Bulletin du centre d'études médiévales d'Auxerre | BUCEMA
Collection CBMA | 2012

Textes anciens en série

Outils informatiques d'organisation et de manipulation de bases de données textuelles

Alain Guerreau

Édition électronique
URL : https://journals.openedition.org/cem/12177
DOI : 10.4000/cem.12177
ISSN : 1954-3093

Éditeur
Centre d'études médiévales Saint-Germain d'Auxerre

Référence électronique
Alain Guerreau, « Textes anciens en série », *Bulletin du centre d'études médiévales d'Auxerre | BUCEMA* [En ligne], Collection CBMA, mis en ligne le 01 mars 2012, consulté le 24 septembre 2022. URL : http://journals.openedition.org/cem/12177 ; DOI : https://doi.org/10.4000/cem.12177

Ce document a été généré automatiquement le 24 septembre 2022.

Creative Commons - Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions 4.0 International - CC BY-NC-SA 4.0
https://creativecommons.org/licenses/by-nc-sa/4.0/

Textes anciens en série

Outils informatiques d'organisation et de manipulation de bases de données textuelles

Alain Guerreau

Verantwortung für die Gemeinschaft und
Engagement für das Gemeinwohl
Hatice Hakyün *Der Tagesspiegel* 3.2.2012

1 On part de la situation suivante : un historien (étudiant ou chercheur débutant une recherche fondée sur l'analyse d'un corpus de textes, historien désireux de « mettre en ligne » un corpus de textes anciens...) disposant - ou pas - d'une série de fichiers numériques où sont stockés les textes en question, devant la nécessité de les « gérer » pour pouvoir les manipuler utilement. Cette nécessité implique eo ipso l'usage d'un ou plusieurs logiciels appropriés. C'est ce sujet qui sera abordé ci-dessous[1]. Avant d'entrer directement dans la matière, il faut énoncer trois préalables fondamentaux :

2 1. il n'existe pas, et ne peut pas exister, de solution passe-partout ; les situations et les problèmes à résoudre sont particulièrement variés, il faut d'abord bien analyser la situation pour déterminer quel est ou quels sont les outils les mieux adaptés ; la solution passe-partout, à supposer qu'elle existe, donnera au mieux des résultats indigents, voire souvent faux. Il faut donc bien comprendre le sens et la structure des textes d'un côté, et les capacités exactes des divers outils disponibles, de l'autre.

3 2. les textes anciens ont des propriétés bien spécifiques, qui les différencient fortement des grands corpus que l'on rencontre sur internet, ou que l'on peut en extraire ; et au surplus les manipulations auxquelles on souhaite se livrer n'ont rien à voir avec celles qui concernent les corpus actuels. Du coup, même les outils standards nécessitent des réglages ou de vraies modifications pour être pleinement utiles ; comme on va le voir, il faut ajouter aux textes eux-mêmes des ensembles d'informations supplémentaires qui devront être codées ; pour ces diverses raisons, on ne peut pas s'en tirer sans « mettre les mains dans le cambouis », autrement dit, travailler avec les fichiers de paramètres et, le cas échéant, avec le code informatique lui-même. S'imaginer que l'on puisse se contenter du « clic-clic » routinier est une erreur fatale, dont les conséquences sont

particulièrement négatives ; le laisser croire, c'est orienter les étudiants ou les collègues dans des impasses fangeuses, dont ils ne se tireront pas sans douleur.

4 3. la situation définie au départ est une situation de recherche : la finalité n'est ni administrative ni commerciale. Au surplus, l'étudiant ou le jeune chercheur ne disposent pas de moyens financiers tels qu'ils puissent, ici ou là, pour voir, se procurer contre deniers des logiciels dont ils se serviront ponctuellement ; et les institutions de recherche et universitaires sont soumises à des restrictions dramatiques qui leur imposent des choix drastiques. Le piratage est à proscrire strictement. Il n'y a donc qu'un choix possible, celui des logiciels **libres** et **open source**, qui sont les seuls à être eux-mêmes pensés dans une logique d'utilisation adaptée finement aux « besoins ». C'est un point hors de discussion[2]. Il ne sera donc question ici que de logiciels libres, et presque uniquement open-source. Au demeurant, dans la majorité des cas, ces logiciels ont atteint un degré d'efficacité bien supérieur à celui des logiciels commerciaux, quand ceux-ci existent.

A. Le matériau

1. Les textes eux-mêmes

5 Il faut disposer de fichiers comportant les fichiers en « mode texte », c'est-à-dire que chaque caractère est transcrit par un code informatique précis. Il faut savoir qu'il existe une grande variété de codages possibles, ce que l'on ne voit pas forcément au premier coup d'œil, et qui peut provoquer des difficultés graves. Il faut donc utiliser LE code le plus efficace, qui est à la fois le plus complet et tend à être le plus répandu, nommé UTF-8. Vérifier l'encodage et, si besoin est, tout convertir en UTF-8 est une opération préalable absolument indispensable[3].

a) Récupération

6 Le plus simple, quand cela est possible, est de récupérer (télécharger) les fichiers souhaités ; il arrive que cela soit même prévu, mais c'est rare[4] ; il faut souvent « aspirer » un site (bons logiciels libres pour cela[5]), mais ensuite on se retrouve avec une masse complexe qu'il faut complètement reformater. De telle sorte que, dans la plupart des cas, pour obtenir un ensemble propre et homogène, un travail important est nécessaire. Il peut arriver que l'on y parvienne directement avec des expressions régulières[6] (regexp), employées dans un logiciel qui les accepte à l'intérieur de la fonction chercher-remplacer (type gedit ou geany, éventuellement emacs ou mined hors de tout environnement graphique). Mais s'il faut employer des suites d'expressions régulières, et traiter un grand nombre de fichiers, la rédaction d'un script dans un langage simple (type perl ou python) est indispensable : exécution infiniment plus efficace et rapide.

7 Pour les timorés, précisons ceci. L'apprentissage des regexp requiert une journée (au grand maximum deux), l'apprentissage de perl ou de python demande quatre ou cinq jours pour maîtriser les principales fonctions (une semaine en comptant large). Autrement dit : personne ne peut dire « c'est trop long » ou « c'est trop compliqué pour moi », ce n'est ni long ni compliqué.

8 Mais bien souvent, on veut étudier des textes qui n'ont jamais été transcrits en format électronique. Deux situations : soit il s'agit de textes purement manuscrits (ou équivalents), soit de textes imprimés. Dans le premier cas, pas d'autre solution que la saisie manuelle ; dans le second cas, si l'impression n'est pas franchement mauvaise, on utilise la reconnaissance optique de caractères (OCR). Ici, trois étapes successives doivent être examinées.

b) OCR

9 Il faut d'abord créer des fichiers-images, le plus ordinaire étant l'emploi d'un scanner. Pour fixer les idées : avec un scanner à plat ordinaire A4, en manipulant manuellement les ouvrages, on peut facilement numériser 80 pages par heure (minimum) : autrement dit, 500 pages en moins de sept heures, 2500 pages en cinq jours[7]. Il est essentiel de procéder au départ à un réglage très fin du logiciel qui pilote le scanner (xsane sous linux) ; contraste, luminosité, seuils doivent être réglés avec le plus grand soin, pour obtenir les images les plus claires possibles ; on utilisera normalement une résolution de 300 dpi[8], en nuances de gris, le format de sortie importe peu.

10 Deuxième étape, l'OCR proprement dit. La solution libre s'appelle ocropus, elle est parvenue à un stade de développement avancé (soutien de google). Avec quelques compétences informatiques, on installe la toute dernière version (0.4.4). S'il s'agit d'une édition ancienne, c'est-à-dire en gros antérieure à 1800, ou comportant des caractères disparus de l'usage (s en forme de f, lettres liées), ou encore pire des abréviations, il faut utiliser un logiciel d'OCR travaillant uniquement en mode « reconnaissance d'images », le plus connu étant gamera[9].

11 Lorsqu'on a récupéré une sortie, le plus difficile reste à faire : éliminer et corriger les fautes. Même les meilleurs logiciels d'OCR commettent des erreurs en quantité excessive (sauf quand il s'agit d'une sortie d'imprimante récente, parfaite, et avec une police très claire). Les recherches sur les méthodes d'aide à la correction ou de correction semi-automatique sont peu développées et peu actives, il y aurait là un champ important à investir massivement.

12 La méthode habituelle consiste à employer un « dictionnaire », c'est-à-dire en réalité une simple liste de formes correctes dans la langue considérée ; le dictionnaire le plus copieux étant ici le plus intéressant. De telles listes circulent de plus en plus fréquemment pour la plupart des langues « anciennes », il reste à faire cependant. Muni d'une telle liste, il faut employer un logiciel comportant une fonction de correction orthographique (spell check), et lui fournir le dictionnaire de la manière appropriée (sous linux, le programme aspell[10] est généralement utilisé en arrière-plan, logiciel qui accepte les dictionnaires « personnels » formés par une simple liste de mots).

2. Informations sur les textes (= pour chaque fichier)

13 Dès lors que l'on dispose de plus de deux ou trois fichiers, on est fréquemment amené à opérer des sélections, c'est-à-dire à examiner seulement un sous-ensemble, ou à comparer les résultats obtenus à partir de plusieurs sous-ensembles. Pour pouvoir définir ceux-ci, il faut pouvoir procéder à des tris en fonction de critères, autrement dit

il faut que chaque fichier soit défini selon une série de critères formels précis. En termes techniques : indexer les fichiers.

14 Bien entendu, il est infiniment préférable d'attribuer à chaque fichier un nom qui en dise le plus long possible sur son contenu, cela sera utile dans tous les cas.

15 Mais la procédure ordinaire consiste à placer en début de fichier une série d'indications plus ou moins codées, correspondant à chacun des critères choisis.

16 Ici, il faut particulièrement insister pour mettre en garde de la manière la plus expresse contre **une tentation aussi stupide que sournoise** : croire qu'il existerait un système universel d'indexation, applicable à n'importe quel type de texte, de n'importe quelle époque. C'est ce que les bibliothécaires croient (ou font semblant de croire) depuis le 19e siècle, à propos des livres : à cette époque furent inventés deux systèmes d'indexation universelle, la classification DEWEY (1876) puis la CDU (classification décimale universelle, 1895), finalement dépassée par la LCC (Library of Congress Classification, 1897). De la LCC est dérivé le RVM québécois, d'où est issu vers 1980 la « base RAMEAU » utilisée dans les bibliothèques françaises. Il s'agit peut-être d'un mal nécessaire aux yeux des bibliothécaires (dont beaucoup cependant se plaignent des lourdeurs, contradictions, insuffisances de ce système), mais surtout il est manifeste que très peu de lecteurs s'en servent autrement qu'en utilisant des noms propres comme clé d'entrée (clés qui sont de fait indépendantes du système d'indexation). En un mot comme en cent : il s'agit d'un outil absurde dans son principe, qui ne peut pas ne pas l'être et qui n'a finalement comme utilité qu'un usage purement bureaucratique.
L'absurdité de ce pseudo-outil tient fondamentalement à son ambition universelle, qui ne fait que ressusciter le fantasme aussi ancien que récurrent d'une « mathesis universalis », autrement dit l'encyclopédie totale et absolue, ce qui revient à peu près au point de vue de dieu créateur…
Ce nonobstant, une poignée d'informaticiens, essentiellement anglo-saxons, dépourvus autant de connaissances historiques que d'esprit critique, entreprirent de transposer ce modèle à l'indexation des textes informatisés. Ce que l'on appelle la TEI (text encoding initiative), 1987. Il s'agissait d'une époque de découvertes, d'explorations, de tentatives foisonnantes ; à ce titre la TEI était « en contexte » ; si elle avait évolué (comme le SGML - totalement oublié aujourd'hui - remplacé par le HTML, universel), elle pourrait peut-être avoir une utilité, modeste mais réelle ; mais l'évolution qui a eu lieu n'a rien arrangé, compliquant la situation au lieu de la simplifier drastiquement. A la limite, on peut (encore que…) retenir les 25 balises les plus courantes (pas une de plus), pour le reste, SURTOUT ÉVITER. Ce n'est que gaspillage et perte de temps[11]. Au demeurant, il n'existe aucun outil libre open source permettant de procéder à un tel encodage.

17 Il faut tout au contraire partir de l'idée fondamentale que tout corpus de textes anciens est spécifique et nécessite une indexation appropriée, finement pensée et adaptée : c'est la seule voie possible si l'on veut vraiment pouvoir exploiter efficacement et commodément le corpus rassemblé.

18 L'essentiel est de bien comprendre que l'indexation d'un corpus textuel est une opération intellectuelle et non technique, qui requiert une connaissance approfondie des textes concernés, et des idées larges sur les utilisations qui pourront être faites de ce corpus.

19 Chaque logiciel de gestion de corpus textuel comprend et utilise un ou des formats d'entrée précis, il faut s'y conformer pour placer les codes dans un en-tête approprié[12].

20 Une fois le système d'indexation défini, il faut l'appliquer, et là apparaît une difficulté technique : on ne peut pas indexer manuellement plus de quelques dizaines de fichiers ; au-delà, il faut recourir à une procédure au moins semi-automatique, de type text mining, que l'on doit faut mettre au point au coup par coup, ce qui n'est pas trivial[13].

3. Informations sur les mots (= codes pour chaque mot)

21 Le second « niveau d'indexation » concerne les mots pris individuellement. A priori, il n'existe pas plus de limite à l'encodage des mots qu'à celui des textes. Cependant, les deux informations les plus habituelles sont le lemme (alias : entrée de dictionnaire), qui aboutit à une sorte de normalisation des formes, et la catégorie grammaticale, nom, verbe, etc. (alias POS = part of speech[14]). On trouve aussi la fonction syntaxique du mot dans la phrase, et diverses formes de catégorisations (champs sémantiques,...)

22 Ici, bien plus encore que pour les textes, l'indexation ne peut être qu'automatique, et requiert donc que le texte soit prétraité (preprocessing). Il existe pour ce faire, s'agissant du POS et du lemme, des outils ad hoc, en général dénommés taggers, outils conçus autour de procédures de text mining adaptées à cette fin. Un des plus employés (libre mais pas open source) est le tree-tagger, conçu à Stuttgart par le professeur Helmut Schmid. Ce logiciel requiert un « fichier de paramètres », spécifique de chaque langue. Si de tels fichiers existent pour la plupart des langues vivantes courantes[15], il n'en va pas de même pour les langues anciennes, pour lesquelles la situation est très variable.

23 Pour le latin (antique et médiéval), un fichier de paramètres libre et open source a été développé, et se trouve (début 2012) en phase de mise au point finale[16]. Fichier élaboré dans le cadre d'un projet ANR (« OMNIA »), auquel ont collaboré des membres du CNRS (IRHT-Paris, Artehis-Dijon, CRH-Paris) ainsi que des membres de l'École Nationale des Chartes.

B. Les logiciels

24 Il existe une très large panoplie d'outils open source : du plus simple au plus sophistiqué, en commençant par le traitement d'un seul texte dénué de toute information complémentaire, jusqu'à celui d'une base de données fortement enrichie aux deux niveaux principaux. On distinguera ici trois niveaux de difficulté, le premier étant tout à fait élémentaire.

1. Le niveau élémentaire

a) TextSTAT et AntConc

25 De simples « éditeurs de textes », qui acceptent les expressions régulières comme entrée de la fonction « recherche » peuvent être bien utiles (gedit, geany) ; geany permet en outre d'appliquer au texte ouvert n'importe quelle opération (« éditer -> format -> envoyer la sélection vers ») : c'est une possibilité commode et puissante, à condition de connaître suffisamment le maniement des commandes en ligne.

26 Très faciles à utiliser, deux logiciels libres permettent d'obtenir pour le texte ouvert soit la liste triée des formes avec leur fréquence, soit une concordance KWIC[17] :

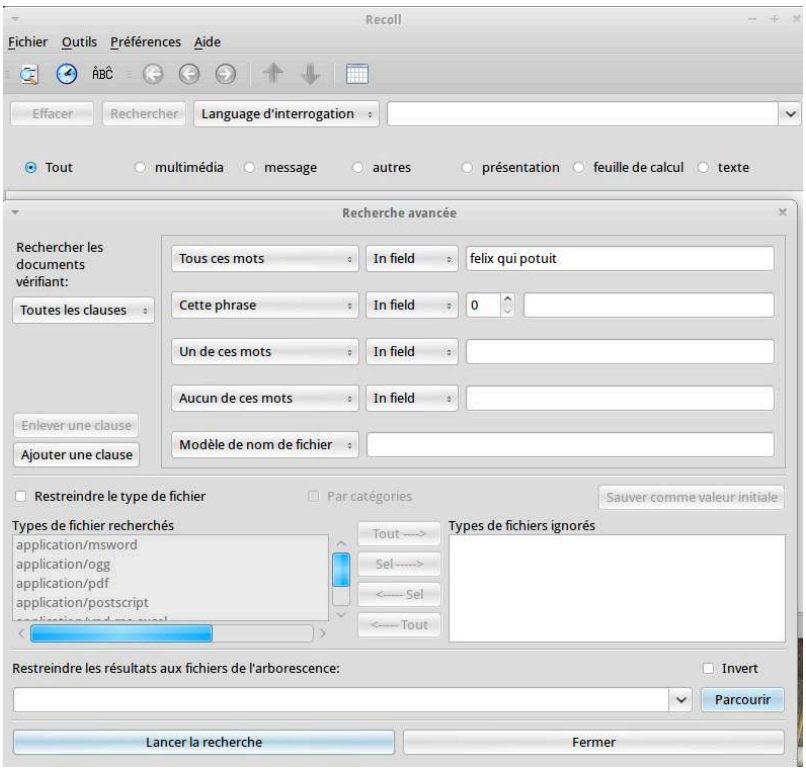
TextSTAT[18](logiciel constitué d'une petite collection de scripts python open source) et AntConc[19] (programme en perl compilé, pas de sources). Très utiles pour un examen rapide (valeur pédagogique maximale !)

27 Mais le problème courant difficile à résoudre est de rechercher un mot ou une phrase dans une importante quantité de fichiers non formatés (i.e de formats divers et variés). Certains outils classiques simples permettent l'opération sur des fichiers-textes purs (uniquement des caractères, pas de compression) : c'est en particulier le cas de trois commandes en ligne puissantes, gawk, sed et grep.

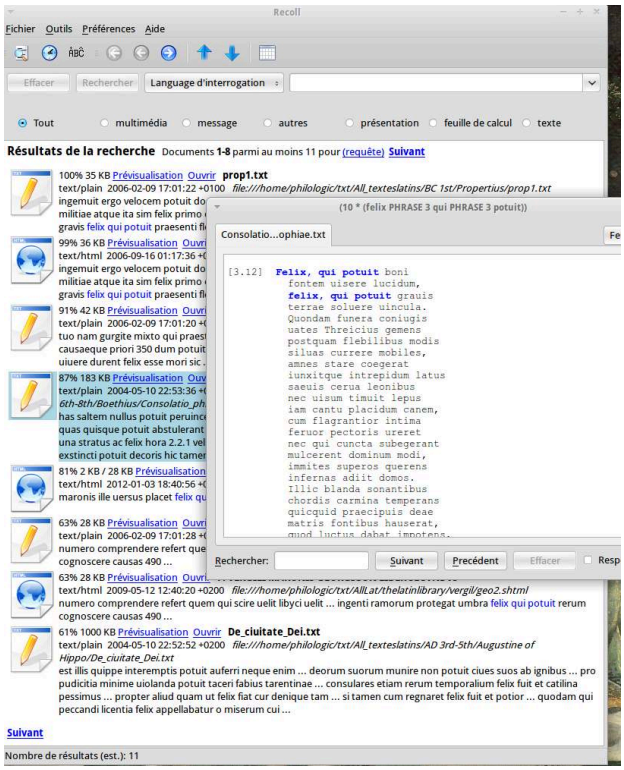
b) Recoll

28 Si les fichiers sont de type quelconque (doc, docx, odt, sxw, rtf, pdf, zip, html, mailbox, ps...), l'outil qui s'impose est **Recoll**[20], un indexeur de fichiers extrêmement rapide et puissant. Recoll fournit une interface très commode à un moteur de recherche très performant, Xapian ; le logiciel indexe d'abord le ou les répertoires indiqués (cela peut prendre un certain temps si l'on veut indexer plusieurs gigaoctets...) ; l'interrogation, simple ou complexe, donne les résultats instantanément. C'est l'outil qui retrouve à la milliseconde une aiguille dans une meule de foin. La seule limite est qu'il ne renvoie qu'une occurrence par fichier : soit on découpe les gros fichiers en morceaux, soit on opère ensuite une recherche sur les fichiers identifiés (avec éditeur de texte ou autre). C'est un must absolu si l'on a des répertoires copieux bourrés de textes en désordre, ou pour lesquels une recherche ordinaire prend entre cinq minutes et une heure...

Recoll : recherche



Recoll : résultats



2. Le niveau intermédiaire

29 On signalera ici deux logiciels libres très performants, organisés en fonction de finalités bien différentes.

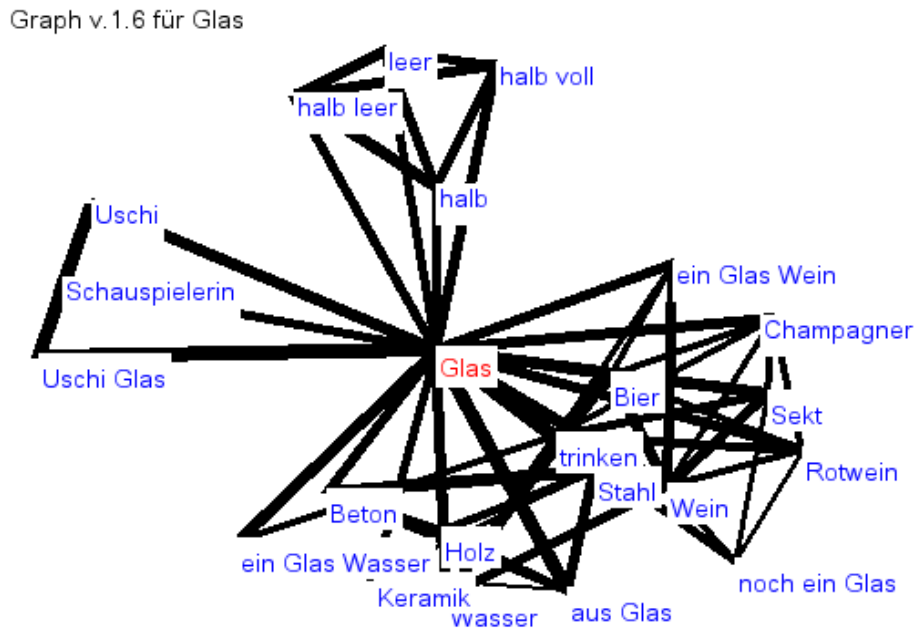
a) CorpusBrowser, ASV Toolbox

30 Le premier est constitué d'un ensemble de programmes java, destinés à manipuler un corpus de textes intégrés sous un format très particulier dans une base de données relationnelle MySQL. Il s'agit des outils développés dans le département d'informatique de l'université de Leipzig ; le programme principal se dénomme **CorpusBrowser**[21], les douze autres forment ensemble le logiciel appelé « ASV Toolbox »[22]. Ces programmes ont été conçus pour permettre une mise en ligne originale et intelligente de vastes corpus de langues, en permettant d'obtenir pour chaque forme les listes de cooccurrents et un « lexicogramme » particulièrement efficace.

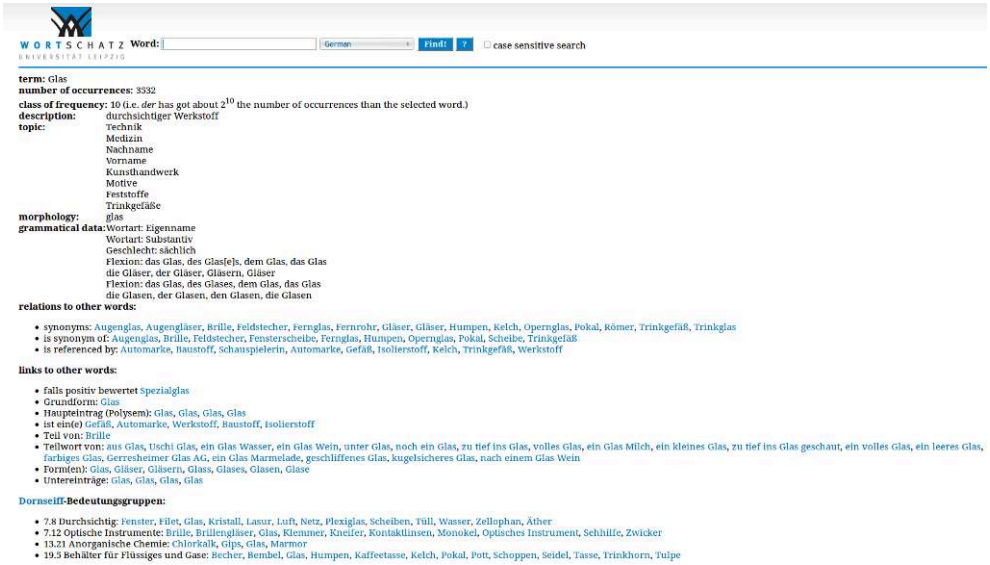
31 Les corpus rassemblés à cet effet sont librement téléchargeables, déjà formatés ; on peut fabriquer son propre corpus avec un petit programme TinyCC[23]. Le programme CorpusBrowser permet d'obtenir sur sa machine le même résultat que celui que l'on trouve en ligne, les douze autres programmes (qui s'emploient sur les mêmes corpus) sont des outils de text mining variés, dont certains fondés sur des hypothèses originales et efficaces.

32 La restriction principale tient au fait que, dans leur état actuel, ces programmes s'appliquent à un corpus entier, sans indexation[24]. Mais il s'agit de programmes open source, on devrait pouvoir introduire une fonction de choix dans un ou des index.

CorpusBrowser : lexicogramme



CorpusBrowser : résultats


WORTSCHATZ Word: German Find! ? case sensitive search

term: Glas
number of occurrences: 3332
class of frequency: 10 (i.e. *der* has got about 2^{10} the number of occurrences than the selected word.)
description: durchsichtiger Werkstoff
topic: Technik
Medizin
Nachname
Vorname
Kunsthandwerk
Motive
Feststoffe
Trinkgefäße
morphology: glas
grammatical data: Wortart: Eigenname
Wortart: Substantiv
Geschlecht: sächlich
Flexion: das Glas, des Glas[e]s, dem Glas, das Glas
die Gläser, der Gläser, Gläsern, Gläser
Flexion: das Glas, des Glases, dem Glas, das Glas
die Glasen, der Glasen, den Glasen, die Glasen
relations to other words:

- synonyms: Augenglas, Augengläser, Brille, Feldstecher, Fernglas, Fernrohr, Gläser, Gläser, Humpen, Kelch, Opernglas, Pokal, Römer, Trinkgefäß, Trinkglas
- is synonym of: Augenglas, Brille, Feldstecher, Fensterscheibe, Fernglas, Humpen, Opernglas, Pokal, Scheibe, Trinkgefäß
- is referenced by: Automarke, Baustoff, Schauspielerin, Automarke, Gefäß, Isolierstoff, Kelch, Trinkgefäß, Werkstoff

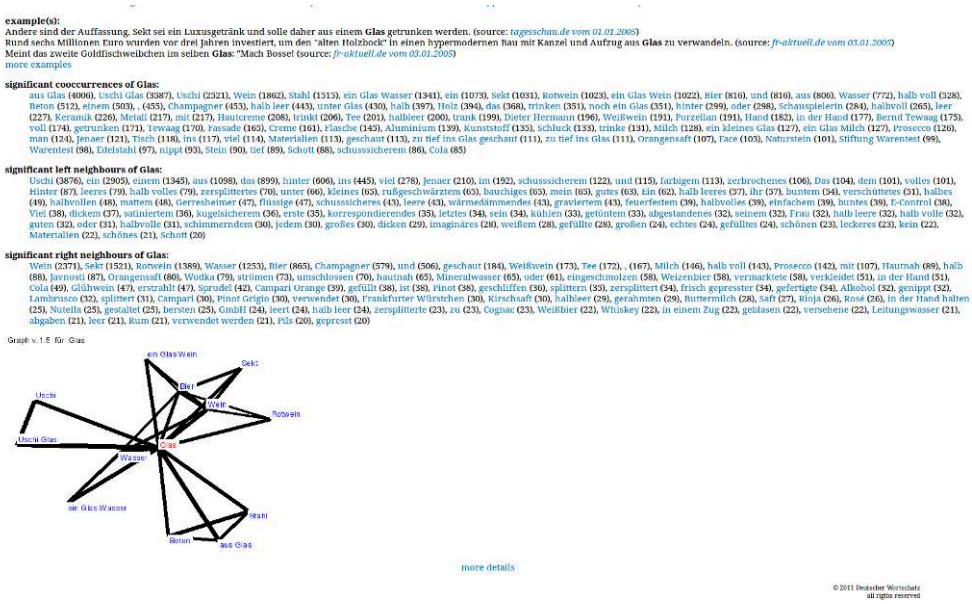
links to other words:

- falls positiv bewertet Spezialglas
- Grundform: Glas
- Haupteintrag (Polysem): Glas, Glas, Glas, Glas
- ist ein(e) Gefäß, Automarke, Werkstoff, Baustoff, Isolierstoff
- Teil von: Brille
- Teilwort von: aus Glas, Uschi Glas, ein Glas Wasser, ein Glas Wein, unter Glas, noch ein Glas, zu tief ins Glas, volles Glas, ein Glas Milch, ein kleines Glas, zu tief ins Glas geschaut, ein volles Glas, ein leeres Glas, farbiges Glas, Gerresheimer Glas AG, ein Glas Marmelade, geschliffenes Glas, kugelsicheres Glas, nach einem Glas Wein
- Form(en): Glas, Gläser, Gläsern, Glass, Glases, Glasen, Glase
- Untereinträge: Glas, Glas, Glas, Glas

Dornseiff-Bedeutungsgruppen:

- 7.8 Durchsichtig: Fenster, Filet, Glas, Kristall, Lasur, Luft, Netz, Plexiglas, Scheiben, Tüll, Wasser, Zellophan, Äther
- 7.12 Optische Instrumente: Brille, Brillengläser, Glas, Klemmer, Kneifer, Kontaktlinsen, Monokel, Optisches Instrument, Sehhilfe, Zwicker
- 13.21 Anorganische Chemie: Chlorkalk, Gips, Glas, Marmor
- 19.3 Behälter für Flüssiges und Gase: Becher, Bembel, Glas, Humpen, Kaffeetasse, Kelch, Pokal, Pott, Schoppen, Seidel, Tasse, Trinkhorn, Tulpe

CorpusBrowser : lexicogramme



b) PhiloLogic

33 Le second est un logiciel comportant des programmes en C et en perl, destiné à mettre en ligne de grands corpus indexés, avec une interface offrant une gamme d'outils de recherche et d'analyse complémentaire. Il s'agit du logiciel **PhiloLogic**[25], développé par Mark Olsen à Chicago, dans le cadre, notamment, du « projet ARTLF » (Robert Morrissey)[26].

34 Programme ancien et solide, utilisé sur un grand nombre de sites, pour des corpus dans les langues les plus variées. L'écriture en perl permet à des non-professionnels de modifier/ajouter des fonctions avec un minimum d'efforts, l'interface en html peut être facilement recomposée en entier.

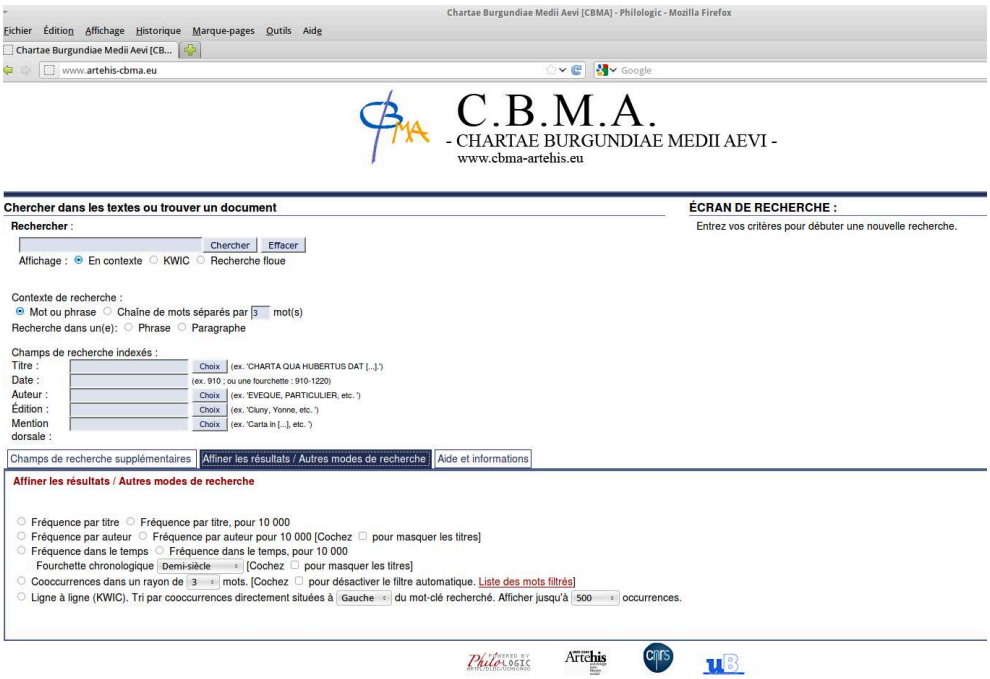
35 Le programme est entièrement conçu pour les mises en ligne, mais on peut s'en servir de la même manière sur sa propre machine (loopback) ; sous linux et sous mac ; l'installation de base est relativement simple et rapide, mais les réglages (nécessaires) le sont nettement moins, et demandent quelques compétences. Les pages qui servent de manuel en ligne sont spartiates.

36 Les textes eux-mêmes sont contenus dans un vaste index organisé sous forme d'arbre (moteur propre, en C), tandis que les champs d'indexation sont dans une base de données MySQL (sur laquelle on peut intervenir directement, si besoin est). Les opérations de sélection d'un sous-corpus par choix dans les index sont particulièrement commodes, les résultats peuvent s'afficher en divers formats, et être triés selon divers critères ; un mécanisme d'affichage des notes en pop-up est prévu, de simples tags html permettent d'afficher des parties de textes en couleur ou avec diverses polices, ou d'insérer des images, l'affichage du texte intégral approprié est toujours possible ; au total, les outils d'indexation, de recherche et d'affichage sont sans équivalent. Deux fonctions complémentaires existent par défaut : un outil de recherche floue (fuzzy search), qui permet par exemple de contourner les graphies fautives, et un calcul de la table de cooccurrence de n'importe quelle forme. Au surplus, Mark Olsen a développé deux annexes particulièrement intéressantes : Philomine[27], qui permet

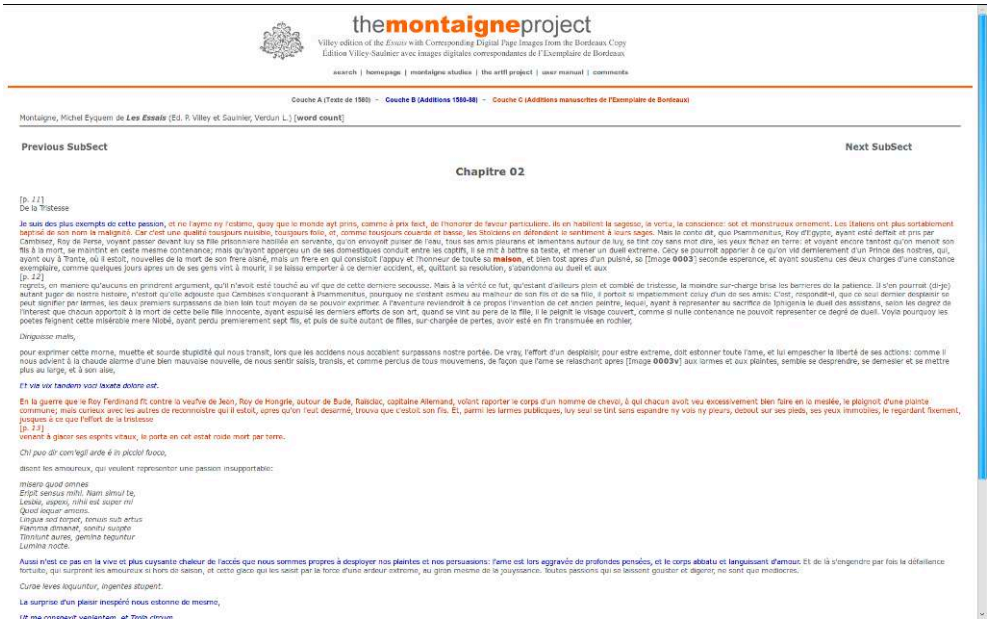
d'utiliser sur les bases intégrées une grande variété d'outils de text mining, et Philoline[28], qui permet de rechercher les passages équivalents dans deux ensembles (identités totales ou approximatives).

37 Si la souplesse est une des grandes forces de PhiloLogic, il faut noter deux limites : d'un côté, le nombre de fichiers intégrables est limité, 32000 normalement et le double grâce à une petite modification ; cela peut paraître beaucoup, mais c'est très insuffisant quand il s'agit de myriades de petits fichiers ; d'un autre côté (et c'est la plus grosse limite), l'indexation ne se fait qu'au niveau global, on ne peut pas étiqueter les mots, donc ni lemmes ni POS. Cette limite restreint considérablement la possibilité d'utiliser PhiloLogic comme outil de récupération de données pour traitement statistique.

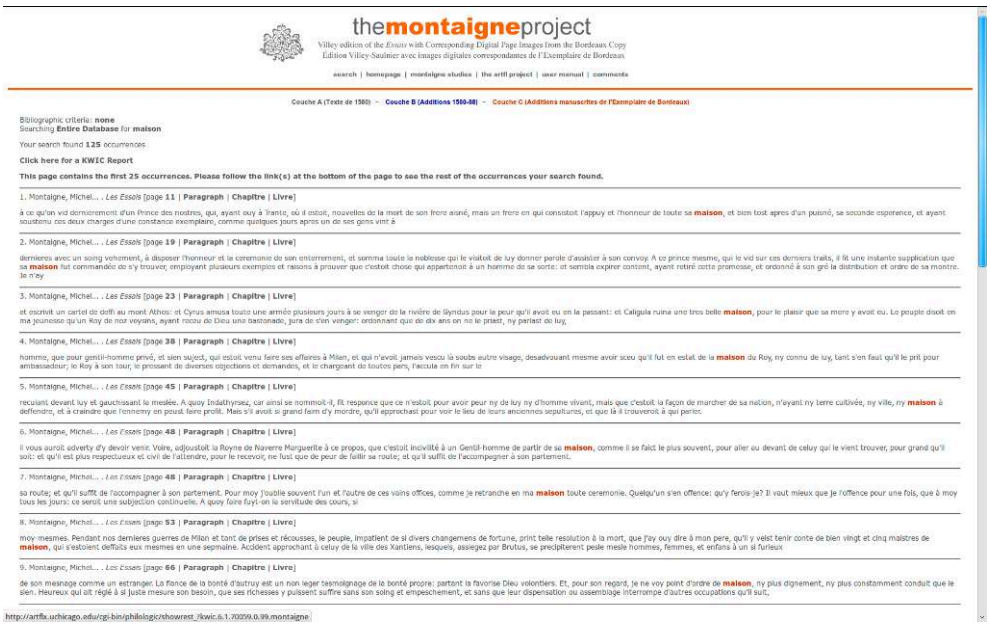
PhiloLogic : recherche dans les CBMA



PhiloLogic : affichage des trois « couches » du texte des Essais



PhiloLogic : résultat d'une concordance dans les Essais



3. Le niveau supérieur

a) CQP, CQP-WEB

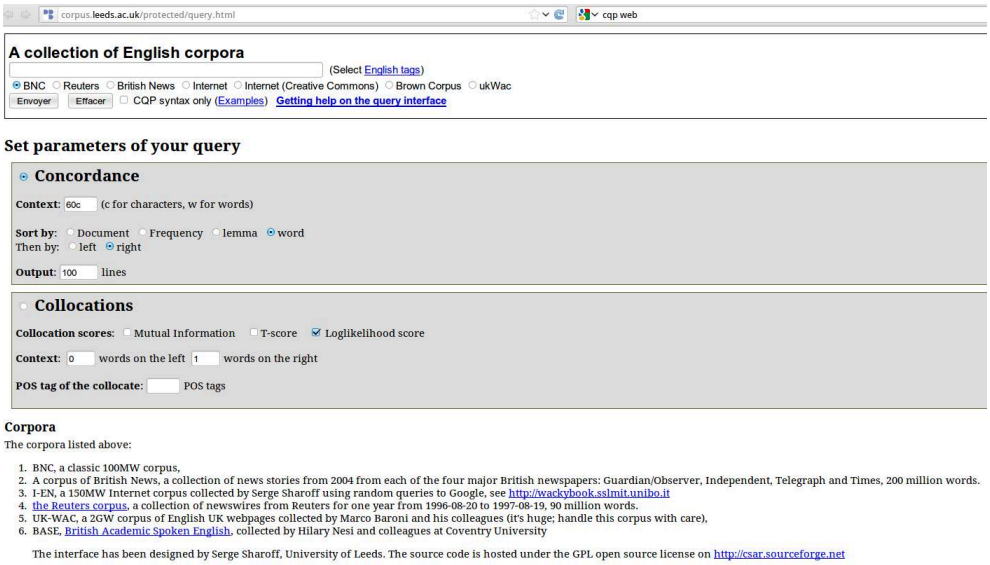
38 Ce niveau correspond à des logiciels capables de traiter simultanément deux niveaux d'indexation. On doit présenter deux logiciels, au demeurant en grande partie complémentaires : Open-CWB et TXM.

39 **Open-CWB** (IMS Open Corpus Workbench[29]) est (comme PhiloLogic) un logiciel ayant une histoire déjà longue : le développement débuta à l'université de Stuttgart en 1993. Après diverses péripéties, le logiciel fut en partie réécrit et passa en open source en

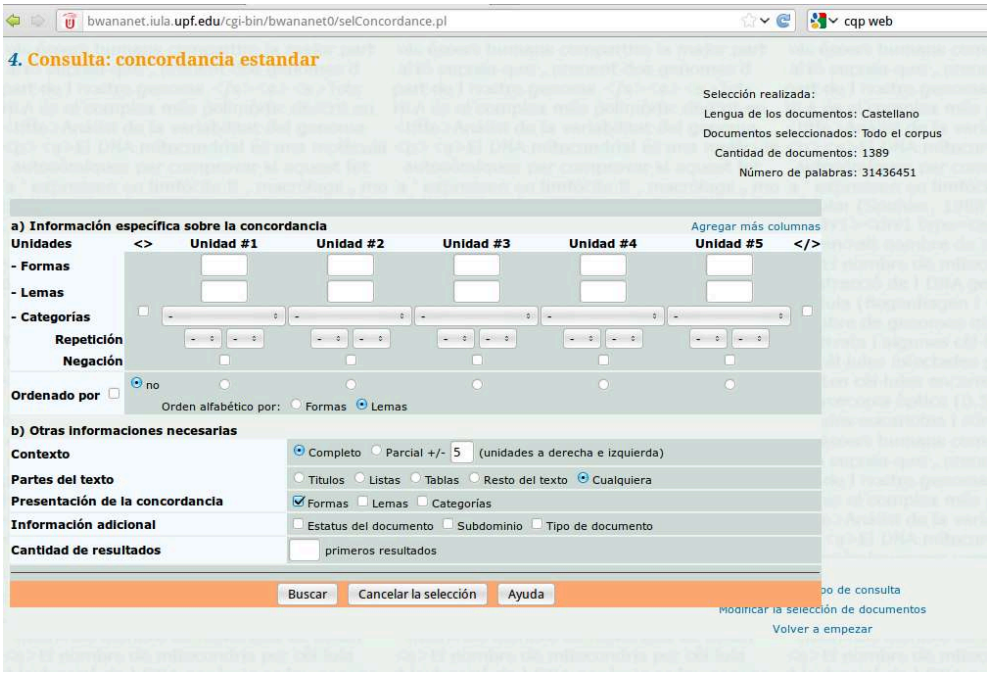
2003, sous la houlette de Stefan Evert[30]. Dès le départ, le projet avait une triple ambition : gérer de très grands ensembles (entre 10 millions et 2 milliards de mots), permettre l'annotation au niveau lexical, comporter un système d'interrogation aussi riche que possible.

40 Le résultat fut la mise au point d'un langage d'interrogation propre, le CQL (Corpus Query Language, lié au programme de base CQP, Corpus Query Processor), qui englobe les expressions régulières (une grande partie) dans un système bien plus riche ; les textes sont stockés sous forme de tables plates (tableaux rectangulaires), ce qui permet de disposer d'autant de tags que l'on veut ; des champs indiquant le nom des fichiers et (éventuellement) de leurs sous-parties sont indexables ; la vitesse de traitement a été spécialement soignée[31]. Le CQP fonctionne uniquement en ligne de commande, mais on arrive à maîtriser l'essentiel du CQL assez rapidement : avec cet outil relativement simple à installer, on dispose d'un moyen particulièrement puissant d'indexation, de stockage et d'interrogation de gros et très gros corpus indexés à deux niveaux.

CQP-WEB : corpus anglais



CQP-WEB : corpus espagnol



41 Les « utilisateurs » (!!!) étant de fait, la plupart du temps, rétifs à l'emploi de la ligne de commande, la nécessité de construire des interfaces dites « conviviales » s'est rapidement fait sentir. Par dessus le marché, la tendance consiste à « mettre en ligne » des corpus, en proposant par construction ladite interface conviviale…

42 Le premier développement de ce type a été entrepris à l'université de Lancaster, d'où est résulté un nouveau logiciel, CQPweb[32], qui utilise Open-CWB comme moteur et fournit à la fois des outils de mise en ligne et une interface graphique, plus diverses fonctions d'analyse supplémentaires. Ce logiciel est d'ores et déjà utilisé pour la mise en ligne de plusieurs corpus en accès libre (ou limité…). Mais, si l'interface est en effet beaucoup plus facile d'accès, l'installation l'est bien moins, et requiert de solides compétences informatiques.

43 Une bibliothèque Rcwb est en développement qui, à terme, devrait permettre d'utiliser directement Open-CWB à partir de R, et faciliter considérablement les opérations de statistique lexicale.

b) TXM, TXM-WEB

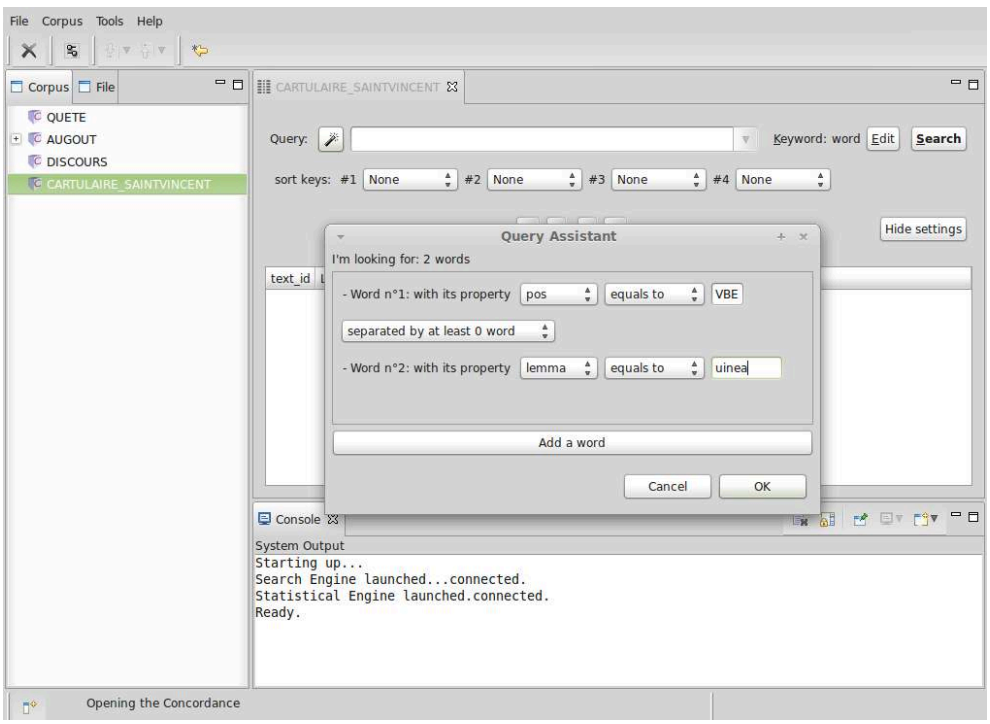
44 Le second projet a été organisé à l'École Normale Supérieure de Lyon dans le cadre d'un projet ANR : le projet « textométrie », en abrégé **TXM**[33]. Le responsable du projet est Serge Heiden. La finalité s'apparente à celle de CQPweb, mais en diffère sensiblement. L'objectif est en effet double : d'une part un outil utilisable directement sur la machine où il est installé (**TXM**[34]) et, d'autre part, un logiciel reprenant les principales fonctionnalités pour une mise en ligne de corpus (**TXM_WEB**[35]). Ces logiciels sont d'ores et déjà en partie opérationnels (début 2012), mais sont encore au stade du développement.

45 Comme dans le cas précédent, TXM est essentiellement une interface construite autour du moteur CQP, dont il utilise toute la richesse et l'efficacité. Les développeurs de TXM ont d'ailleurs contribué au développement du CQP, particulièrement en ce qui concerne

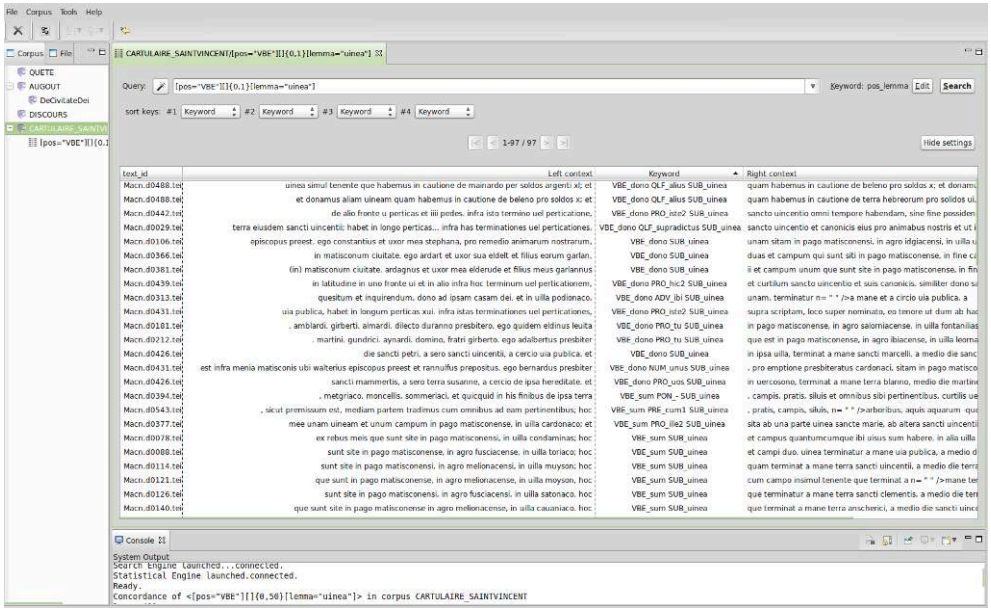
l'utilisation de l'encodage en UTF-8. Trois idées directrices orientent ce projet : * proposer un logiciel très facile à installer sur toutes les plates-formes, * faciliter les ajouts et modifications en intégrant un langage de script - en l'occurrence groovy[36], un dérivé de java - * intégrer dès l'origine l'export des résultats sous **R**[37] (LE grand logiciel libre de statistiques), de manière à rendre le plus commode possible tous les post-traitements statistiques (R fonctionne dans une « boîte » à l'intérieur même de TXM). Autrement dit, tout ce dont un philologue, un linguiste, un sémanticien peuvent rêver ! TXM-WEB devrait, à terme, reprendre la plus grande partie de ces fonctionnalités.

46 Aussi bien pour l'extraction de concordances que pour le calcul de tables de cooccurrences, le champ « recherche » accepte le CQL complet, ce qui à soi seul fournit déjà une puissance considérable. Mais la finalité principale la plus claire de l'interface est justement de permettre un emploi du CQL fortement assisté : on peut se servir de TXM sans connaître un mot de CQL. Les programmeurs ont ajouté une série d'astuces pratiques qui facilitent et accélèrent le travail, par exemple le fait d'activer directement une fonction à partir d'une ligne des résultats d'une autre ; n'importe quel résultat s'exporte sous R en un clic, et on le retrouve aussitôt dans le *workspace*. Cette puissance a cependant quelques contreparties inévitables. La double indexation suppose des fichiers en entrée strictement formatés[38] et parfaitement propres (un seul caractère parasite suffit pour bloquer l'enregistrement de départ) ; la définition d'un sous-corpus demande de l'attention ; pour le moment, certaines requêtes entraînent un temps de traitement quasi-prohibitif (cela devrait se régler). Un manuel copieux est fournit dès à présent, et ne sera pas très difficile à enrichir et à compléter.

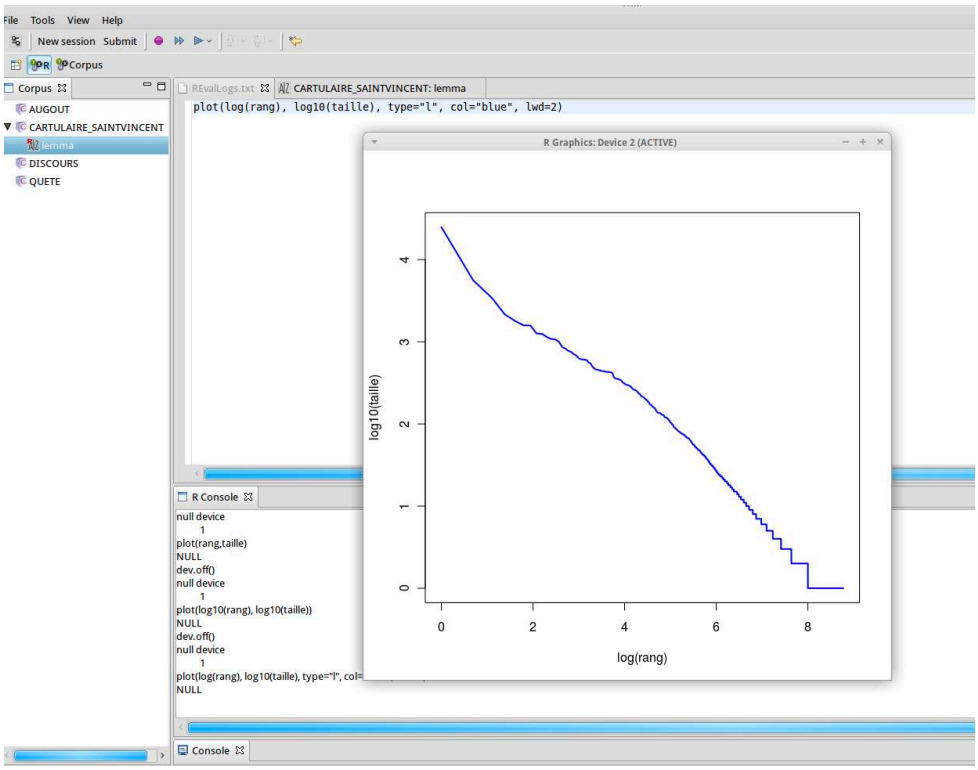
TXM : requête



TXM : concordance complexe



TXM : manipulation des résultats avec R



4. Ce qui manque

47 On doit noter ici que, s'il existe d'ores et déjà une importante panoplie d'outils libres pour manipuler les séries de textes anciens, un total déficit se fait sentir dans le domaine de l'édition de texte, au sens philologique, c'est-à-dire pour enregistrer de manière structurée toutes les variantes d'un texte que l'on trouve dans plusieurs

manuscrits. La difficulté principale me paraît être d'ordre conceptuel : il s'agit de définir la manière de coder et d'enregistrer les variantes au fur et à mesure qu'on les observe en considérant successivement tous les manuscrits qui conservent ce texte. Définition qui ne peut être indépendante des diverses manières dont on souhaite pouvoir ensuite travailler sur ces variantes, ponctuellement ou selon des critères à définir. Il faudrait pouvoir constituer un groupe de philologues-éditeurs ayant au surplus de bonnes notions d'informatique pour espérer parvenir à une telle définition. L'écriture du logiciel ne devrait pas poser de problèmes insurmontables en elle-même. Il est vrai que, pour le moment, cette alliance de compétences est excessivement rare, mais la lacune risque de faire de plus en plus sentir.

Conclusion

48 Des outils libres de manipulation informatique de corpus de textes anciens, des plus modestes aux plus massifs, existent et sont fonctionnels. Le panorama se modifie d'ailleurs constamment, des développements supplémentaires se produisant ici ou là. On ne peut que regretter que les historiens et philologues le sachent à peine, et ne les utilisent qu'à doses homéopathiques.

49 Le point décisif, sur lequel on n'insistera jamais assez, est que toutes ces manipulations requièrent infiniment plus de compétence intellectuelle et disciplinaire (dans le cas qui nous occupe : historique et sémantique) que d'habileté technique. Celle-ci résulte essentiellement de l'expérience, qui ne s'acquiert précisément qu'en manipulant les outils. Pour chaque corpus, il faut bien comprendre et saisir les spécificités, et avoir une idée des procédures souhaitables et possibles, c'est seulement à cette condition que l'on peut 1. choisir les bons outils, 2. s'en servir efficacement. Les prétraitements (indexation(s), pos-tagging en particulier) demandent beaucoup de réflexion et de temps, mais conditionnent la pertinence des sorties des logiciels.

50 Cela dit, et même si les développements récents (TXM au premier chef) visent la facilité d'installation et de manipulation, c'est une lourde erreur de croire que l'on puisse vraiment obtenir le meilleur résultat de tous ces outils sans jamais se salir les mains ! et il faut donc répéter ici ce qui vient d'être dit au paragraphe précédent : les codes ou langages informatiques les plus courants (regexp, html, perl ou python, R) s'apprennent (quant aux bases) très rapidement, mais ne deviennent vraiment utiles que dans la mesure où l'on en acquiert l'expérience (au demeurant, pour des « littéraires », experts en approximations quand ce n'est pas en n'importe quoi, ces langages sont une excellente école de réalisme : quant ça ne marche pas, c'est qu'il y a une erreur !).

51 Il importe cependant, pour terminer, de souligner l'optimisme dont il y a lieu de faire preuve, ici. Dans un environnement général morose, pour ne pas dire inquiétant, la situation globale des logiciels libres de manipulation des textes se présente de manière particulièrement favorable : depuis le milieu des années 90 (en gros) une importante série de logiciels ont vu le jour, se sont développés, et continuent leur progression. Dans bien des cas, la collaboration avec les développeurs permet toutes les améliorations souhaitées. Il ne reste qu'à retrousser les manches, les textes anciens vont apparaître sous un jour complètement différent, la possibilité de cerner le sens qu'ils avaient (pas celui que nous leur imposons) est aujourd'hui à notre portée.

NOTES

1. Il s'agit seulement d'une brève note d'orientation, destinée à fournir, dans un format compact, un aperçu général des problèmes et des solutions (point de vue personnel). Fait suite à la note « Pour un corpus de textes latins en ligne ». Sera suivie - dans un délai 'raisonnable' - d'une note sur les procédures d'analyse sémantique des textes anciens. Tous les logiciels décrits ont été testés - utilisés sous linux ; on signale cependant les cas où le logiciel est disponible pour Mac$ et/ou Window$.

2. Du reste, l'offre commerciale dans ce domaine est quasi-inexistante. Le seul SGDBR classique offrant des outils d'analyse de texte est PostgreSQL (libre et open source), qui étend le langage d'interrogation sensiblement au-delà du SQL, mais cela reste très limité.

3. Les éditeurs de textes courants (gedit, geany...) détectent en général le code d'un fichier à ouvrir ; gedit fournit une case à cocher pour choisir le codage d'un (ré)enregistrement ; libreoffice, en cas de difficulté, ouvre une boîte de dialogue, il faut faire des essais. Il n'existe pas (à ma connaissance) de programme capable de détecter le codage d'un fichier-texte. En cas de grosse difficulté, une solution technique consiste à ouvrir le fichier avec un « éditeur hexa » (type ghex2), à identifier le codage informatique des caractères qui posent problème, et à rechercher sur internet la correspondance avec un code précis. Pour la conversion d'un fichier-texte pur, il existe d'excellents outils en ligne de commande (recode ou iconv). NOTA : **pour tous les termes techniques, consulter wikipedia** !

4. C'est le cas des Chartes Bourguignonnes (CBMA) : http://www.artehis-cnrs.fr/page-documentaire-CBMA.

5. wget en ligne de commande, httrack avec une interface graphique très commode.

6. Tout le monde utilise les « caractères jokers », type * ou ?, que tous les traitements de texte connaissent ; les expressions régulières sont une extension-généralisation de ce principe : elles permettent de rechercher (et/ou remplacer) des chaînes de caractères définies de manière simple ou complexe, selon une syntaxe formelle précise. Les bases sont simples, mais on peut atteindre un degré de complexité considérable.

7. Estimation minimale, en une semaine on doit pouvoir atteindre 4000 pages ; sans doute un peu plus long s'il s'agit d'un grand format nécessitant un scanner A3.

8. C'est la résolution la plus habituelle ; on peut toujours passer à 600 dpi (fichiers 4 fois plus lourds), surtout si les caractères sont très petits. Il y a toujours moyen de diminuer la résolution d'un fichier-image, mais pas de l'augmenter !

9. http://gamera.informatik.hsnr.de/index.html

10. aspell ne se borne pas à identifier les mots absents du dictionnaire, il propose la forme « la plus proche » (calculée selon une procédure formelle). aspell est utilisé pour la correction orthographique par gedit ; geany (et Abiword) utilisent une autre librairie, enchant., qui vise à constituer une synthèse des outils existants. LibreOffice utilise myspell.

11. En théorie, la TEI n'est pas destinée à indexer les contenus, mais seulement les formes ; mais la précision utopique à laquelle elle prétend parvenir implique de facto une analyse du contenu. Au surplus, pour qu'un formalisme soit utile, il doit être fondamentalement équilibré (cf. Claude Shannon), or la TEI ne respecte aucun équilibre rationnel.

12. Les informations placées dans cet en-tête, aussi bien d'ailleurs que toutes les informations liées à chaque mot, se placent dans des « balises xml ». Ce terme d'allure rébarbative désigne seulement un ensemble de règles d'écriture très simples, qui assurent l'homogénéité et la lisibilité de ces informations, bien distinctes du texte lui-même. Ces quelques règles (on parle pompeusement de « syntaxe ») s'apprennent parfaitement en moins d'une heure ! Pour vérifier

qu'elles sont exactement respectées, on peut utiliser l'un ou l'autre des « éditeurs xml », qui comportent une fonction de « validation » (e.g. xmlcopyeditor).

13. Un des plus puissants logiciels libres de text mining - data mining est weka (http://www.cs.waikato.ac.nz/ml/weka). Ce logiciel implémente la plupart des procédures connues, mais il existe d'autres logiciels, des bibliothèques spécialisées pour les principaux langages, etc. Le text mining demande une note de présentation à lui seul (pour le moins), on peut consulter : http://www.cs.sunysb.edu/~cse634/presentations/TextMining.pdf (bonne introduction).

14. En français, « étiquetage morpho-syntaxique ».

15. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

16. http://glossaria.eu

17. = key word in context, autrement dit une ligne pour chaque occurrence.

18. http://neon.niederlandistik.fu-berlin.de/textstat/ (tous OS - Operating Systems -= Linux + Mac$ + Window$).

19. http://www.antlab.sci.waseda.ac.jp/software.html (tous OS).

20. http://www.lesbonscomptes.com/recoll/. Linux. Pour obtenir l'efficacité maximale, il faut installer cinq paquets supplémentaires : pdftotext, ghostscript, antiword, catdoc et unrtf.

21. http://corpora.uni-leipzig.de/download.html. Programme java, donc théoriquement tous OS ; seulement testé sous linux.

22. Voir : http://nl.ijs.si/is-ltc06/proc/13_Richter.pdf. ET http://wortschatz.uni-leipzig.de/~cbiemann/software/toolbox/index.htm

23. http://wortschatz.uni-leipzig.de/~cbiemann/software/TinyCC2.html

24. En fait, l'indication de la source de chaque texte d'un corpus est bel et bien enregistrée, et elle s'affiche avec les exemples ; mais on ne peut utiliser cette information pour des sélections et des tris.

25. http://sites.google.com/site/philologic3/ La dernière version est la 3.2 (août 2010). Une version 4, en python, est en préparation (https://github.com/rwhaling/libphilo/tree/master/python/philologic). Linux et Mac$.

26. http://www.lib.uchicago.edu/efts/ARTFL/

27. http://code.google.com/p/philomine/

28. http://code.google.com/p/text-pair/

29. http://cwb.sourceforge.net/. Conçu pour linux, une version Mac$.

30. http://www.iula.upf.edu/materials/080925evert_hdout.pdf

31. Ce système présente d'évidentes analogies avec les systèmes de bases de données relationnelles classiques ; la différence fondamentale tient au langage d'interrogation, beaucoup plus puissant et infiniment mieux adapté aux textes que le SQL.

32. http://cwb.sourceforge.net/files/EvertHardie_CL2011.pdf et http://lancs.ac.uk/staff/hardiea/cqpweb-paper.pdf. Tous OS.

33. http://textometrie.ens-lyon.fr/

34. http://textometrie.ens-lyon.fr/spip.php?article60 et http://sourceforge.net/projects/txm/files/software/TXM/. Linux + Window$ (paquets deb pour linux, fichier exe pour Window$).

35. http://sourceforge.net/projects/txm/files/software/TXM %20WEB/. Linux.

36. http://groovy.codehaus.org/

37. http://www.r-project.org/. Tous OS.

38. Les développeurs ont réalisé un gros effort pour rendre quasi-automatique l'enregistrement de fichiers de multiples types (sorties de logiciels de prétraitement), et cet effort se poursuit. Mais il est aussi prévu un codage simple, très proche du format requis par CQP (TXM-w), que l'on peut produire avec un script vite écrit.

INDEX

Mots-clés : base de données textuelle, texte latin, recoll, philologic, CQP, TXM

AUTEUR

ALAIN GUERREAU

CNRS - CRH