Alignements unilingues avec MEDITE

Jean-Gabriel Ganascia, Julien Bourdaillet

LIP6 - Université Pierre et Marie Curie (Paris VI) –
8 rue du Capitaine Scott, 75015 Paris – FRANCE

Abstract

MEDITE was initially designed to fill the genetic criticism requirements. It has succeeded, since many researchers use it now ; it has been said that MEDITE changes in some respect the way geneticists deal with their texts. Moreover, it appears that various others who deal with textual data (social scientist, philologists, epistemologists etc.) need an efficient monolingual aligner with an interface adapted to textual evolution studies. Nevertheless, to extend its fields of application, MEDITE has to adapt its algorithms and to incorporate new functionalities. For instance, MEDITE needs to deal with highly repetitive texts and to ignore punctuations or diacritics. This paper describes the algorithmic modifications of MEDITE which make it a general textual aligner.

Résumé

MEDITE a été initialement conçu pour répondre aux besoins de la critique génétique. Il a rempli son office, puisque beaucoup de chercheurs l'utilisent maintenant ; il a même été dit que MEDITE modifiait quelque peu la façon dont les généticiens opèrent sur les textes. De plus, il apparaît que nombreux sont ceux qui traitent de données textuelles (sociologues, philologues, épistémologues etc.) et ont besoin d'un analyseur unilingue efficace avec une interface adaptée à l'étude des évolutions textuelles. Néanmoins, pour étendre ses champs d'application, MEDITE doit adapter ses algorithmes et incorporer de nouvelles fonctionnalités. Par exemple, MEDITE a besoin de traiter des textes contenant de nombreuses répétitions et d'ignorer les signes de ponctuation ou les signes diacritiques. Cet article décrit les modifications algorithmiques qui font de MEDITE un aligneur textuel général.

Mots-clés : alignement de texte, génétique textuelle, philologie, détection d'homologies dans les séquences

1. Introduction

Le logiciel MEDITE (Ganascia et al. 2004 a ; Ganascia et al. 2004 b) fût originellement conçu pour faciliter la critique textuelle génétique (deBiasi, 2000) en comparant automatiquement des états linéaires de textes d'un même auteur à différentes étapes de la conception d'une œuvre. Grâce à MEDITE, le rapprochement opéré manuellement auparavant par les généticiens, se trouve automatisé, à condition toutefois que l'on ait défini des suites d'états textuels linéaires à partir du dossier génétique en cours d'étude. Disponible à tous (cf. http://www-poleia.lip6.fr/~ganascia/medite) MEDITE est utilisé par de nombreux chercheurs. Il a été testé sur plusieurs dossiers de genèse. En confrontant les résultats obtenus avec des interprétations manuelles, on constate que, la plupart du temps, il retrouve les déplacements, les insertions, les suppressions et les remplacements déjà identifiés par les généticiens du texte. De plus, dans les prochains mois, le logiciel MEDITE pourrait s'intégrer à un environnement pédagogique, pour aider les collégiens ou les lycéens à comprendre la nature du travail de réécriture des grands écrivains. Enfin, il est prévu d'adjoindre MEDITE à des éditions génétiques électroniques, afin de permettre aux lecteurs de procéder eux-mêmes aux comparaisons qu'ils jugent nécessaires.

Par ailleurs, quoiqu'il ait été initialement conçu en regard des besoins de la critique génétique, MEDITE s'est révélé utile à beaucoup d'autres chercheurs procédant à des analyses textuelles. Ainsi, Cécile Brousse l'utilise actuellement pour comparer des articles de journaux gratuits aux dépêches d'agences de presse dont leurs rédacteurs se sont inspirés ; le but est d'apprécier la nature exacte du travail journalistique. De même, en se référant à l'« Éloge de la variante » de Bernard Cerquiglini (Cerquiglini, 1989), des médiévistes souhaitent comparer des variantes de textes de la littérature médiévale pour saisir le sens des écarts (Ganascia 2006). Enfin, travaillant avec des épistémologues et historiens des sciences, Jean-Louis Lebrave confronte, toujours à l'aide de MEDITE, les cahiers d'expérience de Claude Bernard et la relation de ces mêmes expériences dans ses écrits ultérieurs (cf. figure 1). Dans tous les cas, il s'est trouvé que les applications d'un aligneur unilingue sont nombreuses et qu'il n'existait pas d'aligneur approprié disponible.

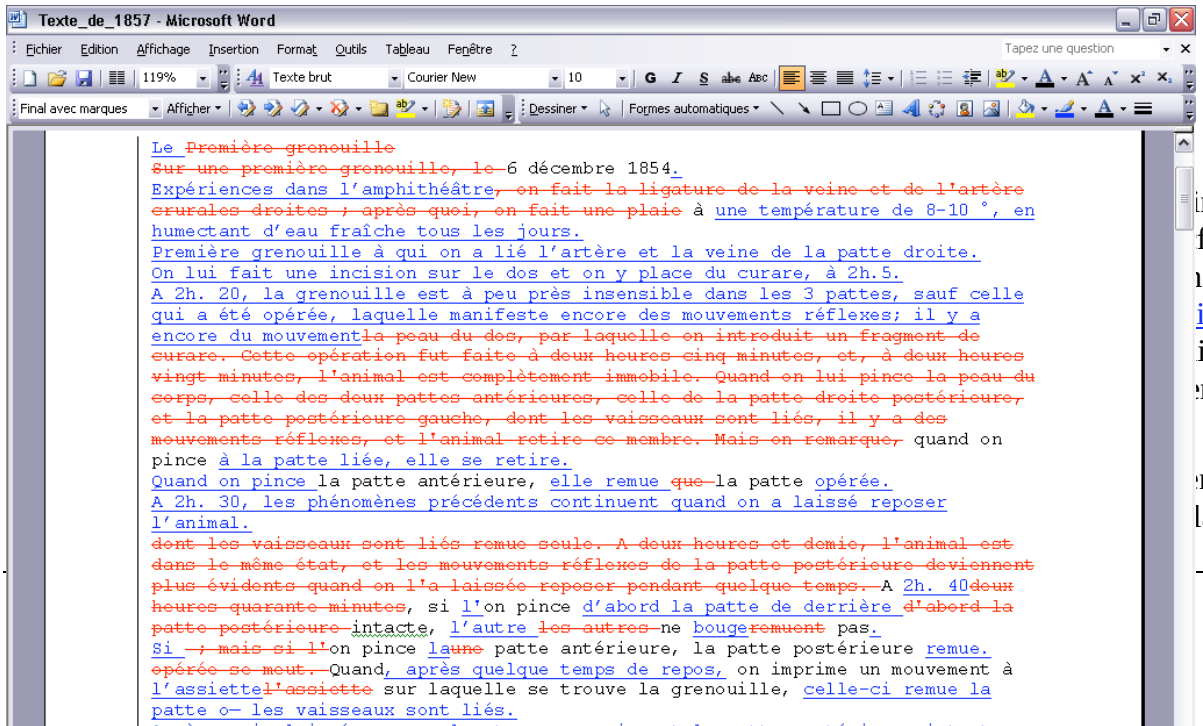
Cahier d'expériences de 1854	**Texte de 1857 relatant ces expériences**
Le 6 décembre 1854. Expériences dans l'amphithéâtre à une température de 8-10 °, en humectant d'eau fraîche tous les jours. Première grenouille à qui on a lié l'artère et la veine de la patte droite. On lui fait une incision sur le dos et on y place du curare, à 2h.5. À 2h. 20, la grenouille est à peu près insensible dans les 3 pattes, sauf celle qui a été opérée, laquelle manifeste encore des mouvements réflexes ; il y a encore du mouvement quand on pince à la patte liée, elle se retire. Quand on pince la patte antérieure, elle remue la patte opérée. À 2h. 30, les phénomènes précédents continuent quand on a laissé reposer l'animal. À 2h. 40, si on pince d'abord la patte de derrière intacte, l'autre ne bouge pas. Si on pince la patte antérieure, la patte postérieure remue. Quand on imprime un mouvement à l'assiette sur laquelle se trouve la grenouille, celle-ci remue la patte o— les vaisseaux sont liés. Après avoir laissé reposer longtemps, en pinçant la patte postérieure intacte, on produit un mouvement dans la patte liée, mais en recommençant aussitôt après, on ne produit rien. Après un quart d'heure de repos, quand on pince la patte postérieure intacte, l'autre remue ; il en est de même pour les membres antérieurs.	Première grenouille Sur une première grenouille, le 6 décembre 1854, on fait la ligature de la veine et de l'artère crurales droites ; après quoi, on fait une plaie à la peau du dos, par laquelle on introduit un fragment de curare. Cette opération fut faite à deux heures cinq minutes, et, à deux heures vingt minutes, l'animal est complètement immobile. Quand on lui pince la peau du corps, celle des deux pattes antérieures, celle de la patte droite postérieure, et la patte postérieure gauche, dont les vaisseaux sont liés, il y a des mouvements réflexes, et l'animal retire ce membre. Mais on remarque, quand on pince la patte antérieure, que la patte dont les vaisseaux sont liés remue seule. À deux heures et demie, l'animal est dans le même état, et les mouvements réflexes de la patte postérieure deviennent plus évidents quand on l'a laissée reposer pendant quelque temps. À deux heures quarante minutes, si l'on pince d'abord la patte postérieure intacte, les autres ne remuent pas ; mais si l'on pince une patte antérieure, la patte postérieure opérée se meut. Quand, après quelque temps de repos, on imprime un mouvement à l'assiette sur laquelle se trouve la grenouille, cet ébranlement produit des mouvements dans la patte qui a été opérée. Mais, si aussitôt on remue de nouveau l'assiette, le phénomène n'a plus lieu. Enfin, on laisse reposer la grenouille un quart d'heure, et quand on pince la patte postérieure intacte, l'autre, dont les vaisseaux sont liés, remue. Il en est de même lorsqu'on pince les pattes antérieures.

***Figure 1** : textes de Claude Bernard numérisés par J-L Lebrave à partir des travaux de Grmek*

Plus exactement, s'il apparaît que le principe d'un aligneur unilingue ressemble à celui de la comparaison de versions telle qu'elle est implantée dans les traitements de texte du commerce comme Microsoft Word, toutefois, il s'en distingue :

- par la finesse requise pour ses analyses,
- par le fait qu'il doit repérer les déplacements, et pas seulement les insertions et les suppressions,
- par le recensement de toutes les transformations qu'il lui faut rassembler dans un fichier pouvant faire l'objet de traitements statistiques ultérieurs et, enfin,
- par une interface appropriée à l'analyse comparée de textes.

Pour s'en convaincre, il suffit d'examiner avec attention la comparaison des deux textes de Claude Bernard donnés ci-dessus avec Microsoft Word (cf. figure 2). L'interface rend le travail malaisé. Indépendamment de ces questions de commodité, il se trouve que beaucoup de similitudes entre les textes ne sont pas repérées.



comparaison de variantes, en philologie médiévale, il faut lui faire subir quelques transformations pour l'adapter aux différents usages, par exemple autoriser une insensibilité à la casse, aux signes diacritiques ou aux séparateurs (blancs ou ponctuations) entre mots, voire, pour la philologie médiévale, aux variations orthographiques. C'est aux difficultés rencontrées dans la création d'un analyseur unilingue et aux évolutions de MEDITE qu'est consacré cet article.

2. Modification de l'algorithme de MEDITE

Comme nous venons de le voir, le programme MEDITE prend en entrée deux états d'un même texte de façon à repérer les transformations qui font passer de l'un à l'autre, ou, plus exactement, l'ensemble minimal de transformations qui fait passer du texte initial au texte « corrigée ». Formulé de la sorte, le problème apparaît très proche de celui posé par le calcul des « distances d'édition » (Sankoff/Kruskal, 1983 ; Crochemore/Rytter, 1994 ; Ganascia, 2001). Cependant, afin de réduire la complexité et de répondre au mieux au problème posé, nous n'avons pas eu recourt aux distances d'édition, et nous avions initialement conçu un algorithme spécifique qui procèdait en trois étapes (Ganascia et al. 2004a) :

1. Détection des blocs communs maximaux disjoints
2. Identification des déplacements et des pivots
3. Calcul des insertions, des suppressions et des remplacements

2.1. Détection des blocs communs maximaux disjoints

La détection des blocs maximaux fait appel à des algorithmes classiques de recherche d'homologies dans les séquences. Nous n'insisterons donc pas sur la mise en œuvre de ces algorithmes, sauf à dire que les algorithmes efficaces employés sont fondés sur la notion d'arbre des suffixes (Ukkonen, 1995), qu'ils ont une complexité linéaire et qu'il y a parfois des recouvrements entre blocs maximaux. Ainsi, les deux chaînes « Il a avalé » et « Il avala » donnent deux blocs maximaux, |Il a | et | aval| qui se recouvrent partiellement. Pour obtenir des blocs maximaux disjoints, il faut introduire une césure en évitant, dans la mesure du possible, la fragmentation des mots.

Notons toutefois que la notion de bloc commun fait référence à une séquence commune au texte source et au texte cible ; or, lorsque les séquences sont répétées, il se peut que des séquences se recouvrent sur certaines de leurs occurrences, mais pas sur toutes, ce qui induit des phénomènes de masquage de blocs par d'autres, qui dégradent localement l'alignement en conduisant à omettre des alignements à l'intérieur de blocs maximaux. La solution adoptée consiste à appliquer récursivement l'algorithme d'alignement entre les blocs alignés, c'est-à-dire entre les pivots (voir section 2.1.2) jusqu'à ce qu'aucun alignement ne soit plus possible.

2.2. Identification des déplacements et des pivots

Parmi l'ensemble des blocs communs maximaux disjoints, certains se retrouvent dans le même ordre dans les deux textes, le texte source et le texte corrigé, tandis que d'autres apparaissent déplacés. Ainsi, si nous avons la séquence de blocs maximaux B_1 B_2 B_3 B_4 B_5 dans le texte source et la séquence B_2 B_3 B_1 B_4 B_5 dans le texte corrigé, on peut inférer que le bloc B_1 a vraisemblablement été déplacé, même si cette appréciation est subjective, car on pourrait tout autant dire que ce sont les blocs B_2 et B_3 qui ont été déplacés. L'algorithme que nous avons mis en œuvre détermine les blocs déplacés en minimisant l'amplitude des

déplacements mesurée en nombre de caractères. À l'issue de cette phase, on distingue parmi les blocs maximaux disjoints, des blocs dits « déplacés » et des blocs qui apparaissent dans le même ordre dans le texte source et dans le texte cible. Ces derniers sont appelés les « blocs pivots » de la comparaison.

2.3. Calcul des insertions, des suppressions et des remplacements

Une fois déterminés les « blocs pivots » et les « blocs déplacés », il reste à calculer les *suppressions*, les *insertions* et les *remplacements* en se référant aux « blocs pivots » considérés comme les parties stables des textes.

3. Algorithme d'alignement de blocs

3.1. Limitations de l'algorithme de MEDITE

La première version de l'algorithme de MEDITE fonctionnait parfaitement sur des textes littéraires comprenant très peu, voire pas du tout de répétitions. Dès que nous l'avons fait fonctionner sur des textes plus ordinaires, en particulier sur les cahiers d'expérience de Claude Bernard (voir figure 1), qui se révèlent être extrêmement répétitifs, il s'est trouvé que les blocs maximaux communs aux textes sources et cibles avaient plusieurs occurrences à la fois dans le texte source et dans le texte cible, ce qui perturbait la procédure de calcul des pivots. Ceci nous a conduit à modifier le principe sur lequel repose l'algorithme de détection de pivots de MEDITE afin de choisir l'appariement le plus propice des blocs maximaux disjoints.

3.2. Formalisation des notions de séquence et d'alignement maximal

Rappelons le problème : les blocs homologues disjoints ont été repérés à l'issue de l'étape 1 ; nous nous trouvons donc en présence de 2 séquences de blocs appartenant à l'ensemble des blocs maximaux disjoints $B = \{B_i, i \in [1 .. n]\}$, les B_i étant éventuellement répétés plusieurs fois dans la séquence source et dans la séquence cible. Par exemple, nous pouvons avoir les séquences $S_1 = B_1, B_2, B_3, B_1, B_4, B_5$ et $S_2 = B_2, B_1, B_3, B_5, B_4$.

Définition : Plus formellement, chaque *séquence* se définit par une fonction d'occurrence O

$O : S \times N \rightarrow B \cup \{\varepsilon\}$, O étant surjective, mais pas injective avec

$\forall\ k \in [1, |\ S_i\ |]\ O(S_i, k) \in B$ sinon $O(S_i, k) = \varepsilon$)

Cette définition étant posée, il faut trouver un alignement « optimal » entre deux séquences, c'est-à-dire un appariement qui minimise une fonction de coût qui correspond aux blocs n'appartenant pas à un alignement. Pour préciser cette notion, nous allons définir formellement la notion d'alignement et celle de coût.

Définition : un *alignement entre deux séquences* S_1 et S_2 se définit par une fonction $align : [1 .. |\ S_1|] \rightarrow [1 .. |\ S_2|] \cup \{0\}$ telle que

1- Si $align(i) = i'$ alors $O(S_1, i) = O(S_2, i')$
2- Si $i > j$ alors $align(i) > align(j)$ ou $align(i) = 0$

Le coût d'un alignement correspond à la somme des blocs absents de l'alignement, c'est-à-dire l'ensemble des blocs de la séquence initiale dont l'image est nulle plus l'ensemble des blocs de la séquence finale sans correspondance avec un bloc de la séquence initiale.

Définition : soit un alignement align, le coût de cet alignement se calcule comme suit :

$$\text{coût(align)} = \sum_{k=1}^{|S_1|} \left|O(S_1,k)\right| \cdot \neg(align(k) = 0) + \sum_{k=1}^{|S_2|} \left|O(S_2,k)\right| \cdot \neg(\exists i \quad align(i) = k)$$

À cette notion de coût, on ajoute un ordre partiel sur les alignements qui caractérise leur complétion :

Définition : soit deux alignements align et align',

align ≥ align' ssi ∃i ∀j si j < i align(j) = align'(j), sinon align'(j) = 0

À partir de cette ordre, on est en mesure de définir la notion d'alignement maximal comme étant un alignement qui n'admet pas d'alignement supérieur (trivial)

Théorème : si align > align' alors coût(align) < coût(align')

Démonstration triviale

3.3. Algorithme de construction des alignements maximaux

À partir de ces définitions, il est possible de concevoir un algorithme efficace qui explore tous les alignements et qui les stocke dans un arbre.

Il part de l'alignement vide qui associe 0 à tous les éléments de la séquence initiale, puis il enrichit progressivement cet alignement. Lorsque plusieurs choix sont possibles, il crée plusieurs fils, ce qui engendre un arbre. Il est facile de montrer que les feuilles terminales de cet arbre sont maximales au sens défini plus haut. La complexité est bien évidemment liée au nombre total d'alignements, ce qui dépend du nombre d'occurrences répétées de chaque blocs.

Pour déterminer l'alignement de coût minimal, on peut construire tout l'arbre des alignements en calculant les coûts des feuilles terminales, et prendre celle de moindre coût. On imagine cependant que la complexité algorithmique est assez importante.

Il suffit pour réduire la complexité de faire appel à une procédure A* en utilisant une fonction heuristique admissible, c'est-à-dire une fonction h qui donne un borne inférieure du coût de la complétion d'un alignement h*. Plus exactement, si l'on considère un alignement partiel *align*, la fonction h aide à déterminer la borne inférieure de tout alignement maximal supérieur à *align* :

∀*align'* si *align'* > *align* et *align'* est un alignement maximal,

alors coût(*align'*) = coût(*align*) + h*(*align*) > coût(*align*) + h(*align*)

Comme le coût d'un alignement est fonction du nombre de blocs qui ne sauraient trouver une correspondance, il est facile de trouver une borne inférieur du coût d'un alignement passant par *align* ; il suffit de calculer le nombre de blocs qui, quoiqu'il advienne, ne pourront pas trouver de correspondance dans la complétion de *align*. Par exemple, si on prend les deux séquences initiales S_1 = B_1, B_2, B_3, B_1, B_4, B_5 et et S_2 = B_2, B_1, B_3, B_5, B_4 et que l'on ait mis en correspondance le premier B_1 de S_1 avec de S_2, nous nous trouvons dans la situation suivante : S_1 = B_1 ↕ B_2, B_3, B_1, B_4, B_5 et S_2 = B_2, B_1 ↕B_3, B_5, B_4, la flèche verticale, ↕, marquant l'état de l'alignement. Il reste à estimer le coût minimal de la complétion de l'alignement de S_1 et S_2, ce qui correspond au coût minimal de l'alignement des séquences de

blocs situés après la flèche, autrement dit, au coût minimal de l'alignement de B_2, B_3, B_1, B_4, B_5 et de B_3, B_5, B_4. Or, celui-ci est au moins égal à $|B_2| + |B_1|$

Bref, le coût minimal de l'alignement de $B_1\updownarrow B_2$, B_3, B_1, B_4, B_5 et B_2, $B_1\updownarrow B_3$, B_5, B_4, est égal au coût de l'alignement de $B_1\updownarrow$ et B_2, $B_1\updownarrow$, c'est-à-dire $|B_2|$, plus le coût minimal estimé de l'alignement de B_2, B_3, B_1, B_4, B_5 et de B_3, B_5, B_4, (c'est-à-dire $|B_2| + |B_1|$) ce qui fait $2.|B_2| + |B_1|$.

4. Extensions de MEDITE

Jusqu'à présent, nous avons surtout évoqué la comparaison de versions de textes littéraires, à l'aide des méthodes de la génétique textuelle. Rappelons que, dans cette éventualité, toutes les transformations qu'un auteur imprime à son texte sont signifiantes. Ainsi, les changements de casse, l'insertion de signes de ponctuation, la position dans la page, les alinéas et les retours chariot, tout doit être mis en évidence. Qui plus est, les modifications internes des mots doivent être soulignées. Ainsi, les corrections orthographiques, par exemple la mise au pluriel ou le changement de temps d'un verbe, sont significatives. Enfin, la réutilisation d'un même radical, supprimé à un endroit du texte puis réinséré plus loin dans le même texte, doit aussi être détectée. Toutes ces considérations nous ont conduit à assimiler les textes à des chaînes de caractères que l'on compare, puis à étudier les insertions, les suppressions, les remplacements et les déplacements de sous chaînes à l'intérieur de ces textes.

Lorsque l'on aborde la comparaison de variantes, qu'il s'agisse de variantes de textes littéraires ou de textes plus ordinaires, comme par exemple d'articles de journaux ou de tracts politiques, les questions se posent d'une façon un peu différente. D'un côté, certains changements sont peu, voire pas du tout significatifs ; ainsi, la mise en majuscules de certaines lettres, l'introduction de signes de ponctuation, de signes diacritiques ou les changements de mise en page importent peu. D'un autre côté, l'orthographe des mots varie parfois sans que cela change fondamentalement la nature du texte. Un logiciel d'alignement doit être capable de faire fi de ces différences pour mettre en correspondance deux variantes écrites à deux époques distinctes par deux scripteurs qui n'obéissaient pas aux mêmes règles typographiques et orthographiques. En résumé, il faut aligner des textes qui présentent des différences notoires, en faisant abstraction de ces différences.

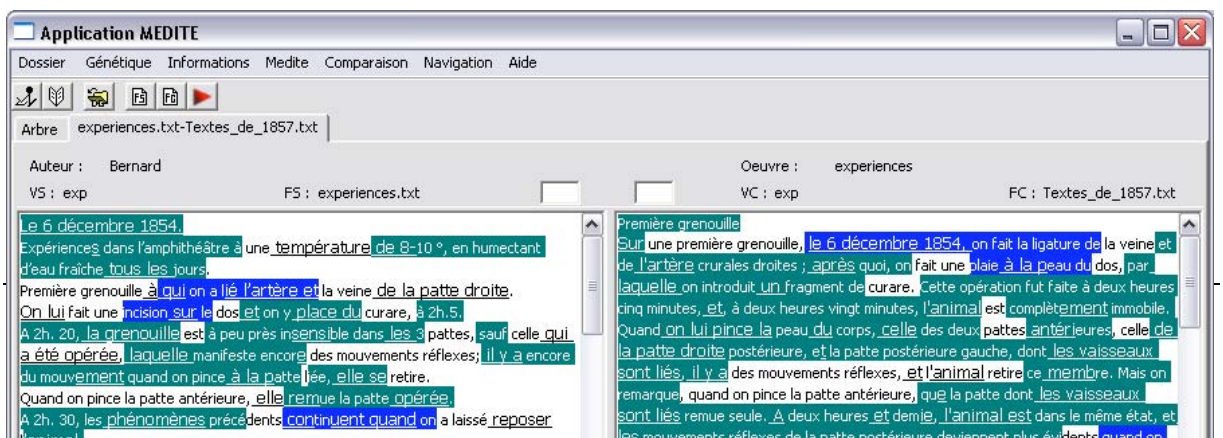
En d'autres termes, là où la génétique textuelle recommande une attention soutenue à la lettre des textes et à leur mise en page, la comparaison de variantes demande de saisir, si ce n'est l'esprit des textes, du moins leur littéralité, pour les mettre en correspondance, même s'ils présentent un certain nombre de différences orthographiques ou typographiques. À cette fin, on doit établir les équivalences admissibles entre caractères et entre mots. L'équivalence entre caractères gomme des distinctions qui n'ont pas lieu d'être, ce qui facilite l'appariement. À titre d'illustration, il se peut que la casse n'ait pas d'importance ou que la présence d'un signe diacritique doive être ignorée, ou encore que les séparations entre mots, qu'il s'agisse de signes de ponctuation ou de blancs, soient toutes considérées comme identiques. L'équivalence entre mots tient compte des différences orthographiques. Ainsi, « *guere* » peut être équivalent à « *gaire* », « *Jhesu* » à « *Jesu* », « *vierges* » à « *virgines* » etc. Nous voulons modifier MEDITE pour prendre en compte ces équivalences au cours de l'alignement des textes.

4.1. Équivalences de caractères

La première modification du programme MEDITE porte sur la prise en compte de l'équivalence de certains caractères. Lorsqu'il s'agit de la casse ou de signes diacritiques ou encore de la cédille, cela ne pose aucun problème ; l'algorithme du programme MEDITE peut être réutilisé presque tel quel. Il suffit de préciser quels sont les caractères considérés comme équivalents. C'est ce qui est déjà implanté dans la version 3.2 du logiciel MEDITE où des options offrent à l'utilisateur la possibilité de s'affranchir de la casse, de signes diacritiques ou de la cédille lors de l'appariement entre textes.

Les choses se corsent lorsque l'on désire prendre en considération l'équivalence des séparateurs. Plus précisément, lorsqu'on introduit des signes de ponctuation, les chaînes de séparation entre les mots doivent toutes être considérées comme équivalentes. Ce qui pose problème n'est bien entendu pas l'équivalence entre signes de ponctuations, blancs et retours chariot, puisque nous nous retrouvons là ramenés au problème précédent, mais c'est la différence de longueur entre les chaînes de séparation. Ainsi, si l'on introduit une virgule ou un point-virgule entre les mots, cela signifie qu'un blanc de séparation est remplacé par soit deux caractères, une virgule est un blanc, soit trois caractères, un blanc insécable, un point-virgule et en blanc.

Sans entrer dans les détails de l'algorithme mis en oeuvre, nous voyons là que cette transformation requiert une modification substantielle de celui-ci. Ceci étant, nous pouvons toujours peu ou prou faire appel à des algorithmes classiques de détection d'homologies sur les chaînes de caractères. Il suffit d'opérer un prétraitement des textes en remplaçant chaque chaîne de séparateurs contigus par un seul caractère séparateur générique. C'est un tel prétraitement qui est mise en œuvre dans la version 3.3 de MEDITE disponible sur le site de MEDITE (http://www-poleia.lip6.fr/~ganascia/medite).



La prise en considération des variantes orthographiques pose des problèmes beaucoup plus délicats. En effet, il faut d'une part connaître les variations possibles et d'autre part tenir compte de l'équivalence des mots qui résulte de ces variations au cours de l'appariement des textes.

4.2.1. Recherche d'homologies sur des chaînes de mots

Pour surmonter la seconde des difficultés, nous avons modifié l'algorithme de recherche d'homologies dans les séquences. Plus exactement, nous avons repris les algorithmes classiques de recherche d'homologies, mais au lieu de rechercher des homologies sur des chaînes de caractères nous recherchons maintenant des homologies sur des chaînes de mots. Le principe au fondement même de l'algorithme de recherche d'homologies n'est pas affecté par cette transformation. En revanche, sa mise en oeuvre informatique est quelque peu changée. Pour comprendre les raisons de ces changements, il convient de rappeler les principes sur lesquels reposent les algorithmes classiques de détection d'homologies : ceux-ci commencent par établir un alphabet à partir du texte, puis ils établissent un arbre des préfixes. Lorsque l'on passe des chaînes de caractères aux chaînes de mots, on peut appliquer exactement le même algorithme si ce n'est que les caractères sont remplacés par les mots. Ainsi, la première étape consiste non plus à construire un alphabet, mais un lexique. Quant aux étapes ultérieures, elles sont identiques, à ceci près que les séquences de caractères sont remplacées par des séquences de mots. Le principe algorithmique n'est pas modifié, mais les dimensions le sont : tandis que le nombre de caractères avoisine la centaine, le nombre de mots qu'il convient de considérer surpasse largement le millier, pour approcher la dizaine de milliers. En conséquence, les structures de données que l'on devra employer pour mettre en oeuvre l'algorithme seront modifiées pour tenir compte de ces nouvelles dimensions. C'est cette nouvelle mise en oeuvre de l'algorithme de détection d'homologies sur les mots que nous sommes en train de programmer.

4.2.2. Équivalence des mots

Étant admis que l'algorithme de détection d'homologies opère sur des chaînes de mots, le problème posé par l'équivalence des mots, qui résulte des variations orthographiques, est identique à celui que posait l'équivalence des caractères sur les chaînes de caractères.

La difficulté ici n'est plus d'ordre algorithmique, mais d'ordre linguistique : il faut être en mesure d'établir une table des variations orthographiques. Or, s'il existe des lexiques et des dictionnaires qui établissent des inventaires de variations orthographiques, il est presque impossible de faire une recension exhaustive de toutes ces variations. Nous ne pouvons donc pas supposer qu'une table des équivalences entre mots soit initialement donnée. À défaut, l'algorithme que nous proposons la construit : nous supposons que les textes adoptent une orthographe régulière, c'est-à-dire que les différentes occurrences d'un mot s'écrivent toutes plus ou moins de la même façon dans un même texte, aux différences flexionnelles près. Quant à la table des équivalences entre mots, elle est établie manuellement, par alignement des lexiques des deux textes en cours de comparaison. Plus exactement, à l'issue de la première étape de construction des lexiques, ceux-ci sont alignés à l'aide d'un certain nombre de règles générales et éventuellement en ayant recours à une intervention des utilisateurs.

4.2.3. Ordre des mots

Le dernier point sur lequel il convient d'insister tient à l'ordre des mots : les variantes ne se restreignent pas à des modifications typographiques ou lexicales ; elles font apparaître des modifications d'ordre syntaxique qui portent sur l'ordre des mots. Il convient donc parfois de faire abstraction de déplacements locaux dans la comparaison des textes. L'algorithme mis en oeuvre dans le programme MEDITE autorise de tels appariements de textes. En effet, ce qui distingue MEDITE des autres aligneurs existants, c'est qu'il est capable de détecter les déplacements. Nous avons vu que, dans le cas de la critique textuelle génétique, cette détection est tout à fait essentielle. Elle rend compte de stratégies de réécriture propres aux auteurs, par exemple d'évitement de la répétition. Lorsque l'on souhaite aligner des textes de la littérature médiévale, cela semble moins pertinent. Toutefois, les déplacements peuvent être mis à profit pour rendre compte de modifications syntaxiques mineures qui affectent l'ordre des mots. Ainsi, on peut distinguer les déplacements à courte distance et des déplacements à longue distance. Les premiers devraient alors être mis en regard de transformations syntaxiques possibles.

5. Conclusion

Les développements actuels de MEDITE, tels qu'ils sont présentés dans cet article, montrent que l'alignement unilingue est loin d'être un problème trivial, ne serait-ce que parce qu'il n'existe pas de critère absolu d'évaluation de la qualité d'un alignement. D'ailleurs, pour l'instant, et quoique le problème semble relativement simple, il ne semble pas qu'il existe d'aligneur unilingue satisfaisant pour le traitement de textes littéraires. La comparaison des versions de Microsoft Word ainsi que des aligneurs comme TUSTEP ou comme MVDViewer ne répondent pas à toutes les exigences requises. Des difficultés d'ordre algorithmique s'ajoutent à des problèmes d'interfaçage.

MEDITE aborde ce problème en choisissant de maximiser le nombre de caractères mis en correspondance. L'algorithme passe par la construction de blocs maximaux disjoints, puis par une optimisation du coût de l'alignement de ces blocs maximaux. Le programme résultant se révèle rapide et efficace en ce sens que les résultats qu'il produit sont pertinents.

Par ailleurs, l'alignement des séquences doit pouvoir faire abstraction de signes diacritiques, de la casse ou de l'introduction de séparateurs (par exemple, de blancs, de retours chariots, etc.). Dès à présent, des options permettent, lorsque le besoin s'en fait sentir, de s'en affranchir au cours de l'alignement.

L'emploi de MEDITE pour répondre aux besoins de la philologie médiévale exige que l'on aligne des textes dont l'orthographe varie. Ceci passe par la constitution d'un lexique et rapproche le problème de l'alignement unilingue de l'alignement multilingue tel qu'il se développe depuis plusieurs années pour les besoins de la traduction automatique. Notons toutefois que les exigences de la philologie sont différentes de celles de la traduction automatique auxquelles doivent se plier les aligneurs multilingues.

Les perspectives présentes portent d'un côté sur la réalisation d'un aligneur unilingue destiné à aider les spécialistes de philologie médiéval et, d'un autre côté, sur le post-traitement des alignements réalisés. Il s'agit, en particulier, dans le cas de la génétique textuelle, d'être en mesure de caractériser syntaxiquement des styles de réécriture, en mettant l'accent sur les catégories syntaxiques le plus affectées par tel ou tel auteur, selon les phases de sa réécriture.

Références

de Biasi P-M. (2000). *La génétique des textes*. Nathan Université.

Cerquiglini, B. (1989). *Éloge de la variante. Histoire critique de la philologie*. Paris, Seuil.

Chedid, A. (1996). La robe noire. In *Les saisons de passage*, Flammarion

Crochemore M. and Rytter W. (1994). Text Algorithms, Approximate pattern matching : 237-251.

Fenoglio I. et Boucheron S. (éd.) (2002). Processus d'écriture et marques linguistiques. *Langages 147*, sept. 2002.

Ganascia J-G., Fenoglio I., Lebrave J-L. (2004 a). EDITE MEDITE : un logiciel de comparaison de versions. A*ctes des 7ième journées d'Analyse statistique de Données Textuelles (JADT 2004) :* 468-478.

Ganascia J-G., Fenoglio I., Lebrave J-L. (2004 b). Manuscrits, genèse et documents numérisés. EDITE : une étude informatisée du travail de l'écrivain, Document numérique, vol. 8/4, 2004 : 91-110.

Ganascia J-G. (2006). EDITE – MEDITE : un passage des versions aux variantes. A*ctes de la XXIVième Conférence Internationale de Linguistique et de Philologie Romane*, Aberystwyth, University of Wales, Royaume Uni, août 2004 (à paraître en 2006).

Grésillon A. (1994). *Éléments de critique génétique*. PUF.

Hay L. (2002). La littérature des écrivains. *Études de critique génétique*, Corti.

Lebrave J-L. (1984). Le traitement automatique des brouillons, in «Programmation et sciences humaines», LISH, CNRS, n°3 : 90-106.

Lebrave J-L. (2000). La nouvelle philologie et l'édition électronique des dossiers génétiques. In Budor D., Perrus C. (eds), *Le texte. Genèse, variantes*, édition. Arzana n° 5, Presses de la Sorbonne nouvelle : 121-151.

Sankoff D., Kruskal J. (1983). Time Warps, String Edits and Macromolecules : The Theory and Practice of Sequence Comparison.

Ukkonen E. (1995). *On-line construction of suffix-trees*. Algorithmica 14 : 249-260.