not a training corpus in the sense used today in Neural Networks and AI, even if Schmid (1994) called it training. It is a fully annotated corpus on which statistics are performed in a perfectly mastered way.

[51] We thank Gérald Purnelle for his help in the preparation of these texts, the list of which can be found at: *http://web.philo.ulg.ac.be/lasla/textes-latins-traites*.

[52] The gender is absent in the corpus we have treated.

[53] The number is needed only to distinguish some forms, mainly in the fourth declension and could be omitted. A lot of tests should be done to optimize the tagset, which have not yet been done.

was found in the file containing all the forms of the texts lemmatized at *L.A.S.L.A.* (form lemmatizer). If a word was not found in the file, the code sent a request to Collatinus which was supposed to run in the background on the same computer. Collatinus answered with the possible lemmatizations of this form. If Collatinus was not able to answer (either because it was not running or because it did not recognize the form), then the program asked the philologist who was supposed to supervise the process and waited for an answer.

However, since it has been found that this sequential and conditional lemmatization induces errors, we turned to parallel lemmatization[54]: the lemmatization is always done both by Collatinus and by a form-lemmatizer based on the *L.A.S.L.A.* data. The cause of the errors in sequential lemmatization was the fact that as soon as one solution was given by the form-lemmatizer, the program assumed that all the solutions were given. But consider, for instance, nouns where dative plural and ablative plural have the same form. It occurs frequently that for some lemmata only one of these two cases has been found in the *L.A.S.L.A.* texts. As a consequence, the program assumed that only one tag could be associated with this form, reducing erroneously the tag-sequences to be tried. Then the error propagates due to the mechanism of the probabilistic tagger, forcing the philologist to correct several analyses in the sentence.

The double lemmatization requires extra work to match, if possible, the lemmata used by *L.A.S.L.A.* with those of Collatinus and to remove the duplicates. The correspondence between the two lexica is rather delicate. Just to give a few examples, *L.A.S.L.A.* distinguishes the two *et*, conjunction or adverb, while Collatinus has a single lemma *et*, with two possible PoS. On the other hand, Collatinus considers (up to now[55]) that *poplus* is a lemma, while *L.A.S.L.A.* considers it as a contracted form of POPVLVS_1[56]. The correspondence has been established by asking Collatinus to lemmatize the

[54] Independently, Patrick Burns developed concurrent lemmatization (see elsewhere in this volume).

[55] In the last version of Collatinus, we have introduced the possibility of giving several forms for a lemma, but we have not yet reviewed the whole lexicon to group those forms.

[56] As a matter of fact, the lemmata in the lexicon of *L.A.S.L.A.* are given in uppercase, with a disambiguation index if necessary. By convention, proper names and the associated adjectives have always an index, N and A (sometimes O, if there are homonyms as *Pallas*, *adis*, f. and *Pallas*, *antis*, m.). Otherwise, the index is present only when there are homonyms and is an integer (1, 2, etc.). In Collatinus' lexicon, the lemmata are written as usual: in lowercase, with an index if there are homonyms (for historical reasons, the index 1 is generally omitted – which is probably not a good idea) and with a capitalized first letter for proper nouns and adjectives.

list of forms found in the *L.A.S.L.A.* files (as mentioned above, the form is associated with a lemma and a code giving the PoS and the analysis). The PoS and the analysis given by Collatinus were compared with the *L.A.S.L.A.* code. In the best case, the match is unique and perfect, and then the two lemmata are linked. Otherwise, a list of suitors is established and an algorithm tries to sort it out. At the end, a manual check has to be done[57].

As mentioned above, Collatinus does not split the enclitics *-que* or *-ne* if the word is recognized as a whole. So this possibility has been added in the editor of the annotated text. On the other hand, Collatinus does not search for compound verbal forms, so *amata est* will remain a participle followed by a verb, just as *fortis est* is an adjective followed by a verb. However, in the double lemmatization, if the compound form has been seen in the *L.A.S.L.A.* corpus (which is the case for *amata* <*est*>) then the program will offer this solution as the preferred one. This particularity may lead to apparent inconsistencies as, for instance, *est amatus* will be recognized as a compound verbal form while *amatus est* will not. But the philologist will have the ability to add any compound forms.

3.3. *Disambiguation*

The results are sorted by frequency, and a first attempt for the lemmatization of the text is obtained by putting together the most frequent individual lemmatizations. This first attempt considers the forms as isolated, independent of their neighbours, and its error rate is expected to be about 20%[58]. Then, the tagger enters play to take into account the context with a simple statistical model. We have made very few trials: the obtained accuracy was about 88% (exact result, i.e. correct lemma and analysis) and the lemma is the correct one in 96% of the cases. As a last step, the philologist can check all the lemmatizations and, if needed, correct them.

As already mentioned, we are not interested in having the lowest error rate for the tagger itself. The only aim is to facilitate the philologist's work with a convenient tool. We did not sacrifice part of the annotated corpus to keep a 'test corpus', so the evaluation of the tagger has to be done on excerpts

[57] As one has to deal with a few thousand lemmata, some errors remain in the look-up table. Some correspondences are also missing.

[58] This figure is evaluated on the training corpus. If we consider the most frequent lemmatization of each form and sum the corresponding numbers of occurrences, we obtain about 80% of the total number of lemmatized forms.

of this same corpus. Some will argue that it is cheating, but laws about entropy show that, when the corpus is large enough, the computer cannot remember all the sequences it has seen and the results will not change significantly. More interesting is the evaluation of the number of changes that the philologist did between the first attempt by the tagger and the final file when facing a completely new text. For example: an extract of Ausonius' *Mosella* with a total of 1,826 tokens[59]. Considering only the lemma and its index, we have observed 218 modifications of which 31 were due to changes in the text: the philologist erased a verse and corrected some OCR errors (e.g. *lam* corrected to *Iam*, *amatam* for *afflatam*). As the lemmata given by Collatinus are in lowercase, a normalization (to the uppercase lemmata used in *L.A.S.L.A.*'s corpus) is needed when the lemma is new to *L.A.S.L.A.* Such a normalization is not related to an error of the tagger[60] and the corresponding cases are excluded from the analysis. In the end, the mistakes of the tagger were 125, an error rate of about 7%. This sample is too small to analyze it statistically, but it turns out that a significant part of the mistakes are due to the ambiguity between participles and adjectives (in both directions, for instance, *compositus* vs *compono*, or *fulgo* vs *fulgens*) and sometimes between noun and adjective (for instance, *Alpinus*). Some errors are due to the mishandling of the capital at the beginning of a verse and could be corrected. More difficult is the case of the enclitic: we have chosen that if the form exists as a whole, we do not try to strip off the enclitic *que*, for instance in *quaque* which, sometimes, has to be split in *qua-que*. Another difficulty comes from *L.A.S.L.A.*'s fine-grained lemmatization: a simple form as *ut* is connected to four lemmata and *quo* to five (each lemma is associated with one PoS). A second analysis on Prudentius' *Psychomania* gives similar results on a sample of 6,133 tokens, and most of the errors are due to the uncertainty between participles and adjectives.

With a probabilistic tagger, it is interesting to note that, although the 'context' is described by the sequences of three tags, the choice of the best tags is done only at the end of the sentence or of the text. In principle, all the possible sequences of tags are considered, but many of them are skipped[61]. In any case, the choice of a tag can influence the analysis of another word further than two words apart. Conversely, it is important to know how far a 'wrong'

[59] This text is part of the work of Marc Vandersmissen for a research project F.R.S.-FNRS-PDR FNRS-2019: Motifs textuels ovidiens et littérature latine tardo-antique.

[60] We could have done the transformation a priori, but we wanted to single out these new lemmata. It allows the philologist to preserve the coherence of the lexicon.

[61] For details about the pruning method, see Schmid (1994).

analysis would spread its effect. An examination of the list of words shows that slightly less than 40% of the forms are associated with a unique analysis (thus a single tag). Thus, the probability of finding two such forms consecutively is 15%, which means that such a pair should be found, on average, every 6 or 7 words. Such a pair splits the text because these unique tags are present in all the tag-sequences, forming fixed points. The fact that we use a second order Markov model implies that the tags that come after a fixed point do not depend on the tags before. Therefore, if the tagger gives the wrong tag to a word, this error will affect some of the following words, but not many. Roughly speaking, it can affect seven words, on average. Obviously, it may happen that a longer series of words can be found between the fixing pairs.

One can imagine a 'multiplex disambiguation' with another method, which would allow for cross-checking the results. A huge benefit[62] can be achieved if the methods differ sufficiently, even if they are trained on the same corpus. Neural networks and AI are presently very promising in this direction. However, their outputs should be cleaned from the absurdities they can contain. For instance, it has been seen[63] that the output of a neural network program contains 'Cum ; cvm ; NOM2 ; Case=Acc|Numb=Sing': the form *cum* is analyzed as the accusative singular of a noun (lemma) *cvm* following the second declension. Clearly, some constraints have to be added to the program. One of the problems with AI methods (in general, this is not specific to this process) is that nobody knows why the program chose one solution instead of another one. This is not the case with *HMM* where the reason for the choice is always that a probability is larger than another one. By looking closer at these probabilities, it should be possible to associate a 'confidence level' to any result. If the larger probability differs from the second one by a small amount, then the confidence level is poor and the philologist should check the result twice. But this remains to be done, and it raises fundamental questions. For instance, what counts as a small difference in probabilities? How can the program, which does not understand what it is reading, know where the difficulties are?

From a more theoretical point of view, it would be interesting to study the sequences of tags to search for correlations. If the order of the words were completely free, one would expect no correlation at all and the tagger would

[62] However, for the philologist who wants zero error, it will not be sufficient. A careful and tedious check will always necessary.

[63] We shall not mention where.

give the same result as a frequency-based lemmatizer. The correlations and the efficiency of the tagger are linked, and the study of the former will give information on the limits in the accuracy. As for the previous point, this work remains to be done. And both points may well be correlated.

3.4. *Comparison*

The content of this section is mainly subjective and speculative. As a matter of fact, nobody will ever lemmatize the same text with each of the two proposed tools. It would mean to do twice the job with no benefit.

The traditional procedure for preparing *L.A.S.L.A.* files is semi-automatic: the lemmatizer proposes to the philologist all the analyses known by the *L.A.S.L.A.* dictionary for each of the forms in the text. The philologist selects the correct analysis, or inserts manually the correct analysis, if needed. The analyses are proposed in an order depending only on the morphosyntactic code, and not on their frequency or on their likeliness in that context.

On the contrary, the tagger proposes the most probable analysis, and therefore the role of the philologist is essentially to correct the results of the analysis proposed by the tagger. This accelerates the work, but also changes the kind of human mistakes that occur. On the one hand, the traditional *L.A.S.L.A.* procedure induces human mistakes caused by the similarity of the possible morphosyntactical analyses, represented by similar alphanumerical codes. The philologist may mistake an accusative for a nominative, or an ablative for a dative, or pick the wrong mood or tense for a verb. It is highly unlikely that, in case of homographic forms, like for instance *salis* (2nd person of the present indicative of *salio*, or genitive from *sal*), the user would select the verbal analysis instead of the nominal or vice versa. On the other hand, the tagger may be lead to such an erroneous choice, but the mistake shall remain unseen by the philologist. Indeed, since the philologists expects, for instance, a genitive, he may think that the form is unambiguous, because the possible analysis as the indicative of the verb *salio* may not occur to him in that context. Therefore, attention may lapse, and the tagger's mistake may be left unseen. With the traditional method, the user would hardly mistake the analysis of the verbal form with the one of a substantive. When using the tagger, on the contrary, the philologist is more conscious of the necessity of checking the proposed solution for clearly potentially ambiguous forms, such as datives/ablatives, and will thus probably pay high attention to the correction. At the moment it is not possible to verify which of the methods causes

more human mistakes, therefore it is not possible to draw any conclusion on this topic. The two methods are synthetically compared in Table 1:

L.A.S.L.A. *Encoding Initiative*	*Collatinus*-L.A.S.L.A. *tagger*
PREPARATION OF THE TEXT	
The text is prepared by an operator from *L.A.S.L.A.*	The text is loaded directly in the program, with a minimal standardization in the splitting of lines/paragraphs/chapters/etc.
PROS: Initial control of the edition, of the splitting, etc.	PROS: The philologist can start to work immediately. He/she has the possibility to correct/change the references and the text during the lemmatization.
CONS: Possible delays, independent of the will of the philologist.	CONS: Possible use of texts (for instance, available on internet) without any indication of the reference to the edition.
Comment: The tagger offers more flexibility, but requires more care and knowledge about the mechanisms of reference and the choice of the edition.	
CHOICE OF THE ANALYSES	
Proposition of all the known analyses, without any priority.	Proposition by default of the 'best' solution, together with all the other possible analyses.
PROS: The philologist has to read carefully all the given analyses to select one of them.	PROS: Fast processing and several cases are solved automatically.
CONS: Constant concentration (even for the simple cases). Slower treatment.	CONS: The default choice may be wrong and still escape the philologist's attention.
Comment: An evaluation of the error rates achieved with the two methods has to be done. It is a difficult task from a methodological point of view because it is not the philologist who is evaluated, nor the complexity of the considered text.	
DICTIONARY	
The dictionary is based on the Forcellini. The addition of new lemmata is controlled by the PI at *L.A.S.L.A.*	The dictionary is based on Gaffiot and Lewis & Short. A personal lexicon is added.
PROS: Internal coherence for the whole corpus of *L.A.S.L.A.* and also in the propositions given in the program.	PROS: More extended lexical base. New entries can be added simply. Distinction between lemmata known by *L.A.S.L.A.* (in uppercase) and those from Collatinus (in lowercase).
CONS: Frustration of the manual insertion of new lemmata/analyses. Risk of error in the repetition of this task.	CONS: Risk of incoherence with the *L.A.S.L.A.*'s corpus. Possibilities of unseen doublets or errors in the indices.
Comment: Strong advantage in the speed of the tagger. If the personal dictionaries were checked and inserted in the *L.A.S.L.A.* dictionary, it would increase its size rapidly.	

L.A.S.L.A. *Encoding Initiative*	*Collatinus*-L.A.S.L.A. *tagger*
Final treatment	
Usually, the treated text is checked (often by another philologist). Correction of the printed index and insertion of them by an operator. Production of the final file, by an operator, at the end of the process (for instance, several books).	The generation and the correction of the index are left to the philologist. The output file is immediately in the standard APN format which makes it usable at once.
Pros: Rigorous verification, in part on printed material.	Pros: The file can be studied as soon as it is completed, without having to wait for the completion of the entire work (if formed of several books).
Cons: Possible delays in the processing (in part independent of the philologist's will).	Cons: Risk of a less careful verification.
Comment: Working with the tagger appears to be a more personal work, with more responsibilities but more independence and flexibility.	
Conclusion: For a work to be completed in a finite amount of time (e.g. for a PhD thesis), the speed of the tagger is a key element. The philologist at work has a complete control of all the steps, but also (as a consequence) a larger responsibility. On a longer time scale, the traditional method is safer for the coherence of the *L.A.S.L.A.* corpus. However, nothing impedes an extra checking of the output of the tagger (by a second philologist) to ensure its quality. The coupling of the two methods could lead to a significant increase of the *L.A.S.L.A.* corpus and dictionary.	

Table 1. *Summary of the differences between the two NLP tools.*

4. *Conclusion*

In this article, we have presented part of the work going on at the *L.A.S.L.A.* and in the Collatinus' development group. We have also put some emphasis on their collaboration and compared the two approaches for the lemmatization and analysis of new Latin texts. We underline the pros and cons of each of them. A kind of trade-off has to be found between speed and precision.

However, the required precision or the tolerable error rate may depend on the envisioned application and remain an open question. Obviously, a perfect lemmatization, with no error at all, is desirable, but probably not needed. Most of the applications are of statistical nature, which means that they contain an intrinsic degree of uncertainty which can often be determined with error-bars, but seldom given or understood. In this context,

what is (or would be) the consequences of a few remaining errors? It is difficult to evaluate, but even more difficult to measure. Due to the lack of realistic objectives (with upper limits on the acceptable error rate, for instance), we stick to perfection.

Acknowledgements

We would like to warmly thank Bret Mulligan (Haverford College) for carefully reading our text and for his very pertinent remarks.

References

Blanco, X. and Bogacki, K. (2014), *Introduction à l'histoire de la langue française*, Bellaterra, Barcelona.

Denooz, J. (1978), *L'ordinateur et le latin. Techniques et méthodes*, in «Revue de l'organisation internationale pour l'étude des langues anciennes par ordinateur», 4, pp. 1-36.

De Meo, C. (2005), *Le lingue tecniche del latino*, Pàtron, Bologna.

Forcellini, E. (1864, [1771[1]]), *Totius Latinitatis Lexicon* [ed. by F. Corradini and G. Perin], Tipografia del Seminario, Padova.

Fögen, Th. (2011), *Latin as a technical and scientific language*, in Clackson, J. (2011, ed.), *A Companion to the Latin language*, Wiley / Blackwell, Oxford, pp. 445-463.

Gaffiot, F. (2016). *Dictionnaire latin-français par Félix Gaffiot, revu et corrigé sous la direction de Gérard Gréco* [available online at *http://gerardgreco.free.fr/spip.php?article47&lang=fr*].

Georges, K.E. (1913), *Ausführliches lateinisch-deutsches Handwörterbuch*, Hahnsche Buchhandlung, Hannover.

Lewis, Ch. T and Short, Ch. (1879), *A Latin Dictionary Founded on Andrew's Edition of Freund's Latin Dictionary*, Clarendon Press, Oxford.

Lewis, Ch. (1890), *An Elementary Latin Dictionary*, American book company, New York / Cincinnati / Chicago.

Morwood, J. (2012), *Pocket Oxford Latin Dictionary*, Oxford University Press, Oxford.

Ouvrard, Y. and Verkerk, Ph. (2014), *Collatinus, un outil polymorphe pour l'étude du latin*, in «Archivum Latinitatis Medii Aevi», 72, pp. 305-311.

Ouvrard, Y. and Verkerk, Ph. (in press), *Collatinus & Eulexis. Latin & Greek Dictionaries in the Digital Ages*, in «Classics@».

Philippart de Foy, C. (2014), *Nouveau manuel de lemmatization du latin* [available online at *http://hdl.handle.net/2268/162433*].

Quicherat, L. (1885, [1846][1]), *Nouvelle prosodie latine*, L. Hachette, Paris.

Rabiner, L.R. (1989), *A tutorial on hidden Markov models and selected applications in speech recognition*, in «Proceedings of the IEEE», 77, 2, pp. 257-289.

Schmid, H. (1994), *Probabilistic part-of-speech tagging using decision trees*, in *Proceedings of the International Conference on New Methods in Language Processing*, UMIST, Manchester, pp. 44-49.

Philippe Verkerk
Laboratoire de Physique des Lasers, Atomes et Molécules
Université de Lille
Bâtiment P5
F59655 Villeneuve d'Ascq Cedex (France)
Philippe.Verkerk@univ-lille.fr

Yves Ouvrard
Retired professor of 'Éducation Nationale'
Yves.Ouvrard@collatinus.org

Margherita Fantoli
Laboratoire d'Analyse Statistique des Langues Anciennes
Université de Liège
Place du 20 Août, 7
B-4000 Liège (Belgique)
mfantoli@uliege.be

Dominique Longrée
Laboratoire d'Analyse Statistique des Langues Anciennes
Université de Liège
Place du 20 Août, 7
B-4000 Liège (Belgique)
dominique.longree@uliege.be