

Petit manuel de lemmatisation

avec Collatinus

Philippe Verkerk
Juin 2019

Introduction

Je ne reviendrai pas sur l'utilité, voire la nécessité, d'une lemmatisation des textes. Je dirais juste qu'il y a un problème de vocabulaire. Le mot « lemmatisation » recouvre lui-même, à mon avis, deux notions voisines. Pour l'utilisateur final, lemmatiser signifie associer à une forme fléchie un (et un seul) lemme. Or en Latin, seulement 40% des formes attestées ne peuvent avoir qu'une seule analyse¹. La fraction des formes n'ayant qu'une seule lemmatisation possible sera nettement plus grande, mais de nombreuses formes peuvent venir de différents lemmes. Il manque un mot pour désigner l'opération qui consiste à donner tous les lemmes pouvant conduire à une forme². Techniquement, il s'agit d'une lemmatisation, dans un sens plus large que celui voulu par l'utilisateur final. Une deuxième étape, de désambiguïsation, est nécessaire pour choisir un lemme particulier parmi la collection des possibles. On pourrait donc parler de lemmatisation/désambi-guïsation pour désigner le processus global demandé par les utilisateurs.

Collatinus est un lemmatiseur qui dispose également d'un tagueur³ probabiliste pour choisir en tenant compte du contexte ce qu'il pense être la meilleure solution. Je n'ai pas évalué ses performances, mais j'ai brièvement mesuré celles de son petit frère, le LASLA-tagger. Pour l'analyse complète, j'ai eu des taux d'erreur entre 8 et 18%. Pour le seul lemme, les taux obtenus sont deux fois plus faibles. Si on veut atteindre l'exactitude, il faudra utiliser tous les ressorts de l'intelligence humaine et faire corriger les résultats de l'ordinateur par un latiniste chevronné. Néanmoins, selon les usages que l'on peut avoir, ces taux d'erreurs peuvent ne pas être bloquant.

-
- 1 L'analyse est ici le lemme ainsi que les traits morphosyntaxiques pertinents. Ce résultat de 40% s'entend en nombre d'occurrences et il est tiré de l'analyse des textes lemmatisés au LASLA.
 - 2 Je considère ici les lemmatiseurs classiques basés sur un lexique. Personne ne sait bien comment fonctionnent les réseaux de neurones qui donnent aussi des résultats intéressants.
 - 3 J'ai choisi de franciser par *tagueur* le mot anglais « tagger » (étiqueteur) employé habituellement dans le domaine des TAL. Cela permet également de distinguer l'outil « tagueur » de l'action « taguer », ce que ne permettait pas l'anglicisme (puisque les nouveaux verbes sont toujours du premier groupe). « Étiqueteur » aurait été plus conforme au sens du mot anglais, mais plus éloigné (et moins rigolo).

Installation

Collatinus est disponible sur le site de [Biblissima](https://outils.biblissima.fr/fr/collatinus/)⁴. A priori, un lien direct pour télécharger l'installateur pour votre système (Windows ou Mac OSX) devrait apparaître. Pour les utilisateurs de Linux, la situation est plus compliquée, en particulier à cause de la variété des distributions.

Plusieurs versions sont disponibles qui se distinguent par le nombre de dictionnaires installés. Si vous ne disposez que d'une vieille machine au disque saturé, la version minimale sera adaptée. Toutefois, je recommande plutôt la version complète qui contient tous les dictionnaires disponibles à ce jour. Même si aujourd'hui, vous n'avez pas besoin du Valbuena (dico Latin-Espagnol du XIXe s.), qui sait si demain il ne vous servira pas. Évidemment, il vaut mieux disposer d'une bonne connexion internet pour télécharger la version complète (~900 Mo). À partir d'une installation minimale ou intermédiaire, il est toujours possible de télécharger et d'installer des dictionnaires supplémentaires⁵.

Une fois que le fichier d'installation est téléchargé (un fichier .exe sous Windows, .dmg sous Mac OS, la plupart du temps dans le dossier « Téléchargements »), il suffit de l'ouvrir. Sous Windows, on est guidé pas à pas et, à la fin, on dispose d'un raccourci sur le bureau pour lancer Collatinus (on peut aussi aller le chercher dans « Program files »). Sous Mac OS, s'ouvre une fenêtre avec l'application Collatinus qu'il suffit de faire glisser dans le dossier « Applications ». On peut alors fermer la fenêtre issue du .dmg et aller chercher Collatinus dans le dossier « Applications » (si on compte l'utiliser souvent, on peut l'installer dans la barre de lancement).

Pour Linux, la version 10.2 de Collatinus est disponible pour l'ensemble des grandes distributions du système. La version 11 est disponible pour Debian avec l'étiquette « unstable ». Il est difficile de pronostiquer un calendrier et de prédire quand les fonctionnalités médiévales de C11.2 seront distribuées. La meilleure solution est probablement d'installer le compilateur Qt⁶ sur la machine que l'on souhaite utiliser, de récupérer les fichiers sources de Collatinus sur GitHub⁷ et de compiler soi-même Collatinus.

4 <https://outils.biblissima.fr/fr/collatinus/>

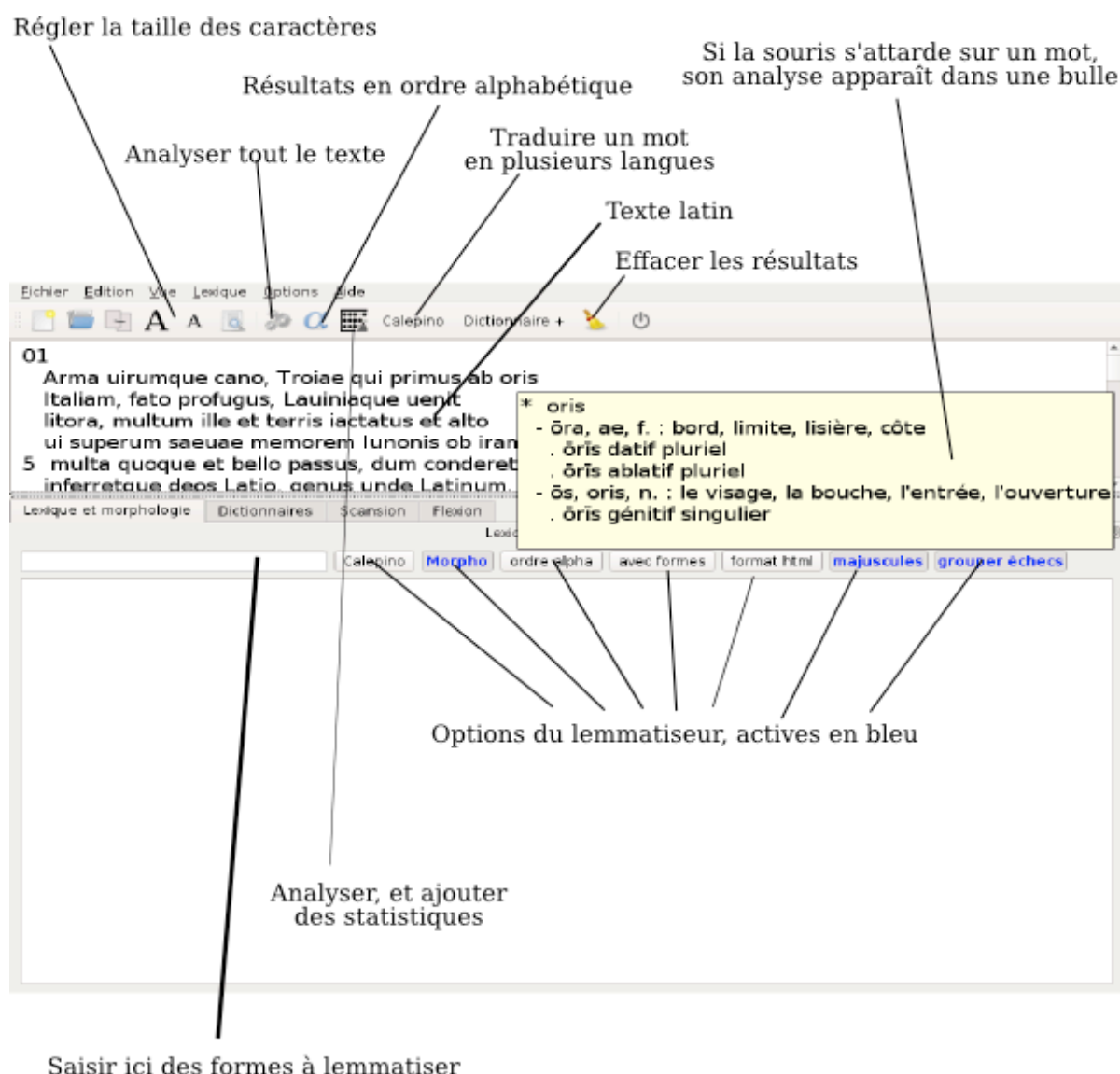
5 Les dictionnaires supplémentaires sont disponibles sur le site de Biblissima. Ils sont dans un format compressé, mais Collatinus sait les gérer et placera tous les fichiers nécessaires là où il faut (voir la FAQ pour plus de détails).

6 Qt est un environnement de développement avec un compilateur et un ensemble de bibliothèques qui permet de porter sur plusieurs plateformes un même programme.
<https://www.qt.io/download>

7 GitHub est un dépôt qui sert de gestionnaire de version. Pour la toute dernière version, on prendra (aujourd'hui) la branche « Medieval » : <https://github.com/biblissima/collatinus/tree/Medieval>

Premiers pas

Collatinus est doté d'une interface graphique relativement simple et conviviale. Sa fenêtre est séparée en deux parties, agrémentées de boutons qui simplifient la tâche de l'utilisateur. Le cadre supérieur va contenir le texte que l'on souhaite traiter. Celui-ci peut être saisi directement dans le cadre ou y être introduit avec un copier-coller (en copiant le contenu de la fenêtre d'un navigateur, par exemple). Une dernière méthode, peut-être la plus utile, consiste à ouvrir un fichier-texte. Pour ce faire, on pourra utiliser le bouton représentant un dossier dont on sort le document, le menu Fichier/Ouvrir ou l'habituel raccourci-clavier Ctrl-O⁸. Si nécessaire, on peut nettoyer le texte introduit de ses décorations (accents, trémas etc...) en utilisant le menu Fichier/Ôter les accents.



Le cadre inférieur de la fenêtre de Collatinus est fait pour afficher les résultats. On dispose d'onglets pour choisir quel type d'informations on souhaite obtenir. Les deux onglets que nous allons utiliser sont « Lexique et morphologie » et

⁸ Sur Mac, on utilisera, comme toujours, la touche Cmd là où Windows utilise Ctrl.

« Tagueur ». Le premier pour débroussailler le terrain et le second pour avoir le lemme le plus probable associé à la forme en tenant compte du contexte.

TextiColor et correction des coquilles

Quand on récupère des textes issus d'une océrisation plus ou moins bien faite, il peut y trainner un certain nombre de coquilles. La relecture d'un texte étant toujours longue et fastidieuse, Collatinus peut vous aider en vous indiquant les mots qu'il ne connaît pas. Évidemment, il peut arriver qu'une coquille conduise à un mot qui existe (par exemple, un « eodem » là où il y avait un « eadem » ou inversement), auquel cas Collatinus ne s'en apercevra pas. Pour mettre des couleurs sur le texte (dans la fenêtre supérieure), il faut utiliser le menu Fichier/Lire une liste de mots connus et lemmatiser le texte (l'onglet Lexique et morphologie doit être sélectionné et on déclenche la lemmatisation avec le bouton représentant des rouages, par le menu Lexique/Lancer ou Ctrl-L). Le texte étant maintenant colorisé, on peut le copier (Ctrl-A puis Ctrl-C) et le coller dans un traitement de texte (Ctrl-V). Je ne garantis pas que tous les TdT conservent les couleurs, mais c'est le cas pour LibreOffice (et probablement aussi pour ms-Word). Le relecteur devra faire attention aux mots en rouge qui sont ceux que Collatinus n'a pas reconnus. Bien évidemment, le fait que le mot ne soit pas connu ne signifie aucunement qu'il contient une coquille. Quand la relecture est terminée, on peut enregistrer la nouvelle version corrigée en « texte seulement », la recharger dans Collatinus et les couleurs auront disparu.

Le TextiColor a été développé initialement pour l'enseignement du Latin. Une liste de mots est donnée à Collatinus qui va mettre en vert les formes dérivées de ces mots connus, en noir les mots que lui connaît, mais qui ne figurent pas dans la liste, et en rouge les mots qu'il ne connaît pas⁹. L'élève étant censé connaître un certain vocabulaire, les mots connus sont mis en vert dans le texte pour l'encourager à les reconnaître. De même, leur analyse et leur traduction ne seront pas données dans les résultats de la lemmatisation. Il faudra donc bien vérifier que l'on a désactivé l'option de mots connus avant de faire la lemmatisation définitive d'un texte que l'on voudra sauvegarder au format csv.

La relecture du texte, et plus particulièrement celle des mots en rouge, conduit à trois catégories de mots non reconnus :

- les coquilles que l'on corrigera
- les variantes graphiques de mots connus
- les mots nouveaux.

Ces deux dernières catégories peuvent être traitées comme une seule selon les besoins. Collatinus-médiéval connaît un minimum de transformations graphiques habituelles. Si dans un contexte particulier il apparaît d'autres variations graphiques fréquentes (par exemple, une confusion du v et du b), il peut être utile d'ajouter cette transformation à la liste des transformations effectuées. Il faut toutefois être prudent car un nombre trop important de transformations risque d'introduire des fausses lemmatisations qu'il faudra ensuite corriger. Si une variante graphique n'apparaît que

9 L'auteur de la liste peut choisir d'autres couleurs que les vert-noir-rouge définis par défaut.

quelques fois ou dans seulement quelques mots, il peut être plus simple d'introduire ces variantes comme des nouveaux mots.

La gestion des variations graphiques par Collatinus est assez rudimentaire¹⁰. Le programme effectue un ensemble de transformations à la fois sur les formes du texte et sur les données classiques. Il s'agit en quelque sorte d'une simplification/normalisation transitoire des graphies. Que le texte contienne « delfini » ou « delphini », le programme cherchera « delfini » (puisque j'ai défini une transformation ph -> f) qu'il lemmatisera comme une forme de « delphinus » (puisque'il redonne la forme « classique » du lemme). J'ai renoncé à mettre un filtre a posteriori pour éliminer les solutions approchées s'il existe une solution exacte, car cela m'a semblé introduire plus d'erreurs que ça n'en corrigeait. Par exemple, la forme « sancte », courante dans les hagiographies, a beaucoup plus de chances d'être la variante médiévale de « sanctae » plutôt que le vocatif masculin singulier « sancte ».

La liste des transformations faites par Collatinus est donnée dans le fichier éditable data/medieval.txt. Si on juge que la confusion b/v est pertinente, on ajoutera une ligne contenant « b;v » dans ce fichier. Éventuellement, on mettra un commentaire (une ligne commençant par un point d'exclamation) pour préciser la portée de cette transformation.

Complétion du lexique

Cette étape est optionnelle. Elle vise à compléter le lexique pour que, lors de la prochaine étape, le programme reconnaisse toutes les formes rencontrées. Dans une version future de Collatinus, nous envisageons d'intégrer un outil permettant de définir des configurations particulières pour s'adapter à un corpus donné. Cet outil gèrera aussi bien les variations graphiques mentionnées ci-dessus que les nouveaux mots.

Une fois que les coquilles ont été corrigées, on va lemmatiser le texte à nouveau, en activant le groupement des échecs et l'ordre alphabétique (les boutons « ordre alpha » et « grouper échecs » apparaîtront en bleu dans la barre d'outils de l'onglet « Lexique et morphologie »). On obtient alors la liste des formes non-reconnues à la fin des résultats de lemmatisation, après un séparateur du genre « --- 375/20132 (2 %) FORMES NON RECONNUES --- ». L'intérêt de l'ordre alphabétique est de rapprocher les formes issues d'un même lemme (pas toujours, mais souvent). Un latiniste voit tout de suite quels mots manquent au programme en opérant les groupements nécessaires¹¹. Il a alors la possibilité de les ajouter à la fin du fichier « data/lem_ext.la » et d'en donner une traduction française dans « data/lem_ext.fr » (éventuellement aussi une traduction anglaise dans

10 Dans la version 11. Une version 12 est en développement pour affiner ce traitement.

11 Nous envisageons de développer un « devineur de lemmes » qui essaierait de faire ces regroupements en cherchant les désinences qui s'ajouteraient à un radical commun. Il préparerait ainsi automatiquement les compléments à ajouter aux fichiers lem_ext.la et lem_ext.fr (tout en laissant les traductions en blanc). Le latiniste sera alors invité à relire et corriger ces propositions avant de les ajouter aux bons fichiers.

data/lem_ext.en). Pour les traductions, le format du fichier est très simple : chaque ligne contient deux champs séparés par un deux-points « : ». Le premier champ donne le lemme, sans quantité mais éventuellement avec un numéro d'homonymie, et le second donne la traduction (libre). Le fichier lem_ext.la donne toutes les informations morphologiques nécessaires au programme. Une ligne est composée de six champs séparés par une barre verticale « | ». Les deux premiers champs doivent nécessairement être renseignés : ils donnent le lemme (avec quantités et numéro d'homonymie, si nécessaire) et le paradigme. Collatinus connaît plus de 100 paradigmes, mais les plus courants sont uita, lupus, templum, ciuis et miles pour les substantifs, doctus, fortis et diues pour les adjectifs et amo, moneo, lego, audio et capio pour les verbes. Pour les noms de la 3^e déclinaison comme pour les adjectifs de la 2^e classe, il faut donner le radical du génitif dans le 3^e champ. Pour les verbes, il faut donner les radicaux du parfait et du supin dans les 3^e et 4^e champs. Pour les verbes du 1^{er} groupe, il n'est pas nécessaire de donner les radicaux s'ils se construisent comme ceux d'amo lui-même (c'est à dire *āmāv* et *āmāt*). En revanche, si les parfait et supin ne sont pas en -av- et -at- (comme pour « **sōnō**, *sōnūī*, *sōnītum*, *āre* ») il faudra donner ces deux radicaux, *sōnū* et *sōnīt* pour cet exemple. Dans le 5^e champ, on donnera les indications morphologiques que l'on jugera utiles : celles que l'on trouve en général dans un dictionnaire. Pour les noms, on prendra soin de donner le genre (ça pourra être utile le jour où on essaiera de construire des arbres syntaxiques pour vérifier un accord en genre entre le nom et son adjectif). Le 6^e champ contient le nombre d'occurrences mesuré pour ce lemme dans les textes du LASLA. Pour les mots que Collatinus ne connaît pas, on mettra la valeur magique 1. Si on attend une ambiguïté pour certaines formes issues de deux (ou plus) lemmes différents, on peut privilégier celui qui est le plus fréquent en mettant un entier supérieur à 1.

Si on a complété le lexique, il faut pour l'instant quitter et relancer Collatinus pour que le lexique soit rechargé. La prochaine version de Collatinus fera ça beaucoup plus astucieusement avec des « modules », associant variantes graphiques et lexique complémentaire, qui pourront être chargés et déchargés à volonté.

Lemmatisation avec le tagueur probabiliste

Cette étape vise à produire un fichier CSV avec la meilleure solution (lemme et analyse) pour chaque mot en tenant compte du contexte. On va donc se placer dans l'onglet « tagueur ». Il y a deux stratégies possibles qui seront choisies en fonction du degré d'optimisme du moment. La nécessité d'exactitude est aussi à évaluer et il faudra en tenir compte. Si on veut un résultat exact, il faudra de toute façon relire attentivement le fichier obtenu¹².

Le tagueur probabiliste propose une option qui consiste à « tout afficher » ou pas. Lorsque cette touche est désactivée, le programme ne propose que la solution

12 Pour les philologues acharnés, il y a une alternative possible avec le « LASLA-tagger ». Ce programme utilise le moteur de Collatinus, ainsi que les données du LASLA, pour produire un fichier au format du LASLA que l'on peut ensuite introduire dans Hyperbase (outil de textométrie). Il couple un tagueur probabiliste et un éditeur de la réponse. Pour l'instant, il ne gère pas les graphies médiévales (mais ça pourrait se faire).

qu'il a identifiée comme étant la plus probable. On n'a alors pour chaque mot qu'un seul lemme associé à une seule analyse. C'est a priori ce que souhaite l'utilisateur final d'un texte lemmatisé. Toutefois, le programme est loin d'être infallible et s'il se trompe, le relecteur devra « inventer » le bon lemme et la bonne analyse. Avec l'option « tout afficher », le programme donne d'abord la solution la plus probable, comme précédemment, mais il donne aussi toutes les autres solutions qu'il a trouvées (comme dans le cas de la lemmatisation). Le travail du relecteur est alors simplifié dans le cas où le programme s'est trompé : il peut choisir la bonne solution parmi toutes celles qui sont proposées et la mettre en première ligne. L'inconvénient est qu'à la fin il ne faut garder que la première solution. Ça doit pouvoir se faire avec un petit programme que je n'ai pas encore écrit. L'autre solution, manuelle, consiste à supprimer au fur et à mesure de la relecture les lignes inutiles.

Quel que soit le contenu de la fenêtre du tagueur, l'export au format CSV traite l'ensemble du texte. Chaque colonne contient une information particulière :

1. Numéro du mot dans le texte
2. Numéro de la phrase
3. Numéro du mot dans la phrase
4. Forme du texte
5. Tag
6. Lemme associé, sans quantité (pour faciliter un tri en ordre alphabétique)
7. Lemme associé, avec ses quantités et ses informations morphologiques
8. Nombre d'occurrences du lemme dans les textes du LASLA
9. Traduction
10. Forme avec quantités et son analyse morphosyntaxique ; si le mot porte l'enclitique, on le retrouve ici¹³ (par exemple, "Pōmpējūs + quē").

Dans le cas particulier où on a demandé toutes les analyses possibles, les mots sont répétés autant de fois que nécessaire avec les mêmes numéros de référence¹⁴. La première ligne, pour chaque mot, est celle que le tagueur préfère. Dans la dernière colonne des lignes suivantes (pour chaque mot), s'ajoute une information supplémentaire qui est un "élément de probabilité" qui est associé au tag de la solution proposée. Il ne s'agit pas d'une probabilité puisque la somme n'est pas normalisée à 1. La solution préférée est donnée une deuxième fois : elle est celle qui a la plus forte de ces *probabilités*. Dans le cas où un même tag (même analyse) est

13 Souvent les « tokeniseurs » séparent l'enclitique de la forme et introduisent ainsi un mot supplémentaire. Certains vont même jusqu'à placer l'enclitique -que devant le mot qui le porte, prenant la place d'un « et » qu'il remplace.

14 À l'heure où j'ai écrit ces lignes, le programme répétait les cinq premières colonnes autant de fois que nécessaire. C'était une erreur que je pense avoir corrigée, car pour les solutions supplémentaires le tag à indiquer serait plutôt celui de l'analyse portée par la ligne (plutôt que le tag préféré).

partagé par deux lemmes différents, c'est le plus fréquent des deux lemmes qui est préféré.

Ces *probabilités* peuvent donner des indications précieuses. Si les deux plus grands nombres donnés sont voisins, cela signifie que le tagueur a dû choisir la solution la plus probable, mais qu'il aurait pu hésiter. Si les statistiques avaient été faites sur plus (ou moins) de textes ou des textes différents, les *probabilités* trouvées auraient pu être légèrement différentes et l'ordre des meilleures solutions inversées. En bref, le tagueur a des chances de se tromper dans de tels cas. Nous laisserons l'utilisateur se faire une opinion sur le sens à donner à *nombres voisins* en fonction des erreurs d'analyses qu'il pourra relever. À noter aussi que le tagueur choisit le tag en tenant compte du contexte : s'il se trompe pour un tag, il y a de fortes chances que cette erreur ait des répercussions un peu plus loin. Donc qu'il y ait des erreurs en chaîne, mais en moyenne pas trop loin (environ 6 ou 7 mots). Chaque phrase étant traitée indépendamment des autres, une erreur ne se propage pas dans la phrase qui suit.

Conclusion

Je n'ai pas parlé ici des autres possibilités de Collatinus car elles sortent du cadre de la lemmatisation. Toutefois, elles peuvent aussi intéresser les Médiévistes : à côté de la consultation des dictionnaires (dont Du Cange), l'outil de scansion est doublé d'un outil d'accentuation, plus adapté au Latin médiéval où l'accent est devenu tonique. Cela peut permettre d'étudier le rythme des clausules, par exemple. Pour les débutants, il y a aussi les tableaux de flexion qui permettent de réviser déclinaisons et conjugaisons.

Parmi les extensions possibles de Collatinus, je mentionnerai une voie à explorer qui mènerait à la construction (semi)automatique des arbres syntaxiques. En s'appuyant sur le résultat du tagueur probabiliste, il faudrait établir une liste (ordonnée) des liens syntaxiques possibles entre les paires de mots et ensuite choisir un ensemble cohérent de liens pour former l'arbre le plus élégant. Idéalement, l'utilisateur devrait pouvoir intervenir en validant ou en interdisant certains liens pour guider l'ordinateur vers une solution satisfaisante.

Une autre voie, déjà évoquée, est d'aborder de façon plus systématique les variantes graphiques et le vocabulaire spécifique à certains corpus. Qu'ils soient classés par époque, par genre ou par auteur, les textes ont des spécificités qu'un traitement « généraliste » ne saurait cerner. La version 12 de Collatinus sera modulaire et les différents modules pourront gérer les variantes graphiques, qu'elles soient médiévales ou archaïques, ainsi que les changements du sens des mots au cours de la longue histoire du Latin. Elle sera accompagnée d'un éditeur dédié à l'élaboration de ces modules qui pourront ensuite être publiés.

Amusez-vous bien.