Collatinus & Eulexis : Latin & Greek Dictionaries in the Digital Ages.

Yves Ouvrard, Philippe Verkerk

Collatinus & Eulexis

Latin & Greek Dictionaries in the Digital Ages.

Yves Ouvrard & Philippe Verkerk

> Words, which are the dress of thoughts, deserve surely more care than clothes, which are only the dress of the person.
>
> Philip Stanhope, 4th Earl of Chesterfield

Collatinus and Eulexis are free open-source programs, designed to lemmatize Latin or Greek texts and to open the digital dictionaries at the proper page. Collatinus[1] has been originally developed by Yves Ouvrard for teaching. It allows to generate a complete lexical aid, with a short translation and the morphological analyses of the forms, for any text which can be given to the students. The program also allows to search for a word in the dictionaries, either in a digital form or as images. Eulexis is newer and is intended as Collatinus' counterpart for Greek texts, though it is based on different mechanisms. As any open-source program[2], Collatinus and Eulexis can be tuned to meet any particular problem. Both programs have an on-line version[3], developed with the help of Régis Robineau, webmaster of the Biblissima project[4]. But, we consider here the features of the off-line versions.

1. Lemmatizer

The understanding of highly inflected languages as Greek or Latin is made difficult by the forms that a word can take, which can vary a lot. So the first thing to do when reading a text is to lemmatize each word, that is to say identify the root-word that has given this form, which can be

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

2 Written in Qt 5.2. The applications are designed with an intuitive GUI. The sources are available: https://github.com/biblissima/collatinus and https://github.com/PhVerkerk/Eulexis_off_line. For Collatinus, we also include a server (on an internal port of the computer) which allows a "command line" use of the functionalities.

3 http://outils.biblissima.fr/en/collatinus-web and http://outils.biblissima.fr/en/eulexis-web

4 Biblissima — *Bibliotheca bibliothecarum novissima* — is an observatory for medieval and renaissance written cultural heritage, developed through the French "Investissements d'avenir" programme. http://www.biblissima-condorcet.fr/en

ambiguous. The human reader is helped in this task by the context and the meaning of the sentence, while the computer has to deal with the form itself. To solve the problem of lemmatization, two different approaches are possible. The first and simplest one is based on a list of known forms which are properly lemmatized and analyzed. The second method reproduces what the human reader does, trying to split the form as a root associated with a standard word-ending.

1.1 Form-Lemmatizer: Eulexis

As mentioned above, the simplest way to build a lemmatizer is to use a list of forms. The main difficulty is to set-up the list of all possible forms or, which seems more feasible and reliable, the list of the attested forms. Each form has to be associated with a root-word and also with the morphological analysis (or analyses). To give a hint of the meaning of the word, a short translation can be given too.

For ancient Greek, we did not try to build the list of forms. We found it in *Diogenes*[5], and use it with the permission of the author. The list was built with tools and texts from the Perseus project[6]. It counts about 912,000 forms, associated with more than 116,000 root-words and 1,550,000 analyses[7]. The original list gave, for each form, the short English translation of the root-word and also some additional data. We simplified it by splitting the two functions, lemmatization and translation. This operation has two advantages: it reduces the size of the file and allows to give a short translation of the root-word in different languages. To demonstrate this ability, we have translated the English translation into French and German using Google-trad©. Clearly, this is just a first step[8] and all the translations should be corrected by a team of Hellenists.

5 https://community.dur.ac.uk/p.j.heslin/Software/Diogenes/

6 http://www.perseus.tufts.edu/hopper/

7 All the figures are rounded. We have no precise value of the number of analyses because some lemmatizations have multiple analyses as, for instance, “neut nom/voc/acc pl” which counts as one.

8 There are several problems in the translations. First of all, the original translation in English is not very precise (the article ὁ is translated as “lentil”; ὅς as “yas” and εἰμί as “sum”). On top of that, the translation from English to any other language suffers from the ambiguities. For instance, φώκη is naturally translated as “seal” which in turn became “*joint*” in French. The clever techniques of disambiguation which are used for texts do not work for lists of single words.

The Greek words in the list are transliterated using betacode. To alleviate the problem of accent and of breathing, these diacritic signs are not taken into account in the search of a form. The case of the iota subscript may be discussed but, as it is encoded with a special character, it is treated as any other diacritics. Forgetting the “decoration” of the forms reduces their number to 853,000. In the case the Greek word is typed with Latin characters, it allows a simple match. On the other hand, if the word comes from a Greek text, with its diacritic signs, this search gives too many results. The exact match, if it exists, appears first and is red-colored. An option has also been introduced to show only this exact match, as seen in Figure 1.

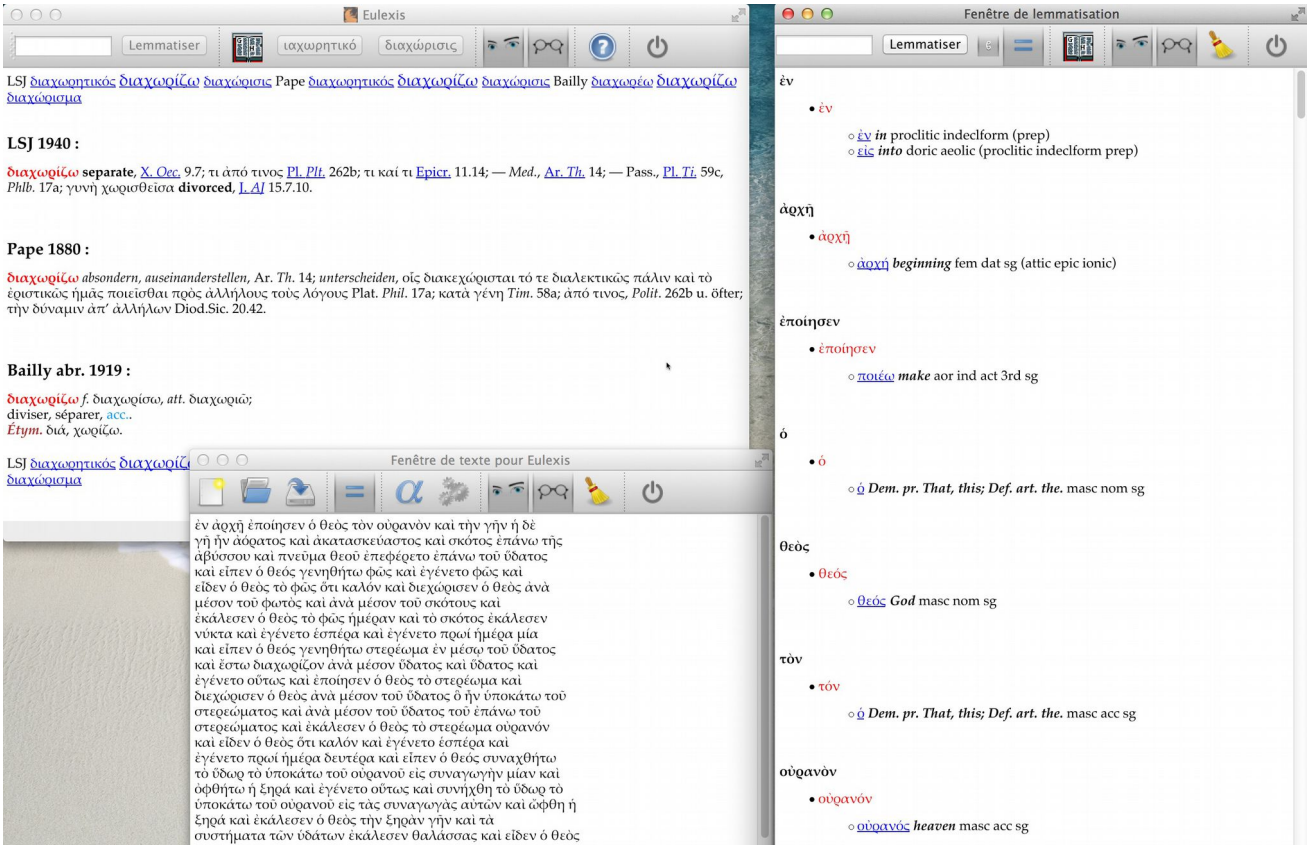

Figure 1: Eulexis can display up to three windows. Its main window, top-left, is devoted to the consultation of the dictionaries. The two other windows are optional. One of them contains the text under study (bottom-left) and the other one (right) presents the result of the lemmatization. During the search, the diacritics are not taken into account, but here we have chosen to display only the exact match, in red.

1.2 Combining Lemmatizer: Collatinus

The limitation of a form lemmatizer is that it is not able to guess anything for a form that has not been found and analyzed before, even if the root-word is known. A more clever method would be to use the declensions or the conjugations endings to construct all the possible forms. This is the way Collatinus works for Latin. It should be possible to adapt this scheme to ancient Greek, but we do not have the knowledge to do that. The advantage of a program as Collatinus is that it is able to recognize forms not yet attested as soon as the root-word is known. It is also easier to improve its base of knowledge: adding the data for a new root-word allows to recognize immediately ten or more (even a hundred, for verbs) forms. Obviously, a program as Collatinus "knows" a lot of forms that are not attested in the texts that have survived[9].

1.2.1 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 a paradigm for the inflection. 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: it has one file that gives the word-endings and the construction rules for each paradigm and another one that connects the lemmas to the paradigms and gives also the roots which cannot be calculated. With this data, the construction of the inflected forms is immediate.

However, 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 root and the last one with a known word-ending. And obviously, we have also to check that the two parts fit together (i.e. that they are associated to the same paradigm with a proper matching rule). 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 just made a list of the morphosyntactical analyses which are possible in Latin and coded the analysis with a simple number. As a matter of fact, these analyses are only 416. The number is converted in its human

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

readable form when needed, i.e. for the display as in Figure 2. By the way, this encoding allows also to translate the analysis in different languages[10].

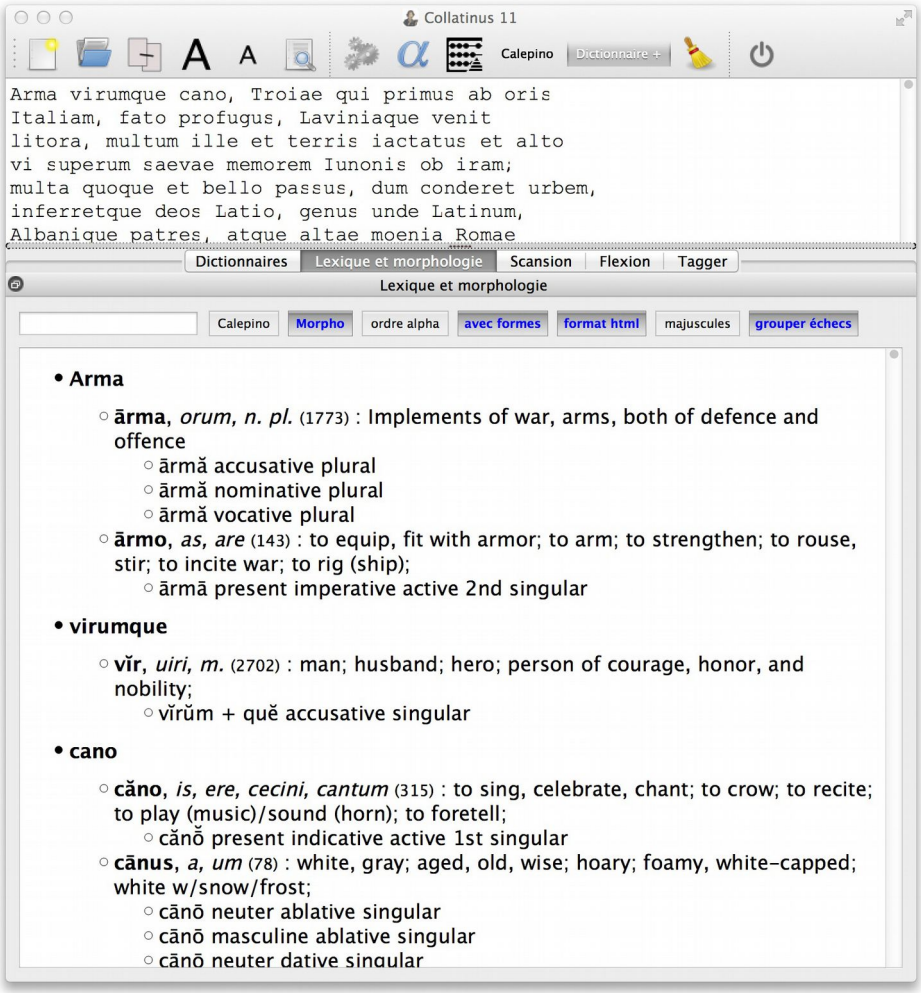

Figure 2: Collatinus' window with the beginning of Aeneid in the upper frame (text under study) and the result of lemmatization (bottom part). Several options are available to display or not some details: here the analyses are given, but one can choose to display only the lemmas with their translation.

A difficulty appears because of the enclitics *-que*, *-ne* and *-ve*. These words are glued at the end of any form, and have to be separated for the lemmatization. In most of the cases, the enclitics *-que* and *-ve* do not lead to ambiguous forms, which is not the case of the enclitic *-ne*.

10 For the moment, French, English and Spanish.

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*. However, the enclitics are not so frequent, thus we 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, it is frequent that a double i appearing in the flexion of a word[11] is written as a single long-i. Some forms of the perfectum 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 pre-verb, assimilation can change the spelling in some cases. It is the case, for instance, of *adfero, adtuli, adlatum* which often becomes *affero, attuli, allatum*[12]. The main assimilations of the prefix are build in Collatinus, so that it avoids the proliferation of forms for the same word.

1.2.2 Distinction between *u* and *v*

Very often the Latinists do not distinguish the letters u and v, and erase the j from the alphabet. However, for counting the syllables or the meters, it is clearly necessary to make the distinction. Thus, Collatinus keeps, in its lexicon and in the word-endings, the two consonants v and j, said to be Ramist consonants[13]. 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 the distinction is the best choice is that the reverse process (restoring v and j) is almost impossible, 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 it[14]. The way to solve the problem is obtained with two steps. In a first step, all the v are replaced by u for

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

12 Gaffiot gives the first forms, while Lewis and Short prefers the second ones.

13 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 Trisino, 1529). See Xavier Blanco i Escoda et Krzysztof Bogacki, *Introduction à l'histoire de la langue française*, Bellaterra, Université autonome de Barcelone, coll. « Documents » (n° 104),2014, p. 160, n. 24 and p. 161.

14 In the worst case, the editors write the capital-U as V. It is not unfrequent to find Vnde at the beginning of a sentence or to meet Vlixes in some texts.

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*[15]. However, some care is taken to avoid wrong lemmatizations of *voluit* or of *volvit*. If the form given in the text contains one (or more) v, it is taken into account by the elimination of the reconstructed forms with a different number of v.

An other group of u are not “real” vowels: it is the case of *suavis* and of *sanguis*. It is also the case for the group “qu”, but in this last case 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 *ām͡bĭgŭīs*[16]. It would have been somehow chocking to write *svavis* or *sangvis* to stress that these words have only two syllables. Instead, we use the expunctuated-u and write *sụāvĭs* and *sāngụĭs*.

1.2.3 Ordering of the solutions

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

Thanks to the statistics made on the lemmatized texts of the LASLA[17], 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 the words. To go back to the previous example, *sus* appears 47 times in the texts of the LASLA, while *suus* appears 7,120 times. As Collatinus does not lemmatize the forms[18], it is

15 *volvit* can also be a form of the present of *volvo*. The meaning of the sentence allows the reader to identify the good 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.

16 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*. For more details, see further the paragraph about scanned texts.

17 “Laboratoire d'Analyses Statistiques des Langues Anciennes” in Liège (Belgium). http://web.philo.ulg.ac.be/lasla/textes-latins-traites. See further the paragraph 2.3 devoted to the disambiguation.

18 We shall come back later on that example through the LASLA tagger (paragraph 4.2).

more difficult to order the two possible cases for the analysis of *suis* as a form of *suus*. We make here a strong assumption: the usage of the cases and number[19] (for nouns and adjectives; replaced by the mood for verbs) does not depend on the particular word. We still take into account the part-of-speech (POS)[20] 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, case that is almost never used elsewhere.

This ordering of the solutions is not sensitive to the context. Its depends only on the form itself and its analyses. According to the statistics done on the lemmatized text of the LASLA, choosing the most frequent analysis gives the good result in 80% of the cases. For instance, in Figure 2, the first analysis given is the correct one (the first mistake is on *oris*). 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, in paragraph 2.3.

1.2.4 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, as the number of occurrences of this lemma in the texts lemmatized by the LASLA. The translations of these lemmata are given in separated files (one for each language) so that the material necessary to inflect the forms is independent from the translations. It also allows to add more languages for the translations without having to duplicate or to change the basic information that rules 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 was containing slightly less than 11,000 entries, which allow to lemmatize a significant part of the classical texts. However, we have decided to boost it by working on the dictionaries in a digital form. The two main

19 Unfortunately, the lemmatization by the LASLA does not give precisely the gender of the adjectives.

20 Mainly: noun, adjective, verb and pronoun, as categorized by the LASLA.

dictionaries we have used are Lewis and Short (L&S), converted in XML by the Perseus Project[21], and Gaffiot, converted in TeX by a team lead by Gérard Gréco[22]. We have also used Georges[23] and Jeanneau[24] in their HTML forms. All these dictionaries are part of Collatinus (see paragraph 2.1.2, for more details). Some extra pieces of information were also used[25].

The first part of the work has been to collect 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 to extract very rich databases. The translations were probably the most difficult part of the job. Subentries, as adjectives that derive from a noun which is the headword, were collected too. The graphical variants, often indicated in an abbreviated form as, for instance, *affĕro (better adf-)*, were expanded and added to the base. This has been done programmatically but checked afterwards, the internal variants, for instance *rĕverto (-vort-)*, being especially difficult to treat, although they are rather intuitive for the human reader. Obviously, one has to face the imperfection of the tagging: some tags as <itype> are missing or do not include all the relevant information.

To deal with this lack of information, we combine the databases coming from the various dictionaries, with the idea that if a supine-form is missing in L&S, we can find it in Gaffiot (or vice-versa). This combination requires a kind of alignment of the files, especially for the homonyms, and the elimination of the redundant doublets. For instance, in L&S, *abscisus* has its own entry with a laconic definition “P. a., v. abscido” and is translated in a subentry of *abscido*. A supervised program allows us to do that in a reasonable amount of time. The quantities can be sufficient to distinguish the homonyms as *pŏpŭlus* vs *pōpŭlus*, but not always. Sometimes, we have to consider the Part-of-Speech, as for instance in *a-spergo, ersi, ersum, 3, v. a.* vs *aspergo,*

21 Charlton T. Lewis & Charles Short, *A Latin dictionary founded on Andrew’s edition of Freund’s Latin dictionary*, Oxford 1879, encoded in XML by Perseus http://www.perseus.tufts.edu/

22 http://gerardgreco.free.fr/spip.php?article47 *Dictionnaire Latin-Français de Félix Gaffiot* (1934) by Gérard Gréco, Mark De Wilde, Bernard Maréchal and Katsuhiko Ôkubo. Thanks to Gérard Gréco, we had access to the file before its publication.

23 Karl Ernst Georges, *Ausführliches lateinisch-deutsches Handwörterbuch*, Hannover 1913.

24 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. A newer version (2017) of this dictionary is now available in Collatinus.

25 The data from Collatinus itself, a short version of Gaffiot, An Elementary Latin Dictionary (English) by Charlton T. Lewis, and the headwords of the Pocket Oxford Latin dictionary.

ĭnis, f., or the gender to recognize the homonyms. As a last chance, the human reader can use the translations to align the entries.

The last step is to convert the collected information in a file which can be understood by Collatinus. The quantities given by the dictionaries are compared, and sometimes they differ in which case we choose the form given by the "majority"[26]. The quantities that can be determined by position are usually not indicated, but the program we used knows the rules[27] so that it was able to supply the missing quantities to Collatinus. Once again, a difficult step is the reconstruction of the roots with the abbreviated indications. For the verb *a-spergo*, the program builds the form *āspērgo*[28] and the two roots, for the perfect and the supine, *āspērs*[29]. While for the noun, it gives *āspērgō*[30] and *āspērgĭn*.

This treatment, mostly automated, leads to a lexicon of about 77,000 lemmata, associated with a paradigm and the necessary roots. But some 7,200 more words have been extracted from the dictionaries and were 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[31], leads to doublets. Most of those due to the assimilation of a prefix have been tracked down and suppressed. Some Greek names are also Latinized (e.g. *Ariadna, ae* for *Ariadne, es*) leading also to doublets. But a similarity a/e or us/os is not sufficient: for instance, *Agylla*, *ae* is an Etrurian city, while *Agylle*, *es* is a nymph. A last

26 In the comparison of the quantities, we have to take into account that Georges and the Elementary Lewis indicate only the long vowels. The unmarked vowels can be either long by position or short.

27 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 an other vowel is short.

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

29 In these case, 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.

30 The rule that says that the final o of the nominative is long when the previous vowel is long (L. Quicherat, *Nouvelle prosodie latine*, 30e édition, Paris 1885, p. 32, which can be downloaded from Gallica) seems not well followed. We prefer to mark it as common.

31 For instance, Gaffiot has *adfero* as a headword, while L&S gives *affero* with the variant *adf-*. Both merge in Collatinus to give a single entry.

group of doublets comes from the singular or plural forms of some words which are chosen as headwords in the different dictionaries. A careful hunting of all these duplicates is still to be done.

At the end, to avoid long loading times, we split the lexicon in two parts. About one third of it corresponds to the 24,000 words that have been found in the texts lemmatized by the LASLA. It is loaded by default and allows the lemmatization of a large fraction of the classical texts. The remaining two thirds, 53,000 words, are rarer words and are loaded only on demand. We had also the project to split the lexicon in more parts, each one specialized in a period of time or a 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[32].

1.2.5 Word-endings and construction rules

As already said, besides the lexicon, Collatinus has an other important file which gives the word-endings and the construction rules. For each paradigm, it gives the list of analyses and the corresponding word-ending. A noun that follows a usual declension has 12 analyses and word-endings, 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[33]. 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 the sole 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

32 For the moment, Collatinus loads the data when booting.

33 The construction rules are also transferred.

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. Other Features

2.1 Opening Dictionaries

The lemmatization of a form is an important step either for the beginner or for the philologist. However, the lemmatizer gives the root-word and a short translation, but no details. To have a more complete information nothing better than a dictionary. For ancient Greek, we have three digital dictionaries. For Latin, the digital offer is larger and is complemented by dictionaries in an image format.

2.1.1 Ancient Greek Dictionaries

In Eulexis, we have three dictionaries, encoded in HTML: Liddel-Scott-Jones (LSJ)[34], Pape[35] and the abbreviated version of Bailly[36]. The larger version of Bailly is under preparation by a team lead by Gérard Gréco[37]. Although LSJ we use has probably its origin in the XML version published by Perseus, it has been converted in HTML long ago, and then was corrected by André Charbonnet[38]. It would have been better to amend the original XML file, but as a personal project one prefers to have an immediately readable result. In the frame of an institutional project, it could be a good idea to check these corrections and to change the XML file accordingly. But this is far beyond the goal of Eulexis. The small Bailly was digitized by André Charbonnet. The Pape

34 Liddell-Scott-Jones - *Greek-English Lexicon* (9e ed. 1940)

35 Wilhelm Pape (1807–1854) - *Handwörterbuch der griechischen Sprache in vier Bänden. Griechisch-deutsches Wörterbuch* (Dritte Auflage 1880, Braunschweig). Only the first two volumes are available. The proper names are in volumes 3 and 4 (*Wörterbuch der griechischen Eigennamen*).

36 *Abrégé du Dictionnaire Grec-français* of Anatole Bailly (1901, 6e éd. 1919) encoded in HTML by Chaerephon, with extra pieces of information coming from the Grand Bailly and from the Dictionnaire étymologique de la langue grecque of Chantraine.

37 http://gerardgreco.free.fr/spip.php?article52

38 Alias Chaerephon: http://chaerephon.e-monsite.com/

dictionary was given to us by Philip Roelli and was corrected by Jean-Paul Woitrain and André Charbonnet.

Thus Eulexis offers Greek dictionaries in three different languages as seen in Figure 1, and also allows to compare them. One can open one or more dictionaries just by giving the root-word. The lemmatization of a form is a separate function, but the obtained lemmata are clickable allowing to reach the definition of the word very quickly. In LSJ, we have worked out the bibliography so that Eulexis can identify the references to an author and a text. The abbreviated references can be made explicit. We are presently trying to extend this possibility to the other dictionaries.

With active correctors as André Charbonnet and Jean-Paul Woitrain, the Greek dictionaries are moving rather quickly. We have thus included a mechanism that allows to update a dictionary very simply. Any user can correct a copy of the files each time he/she finds a mistake. Whenever he/she wants, Eulexis can update its working files and build again the indices. The new version of the dictionary is then available. A point which is not clear yet concerns the way to share the corrections made by different users. A tool as GitHub may be very useful, but we are not sure that a user/corrector of Greek dictionaries is willing to invest energy in such a tool.

2.1.2 Latin Dictionaries

As already mentioned, several digital dictionaries are available for Latin. Three languages are represented for the translations: English with L&S, German with Georges and French with Gaffiot 2016 and Jeanneau. Although the abbreviated version of Gaffiot and the elementary Lewis were used for the lexicon of Collatinus, we did not include them in the program. Some other dictionaries are proposed as images for Italian or Spanish. Quicherat[39] is also available which is very useful for prosody. For these dictionaries, the information is stored in DJVU-format and the page corresponding to the asked word is converted in TIFF for the display.

For the digital dictionaries, each article has been converted in HTML and compressed to save space on disk. When the user looks for a word in the dictionaries (two of them can be displayed simultaneously, an example is given in Figure 3), the program loads the corresponding articles,

39 L. Quicherat, *Thesaurus Poeticus Linguae Latinae*, Paris, 1836. (or the 3rd ed. 1843, which can be found on the web https://books.google.fr/books?id=K9O0AAAAMAAJ&hl=fr)

extracts and displays them. It is interesting to note that the shift from paper to digital display somehow changes things. For instance, in L&S one finds the etymology of the word between square brackets which have been converted in XML with the tag <etym>. However, often the etymology of a word is the same as the one of the previous one, which leads to the short form "[id.]" or, in XML, "<etym>id.</etym>". On a full page, this is not a problem: the interested reader scans the page to get the good etymology. If the display is restricted to the asked article, the short form of the etymology is useless. Thus we have made these etymologies explicit with a program.

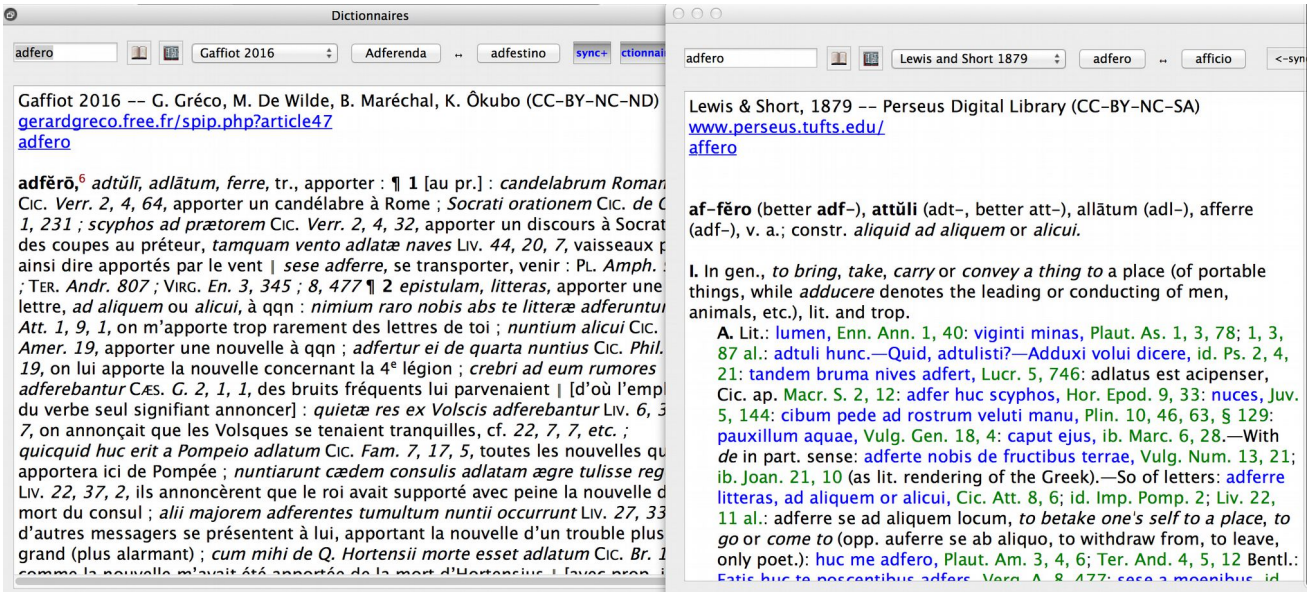

Figure 3: Collatinus can display two windows with an article (or a page) of dictionaries. One can choose to display the same article in two different dictionaries or two independent articles (either from the same dictionary or not).

2.2 Scanning Latin Texts

In the extraction of the words from the digital dictionaries, a special care has been taken to keep the quantities of the vowels. In some cases, the dictionaries do not give the same quantity to some vowel. These words are still to be examined. For some foreign words, the quantities are not

known and we consider these vowels as common[40], which means that they can be short or long, depending on the needs of the author. As soon as the quantities of the canonical forms are known, one can determine all the quantities of all the inflected forms because the quantities of the word-endings are usually given in the Latin grammars[41].

2.2.1 Metrics

Knowing the quantities of the vowels of every forms, Collatinus can scan an entire text. It applies the usual rules for lengthening or elision in the succession of words. As a matter of fact, for dactylic verses, the knowledge of a few quantities can be enough to scan the whole verse. But for other types of verses, it is a much harder task. On the other hand, it may happen that verses or metric patterns are hidden in a text in prose.

When scanning a text, Collatinus replaces the standard form found in the text, by its counterpart with the quantities indicated. When a form has several analyses, which give different quantities for the vowels, all the possible solutions are given. The first one is the most frequent one as for the lemmatization (see 1.2.3 above). The other ones are written between parenthesis. For instance, as seen in Figure 4, the first verse of the Aeneid becomes:

Ārmă (Ārmā) vĭrūmquĕ cănō (cānō), Trōjāe (Trōĭāe) quī prīmŭs ăb ōrĭs (ōrīs ŏrīs)

Knowing that it is a dactylic verse, it would have been possible to choose the proper form and to separate the meters *Ārmă vĭ / rūmquĕ că / nō, Trō / jāe quī / prīmŭs ăb / ōris*. But we have chosen not to do that because it is too specific to dactylic verses. Pede Certo[42] does that in a fantastic way, and they have scanned about 244,000 verses. To be generalized, it would require that the program is able to disambiguate with enough accuracy to choose one of the possible scanned form (see 2.3 below) and/or that it has some information about the used meter, to

40 A common vowel is indicated with both diacritics, macro and breve, on top of it: *ā̆* or *ō̆*. Note that these characters do not exist in Unicode: they are obtained by adding a “combining breve” on the vowel with macron. In the window of Collatinus, the superposition should by fine. But the result of a copy-paste may depends on the text editor and on the selected font.

41 We used the *Grammaire Latine*, A. Cart, P. Grimal, J. Lamaison and R. Noiville (Paris) which is the Latin grammar commonly used in France.

42 Pede certo is a program for the automatic analysing of Latin verses developed by the Università di Udine as part of the Traditio patrum FIRB project. http://www.pedecerto.eu/

indicate the pauses for instance. On the other hand, if one knows that the verse is an hexameter, some solutions can be excluded: for instance, the indicative present *vĕnĭt* does not fit at the end of the second verse in Figure 4.

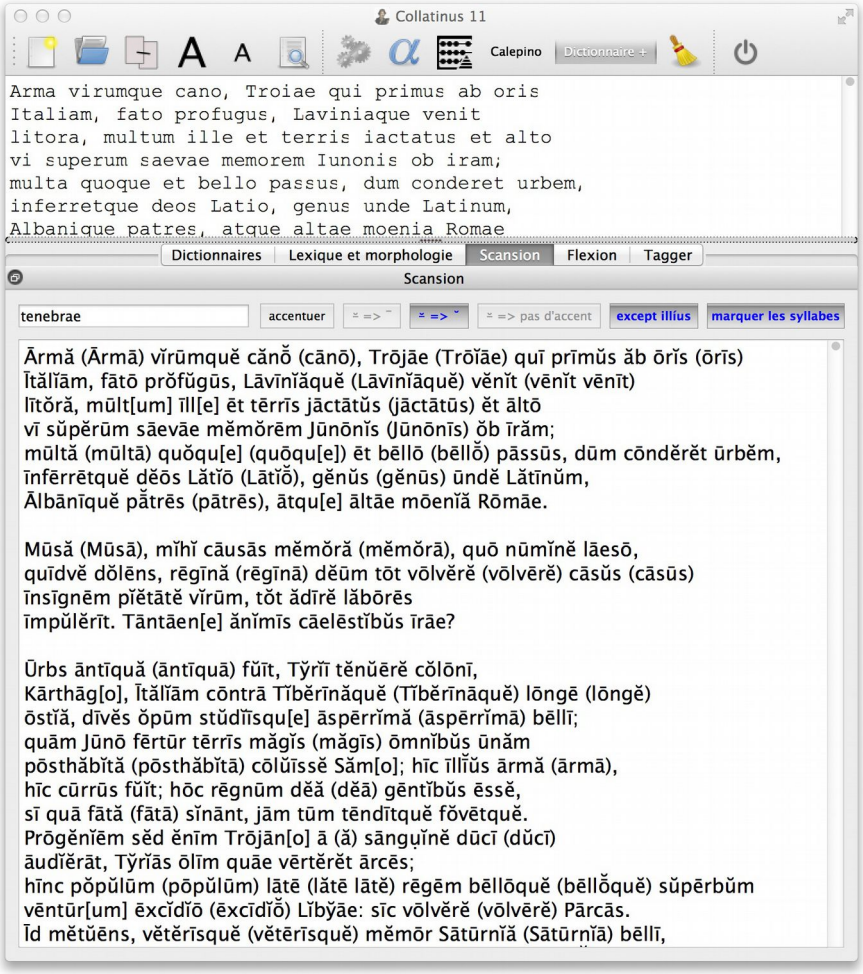

Figure 4: The beginning of Aeneid is scanned in Collatinus. In case of ambiguity, the most frequent solution is given first and the other possibilities are between parentheses. Note the lengthening in *Ītălĭ<u>ām</u> fātō* and the elisions, indicated with the vowels between square brackets, in *mūlt[um] īll[e] ēt* due to the words that follow. To scan properly the first paragraph, one has to correct the end of the second hexameter in *Lāvīnjăquĕ vēnĭt*. The first "wrong choice", *ōrĭs* instead of *ōrīs*, has no consequence, the length of the last syllable of the verse being *indifferens*.

2.2.2 Accented Latin

With time, the quantitative accent disappeared, replaced by a stress accent. This stress accent is widely used in the Middle Ages for rhythmic prose. The rhythm in the clausulae (and in the sentences) follows some rules with fixed scheme, which can in turn facilitate the memorization and/or the understanding of a text. The use of rhythmic prose depends on the place and the period of the writing. Reciprocally, it can be used to date and localize a text or even contribute to its attribution to an author. Thus, it is interesting to study quantitatively the disposition of the stress accents in the sentence or, at least, in the clausulae[43]. We did not implement the statistics about the rhythm in Collatinus for various reasons. But Collatinus can prepare the text so that it is easy to know the length of the word and if it is a paroxyton or a proparoxyton. On a simpler level, the stress accent allows to read Latin in a more musical way[44].

The position of the stress accent depends directly on the quantity of the last-but-one vowel. The stress is on the last-but-one syllable, except if it is a short vowel. In such a case, the accent falls on the last-but-two syllable, and the word becomes a proparoxyton. The enclitics make an exception to the rule of accentuation as they always attract the accent. For instance, the nominative *fīlĭăquĕ* is paroxyton, *filiáque*, even though the a is short. As soon as a program knows the quantities of the syllables, it is rather simple to implement the rules for the stress accent. When the last-but-one vowel is common as in *tĕnēbrāe* (where the second e is short in 20% of the 280 occurrences found by Pede Certo[45]), the user has to choose the behavior of Collatinus. When a form is ambiguous with two (or more) homographs, Collatinus gives the two solutions. For instance, it gives *áccido (accído)*, the first one coming from "*accĭdo*: to fall upon or down upon a thing", and the second from "*accīdo*: to begin to cut or to cut into". The user can then choose the proper one depending on the context.

43 Tore Jansson, *Prose Rhythm in Medieval Latin from the 9th to the 13th Century*, Stockholm, 1975 [Studia Latina Stockholmiensia 20] and Anne-Marie Turcan-Verkerk and Philippe Verkerk, "Un programme informatique pour l'étude de la prose rimée et rythmée" in Le médiéviste et l'ordinateur 33, printemps 1996, p. 41-48.

44 It is especially true in France where Latin is often read without accentuation, mainly because the French language has no accent (or a very weak one, always on the last syllable).

45 We thank Emanuela Colombi and Luigi Tessarolo who gave us the list of scanned forms found in their 244 000 Latin dactylic verses.

Syllabication or hyphenation can be also addressed by Collatinus: it considers that a syllable is based on a vowel with a marked quantity (the u of *lingua* or of *equi* is not taken into account). The rules for the association of the consonants to the syllable are the usual ones. The etymology sometimes plays a role in the syllabication and Collatinus has a file[46] with the list of the "etymological exceptions" (which can be easily edited, if needed). For instance, the perfect *abscidi* will give the two hyphened forms *abs·cí·di* and *áb·sci·di*, which derive respectively from *abs-cīdo* and *ab-scindo*.

2.3 Disambiguation of Latin

As in every language, forms in Latin can be ambiguous. This ambiguity can be at different levels. On one hand, in a declension, different cases can have the same form for the same word. The example everybody knows is the first declension with the word-endings for the nominative and ablative which look the same but are different. On the other hand, different lemmas can take accidentally the same form. For instance, *oris* can derive from *ora, ae* or from *os, oris*. It can be useful to apply the usual technics of disambiguation to propose the most probable analysis first. Obviously, one has also the perfect homographs, as the two *populus* or the two *levis*, that share the same inflected forms and are completely undistinguishable.

2.3.1 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 associate a tag to each form that reflect its nature or its function. The part-of-speech (POS) is often used as a tag, sometimes complemented with some other pieces of information. The method relies of 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

46 This file is the result of the work of Frère Romain Marie, monk at Abbaye Saint-Joseph de Clairval in Flavigny-sur-Ozerain.

47 See for instance "*A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*" by L.R. Rabiner, in Proceedings of the IEEE, 77, 257-285 (1989).

assumes that the most probable sequence is the good one, at least the best 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.

To start with, one has to choose the tag-set and to do some statistics on a training corpus. 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 got the texts lemmatized and analyzed by the LASLA[49]. They count slightly less than two million words, each form being associated with a lemma and a code that gives the full analysis[50]. This code cannot be used as a tag, because it would lead to a excessively large tag-set with more than 3,000 different tags. We cut in these codes some redundant information: for instance, for verbs, the group of the 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[51]. 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.

2.3.2 LASLA Tagger

With the statistical data extracted from the texts lemmatized by the LASLA, we have developed a lemmatizer-tagger for them. It is based on a form lemmatizer: for each word of the text, the

48 For a more detailed description of a tagger, see "*Probabilistic Part-of-Speech Tagging Using Decision Trees*" by H. Schmid, in International Conference on New Methods in Language Processing, Manchester, UK, 1994 (pp. 44-49). http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

49 We thank Dominique Longrée and Gérald Purnelle who gave us these texts, the list of which can be found at: http://web.philo.ulg.ac.be/lasla/textes-latins-traites.

50 The gender is absent in the corpus we have treated.

51 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 are not done for the moment.

program looks in the file of forms and lists all the known lemmatizations. If a word is not found in the file, the code sends a request to Collatinus which is supposed to run in the background on the same computer. Collatinus is able to open a server[52] on an internal port of the computer, and when it receives a form in a properly formulated request, it will answer with the possible lemmatizations of this form. An extra work is necessary to match, if possible, the lemmas used by the LASLA with those of Collatinus. If Collatinus is not able to answer (either because it is not running or because it does not recognize the form), then the program asks the philologist who is supposed to supervise the process and waits for an answer.

The results are sorted according to the frequency, and a first attempt for the lemmatization of the text is obtained by putting together the most frequent individual lemmatizations. This first attempt does not take into account the context and its error rate is expected to be about 20%[53]. Then, the tagger enters to play and the magic of probabilities operates. 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 good one in 96% of the cases. As a last step, the philologist can check all the lemmatizations and, if needed, correct them. We are expecting the feed-back from the users[54] to evaluate the real accuracy of the tagger on new texts.

It is interesting to note that, though 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 a lot of them are skipped[55]. 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” analysis would spread its consequence. An examination of the list of words shows that slightly less than 40% of the forms are associated to a unique analysis (thus a single tag). Thus the probability to find 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

52 This server can also be used by any other program or script to send a request (to lemmatize or scan) to Collatinus.

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

54 A student of Dominique Longrée, Margherita Fantoli, is presently working on Pliny with this program.

55 For details about the pruning method, see the article of H. Schmid quoted in note 48.

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. As a consequence, if the tagger gives the wrong tag to a word, this error will affect the few following words too, but not very far. 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.

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 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.

2.3.3 Tagger in Collatinus

Collatinus is not a form-lemmatizer, which means that it has no access to the exact number of occurrences for each form. The information we have transferred from the statistics on the lemmatized texts is restricted to the number of occurrences of the lemmas and of the sequences of tags. The tag-set[56] has been also simplified to match the POS known by Collatinus. For instance, the LASLA has five different categories of adverbs, while Collatinus knows only one adverb. As the frequency of the form is not known, we have to estimate it. To do so, we make a strong assumption that is that, in a given category, all the words have the same probability to be associated to the possible tags. In this way, the number of occurrences of an analyzed form is simply the number of occurrences of the lemma, which is stored in the lexicon, multiplied by the frequency of the associated tag in the category.

With this approximated value for the number of occurrences, it is possible to calculate the same probabilities as before. Instead of giving only the best solution, we propose the first two sequences of tags, in terms of probabilities. Here, the tagging process is just intended to help the reader: it remains a lemmatization of the sentence or of the text, with a more sophisticated procedure to choose the best solution[57]. Obviously, it is not error-free: in the example of Figure 5, the “wrong choise” of the lemmatizer for oris has been corrected, but not the one for venit.

56 We use 91 tags in Collatinus and 179 for the LASLA tagger.

57 In the lemmatization tab, we just choose the most frequent solution, without taking into account the context.

A few improvements can be thought of. One of them would be to keep track of the first two solutions, not only for the complete sentence but for each sequences of tags between two fixed points. If the estimation made in the last section is true (one fixed point every 6-7 words), it would mean that long sentences split in two or more independent parts. Another extension of this work could be to use the tagger also to scan a text. However, one should be convinced that disambiguation through a probabilistic tagger works for Latin. The efficiency of the tagging process is demonstrated for modern languages but not for Latin, to our knowledge.

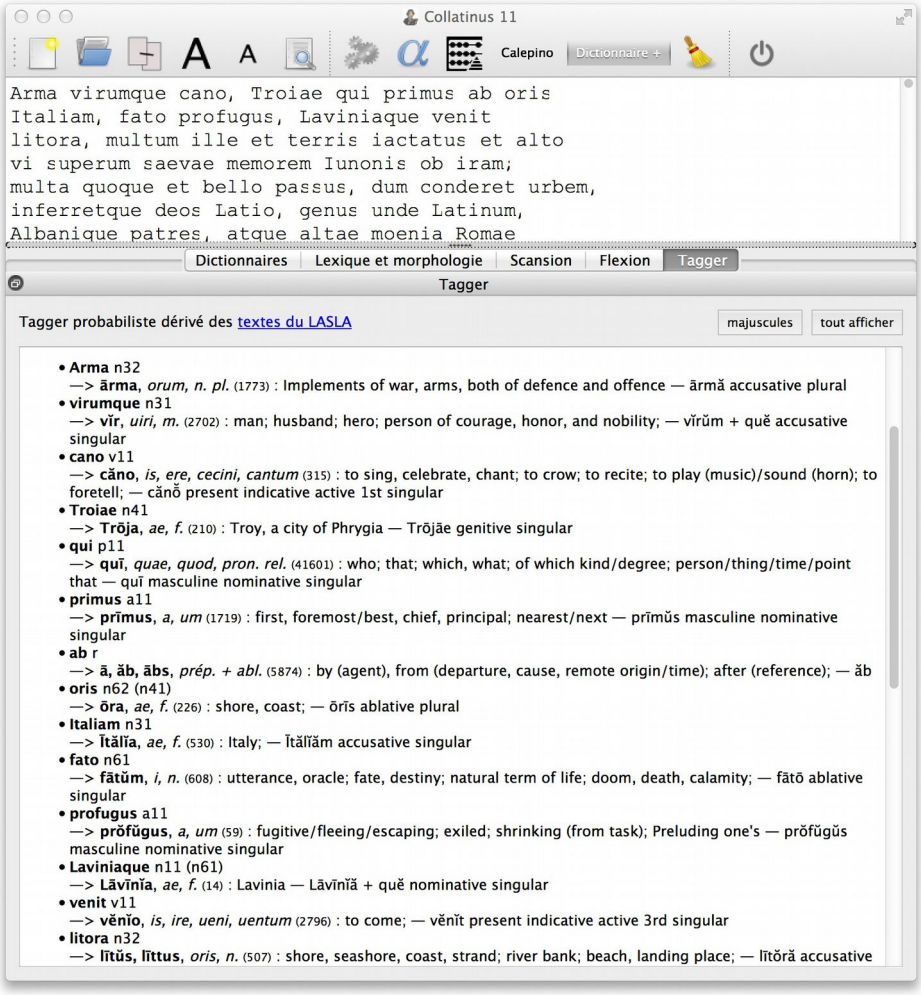

Figure 5: The beginning of Aeneid can be lemmatized taking into account the context through a probabilistic tagger. It corrects the wrong analysis given by the lemmatizer for *oris*, but not those of *laviniaque* and *venit*.

3. Further Developments

Collatinus and Eulexis are open-source programs. Although we have not published an API for them (we are not computer scientists), new features can be added to them or the code of their core can be included in a different environment for a specific task. We present hereafter three projects that we have in mind, at different stages of realization or of reflection.

3.1 Praelector

In 2002, Yves Ouvrard has directed the French translation of *The Art of Reading Latin: How to Teach it*[58] by William Gardner Hale. After that, he has been thinking of a program that would follow these principles. Praelector, built upon Collatinus'core, tries to simplify their application.

Praelector follows the principles of the classical syntactic analysis: a syntactic link connects a governor word to a dependent word. Governor and dependents form a syntactic group. A dependent can be itself the governor of a subgroup. So the language has a recursive structure. A word is dependent of only one other word[59]. Obviously, this rule has an exception: the relative pronoun is dependent of both his antecedent and of another word of the relative clause. The latin language generally applies the principle of projectivity: between the first (leftmost) and the last (rightmost) word of a group, all the words and groups must depend on the governor of this group. But this principle cannot be always applied because of the frequent exceptions, which ancient grammarians call *transgressio*, *disiunctio*, *traiectio*, and which we call *hyperbate*. It is an essential component of the classical rhetoric, and it is difficult to state its rules.

For the interface, the screen is divided into three parts. Above the read-only latin text is displayed. In the middle part, appears the list of syntactic links. And below, the translation is built gradually. The user selects, in the text, the governor and the dependent words, and the program proposes the links that could exist between these two words, together with an approximative translation of each link, see Figure 6. To do that, Praelector uses a set of rules recorded in a file. This file, which can be read and edited by any human being, gives for each

58 http://www.weblettres.net/languesanc/index.php?page=hale

59 One often uses the analogy between syntactic and genealogical trees and then the words at each end of the syntactic link become the father and the son. The mentioned rule can thus be formulated as “a word has only one father”.

link the morphologies of the governor and of the dependent, their relative position, some other constraints, and a translation toward the target language[60]. Then, the user chooses one link, edits its translation, and the translation of the sentence is automatically updated.

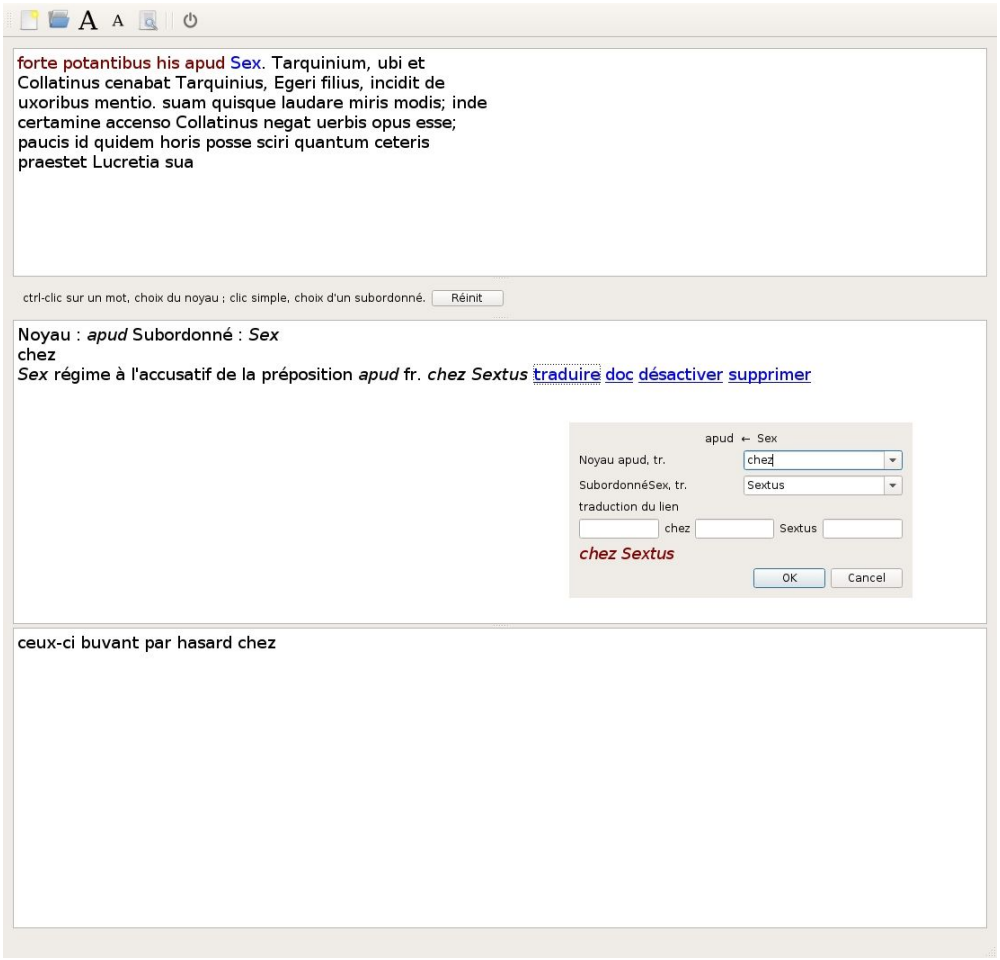

Figure 6: A screen-shot of Praelector. The topmost frame contains the text under study, in which the user selects two words. The program proposes all the possible syntactic links between these two words in the central frame. The user can then choose one of the links and edit its translation. The bottom frame gives the translation of the text which is gradually up-dated. The words that have been already treated are red-colored in the text-frame.

60 Presently, only in French.

Some issues remain. For instance, the coordination is difficult to handle, and the juxtaposition is even worse. The principle of projectivity has frequent exceptions. Ellipses are very frequent, and widely broader and more varied than those of our modern languages. Often many elements unrelated to the structure of the sentence are inserted in it: parentheses, incises, vocatives, etc. As it is, the project is not yet publishable, but its interface is usable and stable enough. The sources can be found on Github[61]

3.2 Medieval Latin

Medieval Latin is not always considered as a language on its own. In France, it is even disregarded by the latinists, although the production of texts during the Middle-Ages is much larger than the corpus of classical texts that reach us. On the other hand, medieval texts are often primary sources for historians, who do not necessarily master the Latin language. Thus it may be interesting to develop a tool able to handle the medieval texts.

Collatinus has fully open and editable lexica, associated with files for word-endings and irregular forms. Thus one can modify these files in order to fit the medieval Latin. This may be important if one wants to render the shift in the meaning of a word. For instance, the word *miles* shifts from the soldier to the knight. However, if the translation is not the principal goal and lemmatization is sufficient, a lighter method could be implemented. It consists in a patch or a filter added to the classical Latin to extend the understanding of medieval Latin by Collatinus. Obviously, new words as those coming from other languages have to be added to the lexicon, but the possibility to continue to use the 77,000 lemmas already known is a huge reduction in the needed investment.

One of the striking characteristics of medieval Latin is the “simplification” of writing, associated with a large variability. For instance, it is well known that the diphthongs *ae* and *oe* became simply e. The group *ti+vowel* becomes very often ci+vowel, so that one finds *leticia* for *laetitia*. Together with a few other transformation rules (interchangeability of i and y or of f and ph, etc...), these changes in writing would allow to recognize a large fraction of the unknown medieval forms.

61 https://github.com/ycollatin/Praelector

The idea is thus to define a set of transformation rules (in an editable file) that are applied both on the data (root-words and word-endings) and on the studied text. Then the usual lemmatization process goes on matching the “simplified” words with the combination of the “simplified” root-word and word-ending. Together with the analysis, one can get also the associated classical form which can be compared to the form found in the text. A distance between these forms can be evaluated and used to sort the solutions. It is not clear that this kind of ranking is convenient. For instance, the form *vite* is found often in medieval texts. Its exact (classical) lemmatization gives the ablative of *vitis*, while in most cases it stands for *vitae*. Obviously, the process will give false lemmatizations. One can take the same example: in a hypothetical text about oenology, vine and grapes, the word *vite* has no reason to be lemmatized as a form of *vita*. There is always a trade-off, but the human reader is well-qualified to choose the proper solution. It should be possible to apply some pre- or post-analysis of the text to determine if the transformation is plausible. For instance, if the text contains some ae, then the equivalence *vite* = *vitae* should be ruled out.

3.3 Disambiguation through Syntactic Analysis

As soon as the rules for syntactic links are known, a computer can use them to build all the possible syntactic trees for a sentence. Even for short and simple sentences, the number of solutions can be very large. However, aesthetic criteria allow to choose a few trees in the forest and the first trials we have made show that the correct solutions is among the first five. Conversely, knowing the best syntactic trees drastically reduces the possible lemmatization and analyses of the words.

For the moment, the method is rather crude and fails for complex sentences. The idea is to analyze all the words and then to establish all the possible syntactic links between each pair of words. Finally, the program chooses one link ending on each word (except for the relative pronoun as pointed out earlier), leaving an orphan, usually the predicate. Obviously, each time it chooses one link, the program adds constraints on the analyses of the two connected words, which means that it has fewer possible choices for the next move. If the number of orphans goes above two, the exploration is aborted and the program tries another combination of links. Then starts the evaluation process: the program gives a score at each tree, more precisely it gives

penalties according to different criteria. The total length of the branches of the tree is one of the criteria. It represents the compactness of the tree in which distant words are seldom directly linked. The number of orphans is the strongest criterium: one prefers the solution where only the predicate has no governor-word[62]. The projectivity gives also a criterium: the links should not cross each other (except, once again, for the relative pronoun). So we add a penalty for each crossing between two links. We have still to play with the weights given to these criteria in order to sort the trees.

One has probably to find out a better algorithm to build fewer useless trees. It is a necessary condition to become able to treat complex sentences. The aesthetic criteria used to select the nicer trees have no strong scientific justification, but they give interesting results. As the probabilistic tagger, this method chooses a solution among the possible lemmatisation of the sentence. It is not necessarily the correct one. It could be interesting to compare or to couple both approaches, maybe to improve the result.

.oOo.

Collatinus and Eulexis are free and open-source programs that allow the lemmatization of Latin and ancient Greek, respectively. They also give the possibility to consult dictionaries. As every open-source program, they can be re-used or adapted for another function. Hoping that they will meet your requirements, feel free to improve them.

62 Usually the predicate is placed on the top of the syntactic tree, which puts the tree upside-down, with its leaves at the bottom.