

# Extension de l'Alignement Multi-critères au Niveau des Mots dans les Documents Médiévaux

Rapport final de projet de semestre

Silvia Quarteroni  
Ecole Polytechnique Fédérale de Lausanne  
silvia.quarteroni@epfl.ch

27 juin 2003

## **Résumé**

Ce document présente une méthode d'alignement textuel, i.e. une technique qui permet de mettre en correspondance dans deux ou plusieurs textes les parties homologues. Une telle méthode a été appliquée à un corpus de textes en français médiéval, sur lequel un alignement au niveau des paragraphes avait déjà été accompli sur la base de critères lexicaux, linguistiques et structurels. L'objectif de la méthode exposée est de mettre en correspondance les mots situés à l'intérieur des paragraphes déjà alignés.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Définition de l’alignement . . . . .	4
1.1.1	Applications de l’alignement . . . . .	4
1.2	L’alignement au niveau des mots . . . . .	4
1.2.1	Contexte du projet . . . . .	5
1.2.2	Applications et implémentation de l’alignement des mots . . . . .	5
<b>2</b>	<b>Prétraitement des textes</b>	<b>6</b>
2.1	Algorithme de traitement initial . . . . .	6
2.1.1	Implémentation de l’alignement initial . . . . .	6
2.2	Résultats du prétraitement . . . . .	7
<b>3</b>	<b>Modèle de l’Alignement</b>	<b>10</b>
3.1	Le modèle probabiliste . . . . .	10
3.2	Le rapport de Dice . . . . .	10
3.3	La probabilité de traduction . . . . .	11
3.4	La probabilité de décalage . . . . .	11
3.4.1	Modèle final . . . . .	12
3.5	Fonction de similitude pour l’alignement . . . . .	12
<b>4</b>	<b>Recherche de l’alignement optimal</b>	<b>13</b>
4.1	La programmation dynamique . . . . .	13
4.2	Parcours du chemin optimal . . . . .	14
<b>5</b>	<b>Post-traitement et évaluation des résultats</b>	<b>17</b>
5.1	Filtrages supplémentaires et adaptation . . . . .	17
5.2	Evaluation des résultats sur les documents . . . . .	20
5.2.1	Alignement prose-vers . . . . .	20
5.2.2	Alignement vers-vers . . . . .	22
5.3	Extensions et améliorations possibles . . . . .	25
<b>6</b>	<b>Implémentation</b>	<b>27</b>
6.1	Lecture des fichiers TMX . . . . .	27
6.2	Alignement en temps réel de deux documents . . . . .	27
6.3	Liste des classes Java ajoutées au paquetage org.epfl.medievalz.utils . . . . .	28

<b>A</b>	<b>Extraits des fichiers TMX utilisés</b>	<b>33</b>
A.1	Document vers-vers : paris871-paris373.tmx . . . . .	33
A.2	Document prose-vers : geneve-paris137.tmx . . . . .	35
<b>B</b>	<b>Extrait de la base de données <i>MedievLex</i></b>	<b>36</b>

# Chapitre 1

## Introduction

### 1.1 Définition de l'alignement

Ce document se propose d'illustrer une approche possible et relativement rapide pour le processus d'alignement textuel au niveau des mots. Pour clarifier le sujet traité, il faut donc d'abord préciser le concept d'*alignement*.

D'un point de vue formel, l'*alignement* est le processus qui met en correspondance dans deux ou plusieurs textes les parties homologues qui sont des traductions directes. Ces parties peuvent être des paragraphes mais aussi des phrases, des expressions ou des simples mots.

#### 1.1.1 Applications de l'alignement

L'alignement trouve bien sûr un intérêt d'application dans le cas de textes que l'on suppose ayant un contenu très proche, par exemple un document en français et sa traduction en anglais, ou encore deux versions légèrement différentes du même texte en latin. Par ailleurs, des méthodes d'alignement ont été appliquées également à des textes dont les langues sont très distantes, comme l'hébreu et l'anglais <sup>1</sup>.

La technique de l'alignement trouve son application principale dans le contexte de la traduction automatique, mais aussi en tant que ressource à disposition du traducteur humain ; les applications ne se limitent cependant pas exclusivement à ce domaine. Il est clair que l'intérêt de l'informatique vers le traitement du langage est très fort, car cela crée un pont entre la communication humaine et la communication homme-machine.

### 1.2 L'alignement au niveau des mots

Ce document traite en particulier de l'alignement dans les documents médiévaux : nous disposons d'un corpus de textes qui sont tous des adaptations en prose et en vers des *Métamorphoses* d'Ovide en français ancien (entre les siècles XII et XV). Nous avons donc à considérer des textes assez semblables, du fait qu'ils sont écrits dans la même langue, bien qu'avec des variations lexicales dues à l'évolution linguistique. Les algorithmes de traitement des documents devront donc s'adapter à cette similitude,

---

<sup>1</sup>consultez à ce sujet Choueka [2].

et pourront souvent se passer de traitements complexes tels que la lemmatisation, nécessaires au contraire dans le cas de langues très distantes telles que les langues sémitiques et les langues de matrice anglo-saxonnes.

### 1.2.1 Contexte du projet

Le point de départ de ce projet est l'alignement au niveau des paragraphes, déjà accompli dans le cadre du projet MEDIEVAL<sup>2</sup>. Cet alignement est basé sur plusieurs critères, notamment structurels (i.e. inhérent à la conformation des textes, au schéma rhétorique) et linguistiques (lexicaux, syntaxiques et sémantiques). La fiabilité d'un tel alignement a été estimée à 75% environ.

Le but du projet est d'aligner les mots correspondants à l'intérieur des paragraphes déjà mis en correspondance. Pour effectuer ce travail un prétraitement des documents a été mis en place, ensuite l'alignement proprement dit a été appliqué selon divers critères ayant pour but de maximiser la valeur d'une fonction de similitude *ad hoc*.

### 1.2.2 Applications et implémentation de l'alignement des mots

L'application des résultats d'un algorithme d'alignement mot par mot pourrait être d'un côté le support à l'étude philologique de la langue française, mais aussi l'aide à l'intégration de nouvelles entrées dans une base de données lexicale telle que *Mediev-Lex*. Puisque les variations lexicales demandent un effort de couverture considérable et diminuent la performances lors de la consultation, on pourrait imaginer un système qui, une fois déterminée la correspondance entre les termes *mais* et *maiz*, insère le mot *maiz* dans la base de données à côté de *mais*.

Une application directe de l'algorithme développé est notamment dans le logiciel *Medievalz*, permettant de visualiser l'alignement entre les paragraphes des documents médiévaux ; un module a été développé, permettant le calcul en temps réel de l'alignement mot par mot du segment de texte considéré. Nous traiterons ce sujet au chapitre *Implémentation*.

---

<sup>2</sup>cf. à ce sujet Ghorbel [1].

## Chapitre 2

# Prétraitement des textes

Pour commencer, un filtrage a été effectué sur les textes de manière à ne considérer que les mots de longueur supérieure ou égale à trois lettres : ceci permet d'éliminer une partie consistante des termes tels que les articles, qui apportent une contribution très modeste à la compréhension mais pourraient ralentir considérablement le traitement des écrits.

### 2.1 Algorithme de traitement initial

Ensuite, un alignement initial approximatif a été effectué. Cet alignement sert de référence pour l'algorithme proprement dit, le but étant de fournir un parallélisme grossier des deux segments considérés.

Dans le cas d'un alignement complexe tel que celui décrit par Choueka [2], il serait nécessaire d'implémenter un algorithme itératif tel que l'*Expectation-Maximisation* pour garantir une discrète fiabilité ; dans le cas présent nous pouvons constater que la similitude des textes à examiner est suffisante pour mettre en place une technique bien moins complexe et donc bien moins coûteuse en temps de calcul.

La stratégie consiste à aligner les mots qui se trouvent aux mêmes positions dans les deux textes : si le texte source  $S$  et le texte destination  $D$  ont le même nombre de mots, alors le  $i$ -ème mot de  $S$  sera aligné avec le  $i$ -ème mot de  $D$  pour tout indice  $i$ . Cependant, puisque dans la plupart des cas les longueurs des textes à comparer diffèrent, et ceci d'autant plus si il s'agit d'un alignement vers-prose, il faut garantir que l'alignement initial ait une distribution la plus homogène possible. Ces considérations ont porté à la proposition de l'algorithme de type tabulaire décrit ci-dessous. Les alignements possibles sont représentés comme entrées non nulles dans une matrice, `initialAlignment`, dont les lignes correspondent aux entrées du premier texte, et les colonnes aux entrées du deuxième texte.

#### 2.1.1 Implémentation de l'alignement initial

L'algorithme `alignDiagonally` prend en paramètre la position en haut à gauche (i.e.  $\langle 0, 0 \rangle$ ) et celle en bas à droite (i.e.  $\langle f, s \rangle$ , où  $f$  est la longueur du premier texte et  $s$  la longueur du deuxième) de la matrice `initialAlignment` et appelle la méthode récursive `alignDynamically`. Celle-ci attribue un coefficient non nul aux alignements correspondant à ces deux positions et à la position centrale

### Algorithme alignDiagonally

```
public double[][] alignDiagonally(){
    alignDynamically(0,0,f,s);
    return initialAlignment;
}
```

FIG. 2.1 – Algorithme alignDiagonally.

### Algorithme alignDynamically

```
public void alignDynamically(int a, int b, int x , int y){
    int milieuX = new Double((x-a)/2).intValue();
    int milieuY = new Double((y-b)/2).intValue();
    initialAlignment[x][y] = 1.0;
    initialAlignment[a][b] = 1.0;

    if(milieuX <= 0 || milieuY <= 0 || a ==x || b==y){
        return;
    } else {
        initialAlignment[milieuX][milieuY] = 1.0;
        alignDynamically(a+1,b+1,x-1,y-1);
    }
}
```

FIG. 2.2 – Algorithme alignDynamically.

(i.e.  $\langle milieuX, milieuY \rangle$ ); à l'étape suivante, il prendra en paramètre la section du tableau qui commence en  $\langle a+1, b+1 \rangle$  et termine en  $\langle i-1, j-1 \rangle$ , et ainsi de suite jusqu'à ce que les indices verticaux ou horizontaux coïncident.

## 2.2 Résultats du prétraitement

Il est intéressant de constater que souvent cet algorithme produit des résultats presque parfaits lorsqu'il est appliqué à deux segments en vers; deux cas sont reproduits ci-dessous pour illustrer les résultats de ce premier traitement.

### Résultats sur un cas vers-vers : documents *Paris-871* et *Paris-373*.

```
Homs doit penser a sauver s'ame
452 Qui du corps est maistrasse et dame,
453 Et a deservir Paradis.
```

```
Homs doit penser a sauver s'ame
452 Qui du corps est maistresse et dame,
```

453 Et a deservir Paradiz.

INITIAL ALIGNMENT :

Aligned Homs with Homs  
Aligned doit with doit  
Aligned penser with penser  
Aligned sauver with sauver  
Aligned ame with ame  
Aligned Qui with Qui  
Aligned corps with corps  
Aligned est with est  
Aligned maistresse with maistresse  
Aligned dame with dame  
Aligned deservir with deservir  
Aligned Paradiz with Paradis

INITIAL ALIGNMENT MATRIX :

```
[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 ]  
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 ]
```

### Résultats sur un cas prose-vers : documents *Genève* et *Paris-137*.

homme doit penser a sauver son ame  
qui du corps est dame et maistresse #

451 Homs doit penser a sauver s'ame,  
452 Qui du corps est maistresse et dame,  
453 Et a deservir Paradis.

INITIAL ALIGNMENT :

Aligned Homs with homme  
Aligned doit with doit  
Aligned penser with penser  
Aligned sauver with sauver

Aligned ame with son  
 Aligned Qui with ame  
 Aligned est with qui  
 Aligned maistresse with corps  
 Aligned dame with est  
 Aligned deservir with dame  
 Aligned Paradis with maistresse

INITIAL ALIGNMENT MATRIX :

```

[ 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 ]
[ 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 ]

```

Nous pouvons constater que cette méthode partielle présente des lacunes évidentes : dans le deuxième exemple, les mots *son* et *ame* sont alignés seulement parce-qu'ils se trouvent aux mêmes endroits dans les deux textes, sans compter le décalage des alignements des derniers termes ; ceci justifie évidemment un traitement successif bien plus rigoureux , qui pouvait paraître superflu dans le premier exemple.

## Chapitre 3

# Modèle de l'Alignement

Dans ce chapitre nous examinerons les détails du modèle développé pour exprimer l'alignement de deux textes, dorénavant appelés source ( $S$ ) et destination ( $D$ ).

Nous devons tout d'abord préciser que l'alignement en question n'est pas orienté : ceci signifie que le choix du document qui joue le rôle de source et de celui qui joue le rôle de destination n'a pas d'influence sur les résultats finaux du processus. Cette interprétation de l'alignement se distingue de celle adoptée par d'autres (e.g. Choueka [2]). La raison d'une telle décision est la compatibilité avec le contexte de l'implémentation, qui exige une symétrie entre source et destination, ce qui sera mieux précisé plus tard. La définition mathématique d'alignement est la suivante :

**Définition.** Un alignement  $a$  est un ensemble de couples  $\langle i, j \rangle$  où  $i$  représente le  $i$ -ème mot du texte source (dorénavant  $S_i$ ) et  $j$  le  $j$ -ème mot du texte destination (dorénavant  $D_j$ ).

### 3.1 Le modèle probabiliste

Il s'agit donc de spécifier quels couples de la forme  $\langle i, j \rangle$  parmi toutes les connexions possibles sont censés faire partie de l'alignement  $a$  en question. Pour cela, un modèle probabiliste a été introduit. Le but étant de maximiser la vraisemblance de traduction de  $S$  en  $D$ , nous sommes intéressés par le calcul de la probabilité que  $D$  soit la traduction de  $S$  sur la base de tous les alignements possibles. Nous posons donc :

$$P(D | S) = \sum_a P(D, a | S) \quad (3.1)$$

où  $a$  couvre tous les alignements possibles.

La probabilité d'une connexions dans  $a$  dépend de trois paramètres, le premier lexical, le deuxième sémantique, le dernier structurel. Ils sont exposés dans les sections 3.2 à 3.4.

### 3.2 Le rapport de Dice

Le rapport de Dice entre deux mots est une mesure de leur similarité lexicale : il dépend du nombre de monogrammes, bigrammes ou trigrammes communs aux deux

termes et de la somme de leur longueur. l'ordre de la mesure de Dice dépend de la taille des n-grammes communs souhaitées.

Dans le cas de flux de caractères de grandes dimensions où l'on veut repérer des régions correspondantes de taille élevée il est habituel d'utiliser le rapport de Dice d'ordre 1.

Dans le cas discuté dans le document présent, le choix de l'ordre 2 semble plus approprié puisque la comparaison se fait au niveau des mots et une mesure moins approximative est nécessaire. La valeur retournée par ce calcul est un nombre réel compris entre 0 et 1 :

$$Dice_2(i, j) = \frac{2 * (nb\_bigrammes\_identiques(S_i, D_j))}{|S_i| + |D_j|} \quad (3.2)$$

### 3.3 La probabilité de traduction

La probabilité de traduction est une mesure de similitude sémantique entre deux mots : elle représente la probabilité que le mot  $j$  soit une traduction possible du mot  $i$  et est calculée à partir d'une base de données.

Elle s'exprime :

$$P_T(D_j | S_i) = similitude\_max(D_j, S_i) \quad (3.3)$$

Pour ce projet, la base de données *MedievLex* a été employée. Les entrées sont structurées par groupes selon la relation de proximité qu'elles présentent : il y a d'abord la **morphologie**, ensuite la **catégorie**, enfin la **famille**.

Sur la base de la similitude maximale présentée par les deux mots, le barème suivant a été adopté :

Niveau max. similitude	probabilité	exemple
identité	1.0	<i>bon et bon</i>
morphologie	0.9	<i>bon et bonne</i>
catégorie	0.75	<i>bon et bien</i>
aucune	0.25	<i>bon et mais</i>
inconnu	0.5	<i>bon et maiz</i>

TAB. 3.1 – Probabilités de traduction

### 3.4 La probabilité de décalage

La probabilité de décalage d'un alignement est un critère d'évaluation structurel : on assume que la probabilité que deux mots soient des traductions est bien supérieure s'ils sont situés aux mêmes positions par rapport à leurs textes que s'ils se trouvent en des positions très disparates.

Nous définissons donc la fonction de probabilité  $P_O(k(i,j))$ , où  $k(i,j)$  est une fonction de  $i$  et de  $j$  qui retourne la différence (en valeur absolue) entre la position  $i$  de  $S$  alignée avec la position  $j$  de  $D$  selon l'alignement  $a$  et la position dans  $S$  alignée avec  $j$  selon l'alignement initial  $I(j)$ .

Le barème fixé pour les valeurs de la fonction est le suivant :

Décalage en mots	probabilité
inférieur à 3	0.75
entre 4 et 5	0.5
supérieur à 5	0.15

TAB. 3.2 – Probabilités de décalage

### 3.4.1 Modèle final

Quelques modifications ont été effectuées pour améliorer la performance de l’alignement ; en particulier il faut constater qu’il serait plus efficace de considérer lors d’un alignement entre deux positions  $i$  et  $j$  non plus l’alignement initial, mais la connection effectuée pour la position dans  $D$  précédente à  $j$ . La position attendue du mot source dans une connection ne dépend donc plus de  $I(j)$ , mais de la connection précédente selon l’alignement  $a$  ; nous calculons donc :

$$a'(j) = a(j_{PREC}) + (j - j_{PREC}) * \frac{N_S}{N_D} \quad (3.4)$$

La nouvelle définition de  $k(i,j)$  est donc :

$$k(i, j) = i - a'(j)$$

Le calcul de  $a'(j)$  s’effectue avec une fenêtre  $f$ , c’est à dire une distance prédéfinie telle que si  $a'(j)$  est supérieur à  $f$ , il est remis à la valeur  $I(j)$ . Ceci empêche de considérer des alignements trop distants de la diagonale principale de la matrice `initialAlignment`.

## 3.5 Fonction de similitude pour l’alignement

Désormais, nous pouvons exprimer la similitude entre le terme  $S_i$  et le terme  $D_j$  comme une combinaison linéaire des trois fonctions définies ci-dessus, chacune ayant même pondération : nous définissons donc le poids, ou similitude globale, d’un alignement comme :

$$W'(< i, j >) = Dice_2(i, j) * P_T(D_j | S_i) * P_O(i - a'(j)) \quad (3.5)$$

## Chapitre 4

# Recherche de l'alignement optimal

Grâce aux développements précédents, nous disposons d'une fonction qui permet d'évaluer le poids d'un alignement potentiel entre deux mots : il s'agit maintenant de sélectionner, parmi toutes les connections possibles, celle qui maximise la pondération totale.

L'alignement le plus probable est donc celui qui maximise le poids suivant :

$$W(a) = \prod_{\langle i,j \rangle \in a} W'(\langle i,j \rangle) \quad (4.1)$$

### 4.1 La programmation dynamique

L'ensemble des connections étant représenté sous la forme d'une matrice, la recherche de l'alignement optimal consiste en un parcours de la matrice qui maximise la similitude globale. Ce parcours est calculé selon le principe d'optimalité locale de Bellman<sup>1</sup> :

Dans une séquence optimale de décisions, quelle que soit la première décision prise, les décisions subséquentes forment une sous-séquence optimale, compte tenu des résultats de la première décision.

Dans le cas présent, à chaque pas, la valeur associée à la connection  $\langle i,j \rangle$  dépend du maximum entre les valeurs des connections  $\langle i-1,j-1 \rangle$ ,  $\langle i,j-1 \rangle$ ,  $\langle i-1,j \rangle$ .

Plus précisément, un *seuil* est fixé à un tiers de la valeur maximale possible pour  $W'(\langle i,j \rangle)$ . Dans une matrice  $T$  les valeurs suivantes sont conservées :

$$T[i][j] = \begin{cases} T[i-1][j-1] + W'[i][j] & \text{si } W'(\langle i,j \rangle) > \text{seuil} \\ \max(T[i-1][j], T[i][j-1]) + \text{seuil} & \text{sinon} \end{cases}$$

---

<sup>1</sup> Voir Bellman [5].

### Algorithme computePath.

```
public static void computePath(double seuil){
    t = new double[f+1][s+1];
    val = new int[f+1][s+1];
    for (int i = 1; i < f+1; i++){
        for (int j = 1; j < s+1; j++) {
            if(wPrimeMatrix[i][j] >= seuil ||
                (wPrimeMatrix[i][j] >= max(t[i-1][j],t[i][j-1]))) {
                t[i][j] = wPrimeMatrix[i][j]+ t[i-1][j-1] ;
                val[i][j] = 1;
            } else {
                t[i][j] = seuil + max(t[i-1][j], t[i][j-1]);
                val[i][j] = argmax(t[i-1][j], t[i][j-1]);
            }
        }
    }
}
```

FIG. 4.1 – Algorithme computePath.

Le chemin est encodé dans une deuxième matrice *Val*, qui présente les valeurs suivantes :

$$Val[i][j] = \begin{cases} 1 & \text{si la connection } [i-1][j-1] \text{ a été choisie} \\ 2 & \text{si la connection } [i-1][j] \text{ a été choisie} \\ 3 & \text{si la connection } [i][j-1] \text{ a été choisie} \end{cases}$$

La programmation dynamique est une technique utilisée dans de nombreux domaines, notamment dans le traitement du langage naturel pour le calcul de la distance lexicographique entre mots ou flux de caractères. En effet, le problème de maximisation de la similitude globale revient au problème de minimisation de la distance (ou dissimilarité) globale plus souvent cité dans la littérature<sup>2</sup>. L'algorithme récursif *computePath* reporté en pseudo-code Java utilise cette technique.

Encore une fois, l'algorithme tend à privilégier les positions situées sur la diagonale de la matrice ; les positions en haut et à gauche d'une case ne sont prises en considération que si le poids de l'alignement dans ce même cas n'excède pas le seuil fixé.

## 4.2 Parcours du chemin optimal

Il s'agit maintenant de parcourir le chemin construit de manière à conserver la suite de connections optimales : pour cela, il faut parcourir la matrice *Val* depuis l'extrémité en bas à droite et remonter en suivant les pas encodés dans ses entrées ; l'algorithme *checkValues* effectue cette tâche de manière récursive et écrit l'alignement dans la matrice *alignment*.

---

<sup>2</sup>Voir à ce sujet Belaid [4].

**Algorithme checkValues.**

```
public void checkValues(int i, int j){
    int[][] values = shiftBackMatrix(val);
    if(i==0 || j==0){
        return;
    }else{
        if(values[i][j] ==1 ){
            alignment[i-1][j-1]= 1;
            checkValues(i-1,j-1);
        }else{
            if(values[i][j] ==2){
                alignment[i-1][j]= 1;
                checkValues(i-1,j);
            }else{
                if(values[i][j] ==3){
                    alignment[i][j-1]=1;
                    checkValues(i, j-1);
                }
            }
        }
    }
}
```

FIG. 4.2 – Algorithme checkValues.

**Filtrage.** Un dernier filtrage s'avère nécessaire pour éliminer de l'ensemble de connexions trouvées celles qui ne présentent pas une similitude satisfaisante : un seuil de 0.01 a été fixé.

## Chapitre 5

# Post-traitement et évaluation des résultats

Ce chapitre décrit dans un premier temps les derniers traitements nécessaires pour adapter le comportement de l'algorithme exposé précédemment aux exigences de l'utilisateur, puis les prestations de la méthode développée sur deux échantillons, le premier prose-vers, le deuxième vers-vers. Une dernière section discutera les éventuelles extensions futures de ce travail ainsi que des améliorations possibles.

### 5.1 Filtrages supplémentaires et adaptation

Il faut considérer que l'algorithme dynamique exposé dans le chapitre 4 ne produit pas un alignement bijectif : par cela il faut comprendre que plusieurs mots de  $S$  peuvent être alignés avec le même mot  $D_j$ , et réciproquement, plusieurs mots de  $D$  peuvent se trouver alignés avec le même mot  $S_i$ .

Il s'avère donc nécessaire de filtrer les alignements ainsi obtenus de manière appropriée aux exigences de l'utilisateur ; pour cela trois fonctions ont été implémentées, permettant de :

- visualiser tous les mots alignés avec chacune des entrées du texte source,
- visualiser tous les mots alignés avec chacune des entrées du texte destination,
- visualiser seulement les alignements tels que pour chaque mot de la source il y ait un seul mot de la destination et pour chaque mot de la destination il y ait un seul mot de la source.

La dernière solution est bien évidemment la plus intuitive, malgré la pauvreté qu'elle puisse comporter dans certains cas ; c'est aussi la solution à retenir pour toute application qui veuille traiter le texte source et le texte destination indifféremment, comme celle développée lors de l'implémentation dans *Medievalz*.

Les trois algorithmes de filtrage sont reportés ci-prés.

### Algorithme computeInjectiveMatrix.

```
/**
 * Modifie la matrice des poids pour que les seuls
 * elements non nuls soient les poids maximaux de
 * chaque colonne (i.e. pour tout j dans s il y a
 * le i dans f tel que  $w'(i,j)$  soit maximal pour tout i.)
 */
public int[][] computeInjectiveMatrix(){
    int [][] ima = new int[f][s];
    for(int j = 0; j < s; j++){
        double currentMax = 0.0;
        int currentI = 0;
        for(int i = 0; i < f; i++){
            if(getDynamicAlignment()[i][j] == 1){
                if(currentMax < wPrimeMatrix[i][j]){
                    currentMax = wPrimeMatrix[i][j];
                    currentI = i;
                }
            }
            ima[currentI][j] = 1;
        }
    }
    return ima;
}
```

FIG. 5.1 – Algorithme computeInjectiveMatrix

### Algorithme computeSurjectiveMatrix.

```
/* Dans cette matrice, les alignements sont ceux
 * tels que pour tout i, seul le meilleur j de
 * dynamicAlignment est retenu.
 */
public int[][] computeSurjectiveMatrix(){
    int [][] sma = new int[f][s];
    for(int i = 0; i < f; i++){
        double currentMax = 0.0;
        int currentJ = 0;
        for(int j = 0; j < s; j++){
            if(getDynamicAlignment()[i][j] == 1){
                if(currentMax < wPrimeMatrix[i][j]){
                    currentMax = wPrimeMatrix[i][j];
                    currentJ = j;
                }
                sma[i][currentJ] = 1;
            }
        }
    }
    return sma;
}
```

FIG. 5.2 – Algorithme computeSurjectiveMatrix

### Algorithme computeBijectiveMatrix.

```
public int[][] computeBijectiveMatrix(){
    int[][] toReturn = new int[f][s];
    int[][] injective = computeInjectiveMatrix();
    for(int i = 0; i < f; i++){
        double currentMax = 0.0;
        int currentJ = 0;
        for(int j = 0; j < s; j++){
            if(injective[i][j] == 1){
                if(currentMax < wPrimeMatrix[i][j]){
                    currentMax = wPrimeMatrix[i][j];
                    currentJ = j;
                }
                toReturn[i][currentJ] = 1;
            }
        }
    }
    return toReturn;
}
```

FIG. 5.3 – Algorithme computeBijectiveMatrix

## 5.2 Evaluation des résultats sur les documents

L'algorithme d'alignement modélisé au chapitre précédent a été testé sur deux fichiers : le premier présente l'alignement grossier de 115 paragraphes en prose avec 115 paragraphes en vers, le deuxième présente 44 alignements grossiers vers-vers.

**Le format TMX.** Les fichiers sont structurés selon un format XML spécialement conçu pour mettre en évidence le parallélisme entre deux textes : il s'agit du format TMX [3]. Chaque document y est subdivisé en *translation units*,  $\langle tu \rangle$ , c'est à dire unités de traduction, qui contiennent un couple de segments textuels mis en relation *segments*,  $\langle seg \rangle$ . C'est donc sur ces segments que le processus a été appliqué.

### 5.2.1 Alignement prose-vers

Dans le cas prose-vers, le fichier `geneve-paris137.tmx` a été adopté. Les performances de l'algorithme sont assez satisfaisantes dans la plupart des cas, malgré le fait que dans certaines unités de traductions seulement une petite partie des alignements possibles dépasse le seuil jugé minimal pour l'acceptation. Les figures suivantes présentent les résultats d'applications dans trois cas, le premier jugé satisfaisant, le deuxième assez satisfaisant, le dernier peu satisfaisant.

#### Segment 115.

```
homme doit penser a sauver son ame qui du corps
est dame et maistresse #
```

```
451 Homs doit penser a sauver s'ame,
452 Qui du corps est maistresse et dame,
453 Et a deservir Paradis.
```

AFTER DYNAMIC ALGORITHM :

```
Offset matrix computed.
Translation matrix computed.
```

VAL MATRIX :

```
[ 1 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 1 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 1 3 3 3 3 3 3 3 3 3 ]
[ 2 2 2 1 3 3 3 3 3 3 3 3 ]
[ 2 2 2 2 3 3 3 3 3 3 3 3 ]
[ 2 2 2 2 1 3 3 3 3 3 3 3 ]
[ 2 2 2 2 2 1 3 3 3 3 3 3 ]
[ 2 2 2 2 2 2 1 3 3 3 3 3 ]
[ 2 2 2 2 2 2 2 1 3 3 3 3 ]
[ 2 2 2 2 2 2 2 2 3 1 3 3 ]
[ 2 2 2 2 2 2 2 2 1 3 3 3 ]
```

FINAL FILTERED ALIGNMENT :

Aligned Homs with homme  
Aligned doit with doit  
Aligned penser with penser  
Aligned sauver with sauver  
Aligned ame with ame  
Aligned Qui with qui  
Aligned corps with corps  
Aligned est with est  
Aligned maistresse with maistresse

Dans ce cas l'alignement résulte satisfaisant, aussi grâce au fait que les segments source et destination sont très similaires.

### **Segment 90.**

+Beaulx angeles estoient  
mais maintenant sont laiz et horribles dyables #

367 Angres estoient, or sont dyables  
368 Orrible et let, mal et doubtables.

VAL MATRIX :

```
[ 1 1 1 1 1 1 1 1 ]  
[ 1 3 3 3 3 3 3 3 ]  
[ 2 1 3 3 3 3 3 3 ]  
[ 2 2 3 3 3 3 3 3 ]  
[ 2 2 3 3 3 3 3 3 ]  
[ 2 2 1 3 3 3 3 3 ]  
[ 2 2 2 3 3 3 3 3 ]  
[ 2 2 2 3 1 2 3 3 ]  
[ 2 2 2 1 3 3 3 3 ]
```

FINAL FILTERED ALIGNMENT :

Aligned Angres with angeles  
Aligned estoient with estoient  
Aligned sont with sont  
Aligned dyables with dyables

Ces résultat est assez satisfaisant compte tenu la différence lexicale des deux segments ; nous remarquons tout de même que l'alignement <Horrible, Orrible> n'a pas été repéré.

### **Segment 1.**

Toutes escriptures soient bonnes et mauvaises  
sont pour nostre prouffit et doctrine faittes #

@ 1 Se l'escripture ne me ment,[5a]

```

2 Tout est pour nostre enseignement
3 Quant qu'il a es livres escript,
4 Soient bien ou mal li escript.

```

```

AFTER DYNAMIC ALGORITHM :
Offset matrix computed.
Translation matrix computed.

```

```

VAL MATRIX :

```

```

[ 1 3 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 1 2 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 3 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 1 3 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 2 1 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 2 2 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 2 2 3 3 3 3 3 3 3 3 ]
[ 2 2 3 3 2 2 3 3 3 3 3 3 3 3 ]

```

```

FINAL FILTERED ALIGNMENT :

```

```

Aligned ment with soient
Aligned pour with pour
Aligned nostre with nostre
Aligned livres with faittes
Aligned escript with faittes

```

Dans ce cas la diversité des segments de départ se traduit en un résultat peu fiable.

## 5.2.2 Alignement vers-vers

Pour le cas d'application où les deux documents se présentent en vers, le fichier `paris871-paris373.tmx` a été analysé. Généralement les segments analysés montrent des très bons résultats déjà en phase de prétraitement, qui se confirment plus tard avec l'application de la méthode dynamique. Les figures suivantes présentent trois résultats.

### Segment 2.

```

Homs doit penser a sauver s'ame
452 Qui du corps est maistrasse et dame,
453 Et a deservir Paradis.

```

```

Homs doit penser a sauver s'ame
452 Qui du corps est maistresse et dame,
453 Et a deservir Paradiz.

```

AFTER DYNAMIC ALGORITHM :  
Offset matrix computed.  
Translation matrix computed.

VAL MATRIX:

```
[ 1 3 3 3 3 3 3 3 3 3 3 3 ]  
[ 2 1 3 3 3 3 3 3 3 3 3 ]  
[ 2 2 1 3 3 3 3 3 3 3 3 ]  
[ 2 2 2 1 3 3 3 3 3 3 3 ]  
[ 2 2 2 2 1 3 3 3 3 3 3 ]  
[ 2 2 2 2 2 1 3 3 3 3 3 ]  
[ 2 2 2 2 2 2 1 3 3 3 3 ]  
[ 2 2 2 2 2 2 2 1 3 3 3 ]  
[ 2 2 2 2 2 2 2 2 1 2 3 3 ]  
[ 2 2 2 2 2 2 2 2 3 1 3 3 ]  
[ 2 2 2 2 2 2 2 2 3 2 1 3 ]  
[ 2 2 2 2 2 2 2 2 3 2 2 1 ]
```

FINAL FILTERED ALIGNMENT :

Aligned Homs with Homs  
Aligned doit with doit  
Aligned penser with penser  
Aligned sauver with sauver  
Aligned ame with ame  
Aligned Qui with Qui  
Aligned corps with corps  
Aligned est with est  
Aligned maistresse with maistraille  
Aligned dame with dame  
Aligned deservir with deservir  
Aligned Paradiz with Paradis

## Segment 22.

365 aucuns qui s'orgueillirent  
366 Ou tenebreus enfer cheïrent -  
367 Angelz estoient,

365 aucuns qui s'orgueillirent  
366 Ou tenebreux enfer cheïrent  
367 Anges estoient,

AFTER DYNAMIC ALGORITHM :

Offset matrix computed.  
Translation matrix computed.

VAL MATRIX :

```
[ 1 3 3 3 3 3 3 3 ]
[ 2 1 3 3 3 3 3 3 ]
[ 2 2 1 3 3 3 3 3 ]
[ 2 2 2 1 3 3 3 3 ]
[ 2 2 2 2 1 3 3 3 ]
[ 2 2 2 2 2 1 2 3 ]
[ 2 2 2 2 2 3 1 2 ]
[ 2 2 2 2 2 3 3 1 ]
```

FINAL FILTERED ALIGNMENT :

```
Aligned aucuns with aucuns
Aligned qui with qui
Aligned orgueillirent with orgueillirent
Aligned tenebreux with tenebreus
Aligned enfer with enfer
Aligned cheïrent with cheïrent
Aligned Anges with Angelz
Aligned estoient with estoient
```

### Segment 32.

```
esté 292 Monstrerent lors leur clarté.
293 Ly soulaus et la lune luirent
294 Qui lors primes apparurent. 295
```

```
esté 292 Monstrerent lors leur clarté.
293 Li solaux et la lune lurent
294 Qui lores primes apparurent. 295
```

AFTER DYNAMIC ALGORITHM :

```
Offset matrix computed.
Translation matrix computed.
```

VAL MATRIX :

```
[ 1 3 3 3 3 3 3 3 3 3 3 ]
[ 2 1 3 3 3 3 3 3 3 3 3 ]
[ 2 2 1 3 3 3 3 3 3 3 3 ]
[ 2 2 2 1 3 3 3 3 3 3 3 ]
[ 2 2 2 2 1 2 3 3 3 3 3 ]
[ 2 2 2 2 3 1 2 3 3 3 3 ]
[ 2 2 2 2 3 3 1 2 3 3 3 ]
[ 2 2 2 2 3 3 3 1 2 3 3 ]
[ 2 2 2 2 3 3 3 3 1 3 3 ]
```

```
[ 2 2 2 2 3 3 3 3 2 1 2 3 ]
[ 2 2 2 2 3 3 3 3 2 3 1 3 ]
[ 2 2 2 2 3 3 3 3 2 3 2 1 ]
```

FINAL FILTERED ALIGNMENT :

```
Aligned esté with esté
Aligned Monstreerent with Monstreerent
Aligned lors with lors
Aligned leur with leur
Aligned clarté with clarté
Aligned solaux with soulaus
Aligned lune with lune
Aligned lurent with lurent
Aligned Qui with Qui
Aligned lores with lors
Aligned primes with primes
Aligned apparurent with apparurent
```

L'alignement dans ces cas est tout à fait correcte.

### 5.3 Extensions et améliorations possibles

Une première observation concerne la base de données : l'accès à *MedievLex* est lent et l'on constate que le nombre d'entrées est plutôt limité. Pour résoudre le premier problème, la symétrie du concept de similitude a été exploitée de manière à accéder une seule fois à la base de données pour évaluer  $similitude\_max(i,j)$  et  $similitude\_max(j,i)$  : ceci veut dire que pour des matrices carrées le temps de calcul de la matrice des traductions a été réduit de moitié. Il faut quand même remarquer que cette amélioration ne suffit pas à accélérer substantiellement l'algorithme, ce qui pourrait poser quelques problèmes lors d'une application en temps réel.

L'enrichissement en termes de *MedievLex* ou l'emploi de bases de données sémantiques comme *WordNet*<sup>1</sup> seraient un remède très efficace à la pauvreté actuelle de ressources sémantiques. L'emploi de *WordNet* par exemple serait utile pour le traitement de toutes les formes synonymes actuellement non reconnues, par exemple des adverbes de même signification.

Une autre amélioration possible est au niveau de l'algorithme dynamique : au lieu de considérer un voisinage de largeur 1 et hauteur 1, la fenêtre pourrait être augmentée à  $(l,k)$  : cette solution serait mieux adaptée pour les segments qui présentent de fortes variations, mais ralentirait l'exécution de l'alignement et serait donc peu souhaitable pour une application en temps réel.

Une extension substantielle au projet serait l'emploi de techniques telles que les algorithmes génétiques et la méthode d'Expectation-Maximisation pour calibrer au

---

<sup>1</sup> dictionnaire informatisé dont l'unité de base est le concept (et non pas le mot).

mieux les paramètres de l'alignement : actuellement les barèmes pour les probabilités de traduction et de décalage sont choisis "à la main", et cette même considération s'applique aussi aux coefficients de la combinaison linéaire dans la fonction de pondération  $W(\langle i, j \rangle)$ .

Ces techniques permettraient un calcul *ad hoc* des constantes, qui serait optimales pour tout document pris individuellement. Encore une fois ces solutions itératives sont envisageables seulement dans un contexte statique.

## Chapitre 6

# Implémentation

L'implémentation de ce projet concerne l'utilisation de la méthode d'alignement créée dans *Medievalz*, un logiciel développé en *Java* au LITH-EPFL pour visualiser des documents médiévaux en plusieurs modes, incluant la lecture, l'alignement et la recherche codicologique<sup>1</sup>.

### 6.1 Lecture des fichiers TMX

Le logiciel en question était configuré pour lire des fichiers d'extension `.txt` et les parser avec le parseur `JDom`<sup>2</sup>. La première tâche a donc consisté à l'ajout d'une fonctionnalité permettant la lecture de fichiers TMX. Les fichiers faisant partie du paquetage `org.epfl.medievalz.ui` ont été repris et modifiés. Le résultat est l'affichage de l'alignement *par paragraphes* contenu dans les fichiers TMX dans la fenêtre principale, à l'intérieur de deux `DocumentPanels`.

### 6.2 Alignement en temps réel de deux documents

Une fois l'affichage réalisé, l'utilisateur devrait pouvoir sélectionner le segment qui l'intéresse et démarrer, en *temps réel*, le calcul de l'alignement au niveau des mots pour ce segment et son correspondant. C'est pourquoi une fonctionnalité supplémentaire a été prévue pour le logiciel, qui consiste à faire apparaître un menu contextuel quand l'utilisateur clique sur un segment TMX. Ce menu présente une entrée appelée 'Aligner mots' qui lance le calcul de l'alignement ; une fois calculé, l'alignement est affiché dans une fenêtre contenant un tableau. Pour chaque ligne de ce tableau il y a trois colonnes : dans les deux premières sont affichés le mot  $S_i$  du document source et son correspondant  $D_j$  du document destination, dans la troisième le poids de l'alignement correspondant,  $W(<i,j>)$ . Ceci donne une mesure de la fiabilité de la connection établie.

---

<sup>1</sup>voir [1].

<sup>2</sup>voir [6].

### 6.3 Liste des classes Java ajoutées au paquetage `org.epfl.medievalz.utils`

Classe	principale fonctionnalité
<code>Cooccurrence.java</code>	Recherche des mots cooccurrents et calcul du rapport de Dice d'ordre 2 pour tous les mots de deux textes.
<code>Dice_2.java</code>	Calcul du rapport de Dice d'ordre 2 pour deux mots.
<code>DistanceLexico.java</code>	Méthodes calculant la distance lexicographique.
<code>InitialAlignment.java</code>	Algorithme d'alignement initial.
<code>ProgDynamique.java</code>	Algorithme dynamique.
<code>Similitude.java</code>	Accès à la base de données <i>MedievLex</i> (Cette classe a été en partie créée par Sandro Saitta [sandro.saitta@epfl.ch]).
<code>TmxReader.java</code>	Lecture et parsing des fichiers TMX.
<code>Trimmer.java</code>	Boîte à outils pour le traitement des chaînes de caractères, utilisation des expressions régulières pour la version jdk 1.3 de <i>Java</i> .
<code>WordAlignment.java</code>	Classe principale qui s'occupe de tout le processus d'alignement de deux segments.

TAB. 6.1 – Nouvelles classes Java

## **Remerciements**

Je tiens à remercier le Professeur G. Coray pour m'avoir permis de réaliser ce projet de semestre qui fût très stimulant et qui m'a appris beaucoup sur ce sujet plutôt prometteur.

Je remercie également le Dr. Hatem Ghorbel pour ses conseils et sa disponibilité pendant toute la durée de mon travail.

# Bibliographie

- [1] H. Ghorbel : *Alignement multicritères des textes appliqué aux documents médiévaux : critères linguistiques et structurels*, Thèse, EPFL, Lausanne, 2002.
- [2] Y. Choueka, E. S.Conley and I. Dagan : *A comprehensive bilingual word alignment system. Application to disparate languages : Hebrew and English*. In J. Véronis ed., *Parallel Text processing*, pages 69-99, Boston and London, 2000.
- [3] Projet TMX : <http://www.lisa.org/tmx>
- [4] A. Belaid et Y. Belaid, *Reconnaissance des formes : Méthodes et applications*. InterEditions, 1992.
- [5] R. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.
- [6] Parser JDom : <http://xml.apache.org/xerces-j/apiDocs/org/apache/xerces/parsers/DOMParser.html>

# Table des figures

2.1	Algorithme alignDiagonally.....	7
2.2	Algorithme alignDynamically.....	7
4.1	Algorithme computePath.....	14
4.2	Algorithme checkValues.....	15
5.1	Algorithme computeInjectiveMatrix .....	18
5.2	Algorithme computeSurjectiveMatrix.....	19
5.3	Algorithme computeBijectiveMatrix .....	19

# Liste des tableaux

3.1	Probabilités de traduction . . . . .	11
3.2	Probabilités de décalage . . . . .	12
6.1	Nouvelles classes Java . . . . .	28

## Annexe A

# Extraits des fichiers TMX utilisés

### A.1 Document vers-vers : paris871-paris373.tmx

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tmx version="1.3">
<header creationtool="Perl script" creationtoolversion="0.1"
datatype="PlainText" segtype="sentence" adminlang="FR"
sclang="paris871 " o-tmf="ABCTransMem">
<prop>paris871 paris373</prop>
</header>
<body>
<tu tuid="88">
<tuv xml:lang="FR-paris871">
<seg> 1 Se l'escripture ne me ment,[1a]
2 Tout est pour nostre enseignement
3 Quanqu'il a en livres escript,
4 Soient bon ou mal li escript.
5 Qui bien y vourroit prendre esgart
6 Ly maulx y est que l'en s'en gart,
7 Ly biens pour ce que l'en le face;
8 Et a qui Dieux donne cuer et grace
9 De conquerre sens et savoir,
10 Il ne doit pas la bouche avoir
11 Trop chiere au bien dire et espondre,
12 Quar nulz ne doit son sens respondre,
13 quar ne vault sens que l'en enserre
14 Ne plus qu'avoir respus en terre. 15</seg>
</tuv>
<tuv xml:lang="FR-paris373">
<seg> 1 C L'escripture ne me ment,[1a]
2 Tout est pour nostre enseignement
3 Quanqu'il a es livres escript,
4 Soient bon ou mal li escript.
5 Qui bien y veult prendre regart,
6 Le mal y est que l'en s'en gart,
```

7 Le bien pour ce que l'en le face;  
8 Et cui Dieux donne cuer et grace  
9 De conquerre sens et savoir,  
10 Il ne doit pas sa bouche avoir  
11 Trop chiere au bien dire et espondre,  
12 Car nul ne doit son sens repondre,  
13 Car ne vault sens que on enserre  
14 Ne plus qu'avoir muciez en terre. 15</seg>

</tuv>

</tu>

<tu tuid="86">

<tuv xml:lang="FR-paris871">

<seg>Pour ce me plest que j'encommans

16 Traire de latin en romans

17 Les fables de l'ancien temps;

18 S'en diray ce que j'en entens 19</seg>

</tuv>

<tuv xml:lang="FR-paris373">

<seg>Pour ce me plait que je commans

16 Traire de latin en rommans

17 Les fables de l'ancien temps;

18 S'en diray ce que j'en entens 19</seg>

</tuv>

</tu>

.....

<tu tuid="4">

<tuv xml:lang="FR-paris871">

<seg>La mue n'a riens ou penser

450 Fors a son corps paistre et tensor.[3c] 451</seg>

</tuv>

<tuv xml:lang="FR-paris373">

<seg>La mue n'a riens ou penser

450 Fors a son corps paistre et tensor. 451</seg>

</tuv>

</tu>

<tu tuid="2">

<tuv xml:lang="FR-paris871">

<seg>Homs doit penser a sauver s'ame

452 Qui du corps est maistrasse et dame,

453 Et a deservir Paradis.</seg>

</tuv>

<tuv xml:lang="FR-paris373">

<seg>Homs doit penser a sauver s'ame[3c]

452 Qui du corps est maistresse et dame,

453 Et a deservir Paradiz.</seg>

</tuv>

</tu>

</body></tmx>

## A.2 Document prose-vers : geneve-paris137.tmx

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tmx version="1.3">
<header creationtool="Perl script" creationtoolversion="0.1"
datatype="PlainText" segtype="sentence" adminlang="FR"
sclang="prosel37" o-tmf="ABCTransMem">
<prop>prosel37 geneve</prop>
</header>
<body>
<tu tuid="1">
<tuv xml:lang="FR-prosel37">
<seg> %Toutes escriptures soient bonnes et mauvaises sont
pour nostre prouffit et doctrine faittes # </seg>
</tuv>
<tuv xml:lang="FR-geneve">
<seg> @ 1 Se l'escripture ne me ment,[5a]
2 Tout est pour nostre enseignement
3 Quant qu'il a es livres escript,
4 Soient bien ou mal li escript. </seg>
</tuv>
</tu>
<tu tuid="2">
<tuv xml:lang="FR-prosel37">
<seg> les bonnes affin d'y prendre exemple de bien faire
et les mauvaises affin que on se garde et abstienne de
mal faire # </seg>
</tuv>
<tuv xml:lang="FR-geneve">
<seg> 5 Qui bien y veult prendre regart,
6 Le mal y est que l'en s'en gart,
7 Le bien pour ce que l'en le face. </seg>
</tuv>
</tu>
.....
<tu tuid="113">
<tuv xml:lang="FR-prosel37">
<seg> * +Avoir doit difference entre homme et beste mue qui n'a
riens a quoy penser fors a lui paistre et a son corps garder </seg>
</tuv>
<tuv xml:lang="FR-geneve">
<seg>444 Homs est plus noble creature,
445 Si doit, puis qu'il a congnoissance,
446 Avoir aucune difference 447 Entre l'omme et la beste mue
448 Qui n'a raison ne entendue. </seg>
</tuv>
</tu>
```

## Annexe B

# Extrait de la base de données *MedievLex*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- edited with XML Spy v4.2 (http://www.xmlspy.com)
by chatelain (unige) -->

<Medlex2>

  <Famille Nom="abaissier">
    <Categorie Type="Verbe" Groupe="1 (-er + -ier)"
      Lemme="abaissier" Transitif-direct="OUI">
      <Morpho Mode="Infinitif" Radical="abaiss">
        <Surface>abaissier</Surface>
      </Morpho>
      <Sens>abaissier</Sens>
    </Categorie>
  </Famille>
  <Famille Nom="abatre">
    <Categorie Type="Verbe" Groupe="3c (-re)" Lemme="abatre"
      Transitif-direct="OUI">
      <Morpho Mode="Infinitif" Radical="abat">
        <Surface>abatre</Surface>
      </Morpho>
      <Sens>abattre</Sens>
    </Categorie>
  </Famille>
  <Famille Nom="abelir">
    <Categorie Type="Verbe" Groupe="2 (-ir + -iss-)" Lemme="abelir"
      Transitif-direct="OUI" Intransitif="OUI">
      <Morpho Mode="Indicatif" Temps="Passe" Radical="abeli"
        Personne="3" Nombre="singulier">
        <Surface>abeli</Surface>
      </Morpho>
      <Sens>plaire</Sens>
```

```
</Categorie>
</Famille>
```

.....

```
<Famille Nom="my">
  <Categorie Type="Complementaire">
    <Morpho>
      <Surface>my</Surface>
    </Morpho>
    <Sens/>
  </Categorie>
</Famille>
<Famille Nom="ne">
  <Categorie Type="Adverbe">
    <Morpho>
      <Surface>ne</Surface>
    </Morpho>
    <Sens/>
  </Categorie>
</Famille>
<Famille Nom="or">
  <Categorie Type="Conjonction">
    <Morpho>
      <Surface>or</Surface>
      <Surface>ore</Surface>
      <Surface>ors</Surface>
    </Morpho>
    <Sens/>
  </Categorie>
</Famille>
<Famille Nom="ou">
  <Categorie Type="Complementaire">
    <Morpho>
      <Surface>ou</Surface>
    </Morpho>
    <Sens/>
  </Categorie>
</Famille>
<Famille Nom="pas">
  <Categorie Type="Adverbe">
    <Morpho>
      <Surface>pas</Surface>
    </Morpho>
    <Sens/>
  </Categorie>
</Famille>
<Famille Nom="point">
  <Categorie Type="Complementaire">
    <Morpho>
```

```

        <Surface>point</Surface>
    </Morpho>
    <Sens/>
</Categorie>
</Famille>
<Famille Nom="quant">
    <Categorie Type="Adverbe">
        <Morpho>
            <Surface>quant</Surface>
            <Surface>Quant</Surface>
        </Morpho>
        <Sens/>
    </Categorie>
</Famille>
<Famille Nom="un">
    <Categorie Type="Determinant">
        <Morpho Genre="feminin" Nombre="singulier" Radical="un">
            <Surface>une</Surface>
        </Morpho>
        <Morpho Genre="masculin" Nombre="singulier" Radical="un">
            <Surface>un</Surface>
            <Surface>uns</Surface>
            <Surface>ung</Surface>
        </Morpho>
        <Sens/>
    </Categorie>
</Famille>
</Medlex2>

```