# Proposition de recommandation pour l'annotation syntaxique des corpus du CCFM – avec logiciels perl*–

Nicolas Mazziotta, Université de Liège
nicolas.mazziotta@ulg.ac.be

5-6 octobre 2006, ENS Lyon

Si nous voulons ouvrir la possibilité d'études générales sur la syntaxe de l'ancien français et ses relations avec les autres parties de la grammaire et avec la structuration de la langue écrite, il faut pouvoir donner des analyses syntaxiques exhaustives des textes que nous rassemblons.

Le choix d'XML comme norme d'encodage généralisée, conséquence prévisible de la nécessité de disposer rapidement d'outils efficaces pour traiter les données, mène à un autre choix concernant les annotations syntaxiques : celui d'un modèle arborescent. L'adéquation suffisante du modèle arborescent pour l'ancien français nous autorise à le sélectionner, mais il est loin de faire l'unanimité [1]. Nous ne reviendrons pas sur ce choix.

Cette contribution est composée de deux parties : dans la première, nous énoncerons une série de principes qui nous semblent intéressants à suivre pour atteindre le but recherché ; dans la seconde, nous ferons des propositions de règles concrètes de balisage qui pourraient faire l'objet d'une extension à la recommandation de la TEI (TEI P5 0.4.2).

Nous founissons en annexe quelques logiciels qui exploitent les conventions qui vont être décrites.

## 1 Principes

Dans un premier temps, nous expliquerons comment il est possible de détacher une analyse de l'édition sur laquelle elle porte, et pourquoi nous trouvons cette solution incontournable (1.1). Nous insisterons ensuite sur la nécessité d'éviter à tout prix, dans la perspective d'une collaboration de chercheurs ayant des intérêts différents, la *pensée unique* (1.2). Enfin, nous mettrons l'accent sur le besoin que les propositions de recommandation soient suffisamment ergonomiques pour être applicables (1.3).

### 1.1 Séparer l'analyse de l'édition

Les créateurs de l'*American National Corpus* [2] recommandent que les données issues de l'analyse soient détachées de celles fournies par les matériaux eux-mêmes :

*La composition de la présente contribution, ainsi que la conception des logiciels fournis en annexe, ont été entièrement réalisées à l'aide de logiciels libres.

1. Habert *et al.* (1997 : 44-45) mentionne l'existence d'alternatives.
2. Voir http://americannationalcorpus.org/.

> The recommended practice in encoding annotated corpora is to maintain all or most annotations in separate documents, each of which references appropriate locations in the document containing the original data. (Ide 1998)

Ce procédé est connu sous le nom de *stand-off markup* (balisage « à part »). Nous commencerons par expliquer l'intérêt d'une telle pratique avant de montrer comment elle peut être mise en œuvre avec des documents XML.

### 1.1.1 Intérêt

Du point de vue (abstrait) de la relation entre les analyses, les textes analysés et les lecteurs, on peut faire trois observations générales :

1. les analyses sont des constructions qui viennent *après* l'édition des textes et considèrent ces derniers comme des *données* ;
2. les analyses ne sont ni complètement consensuelles, ni parfaites ;
3. il y a une infinité d'analyses d'égale valeur scientifique qui puissent être faites à partir d'un texte donné.

Reprenons chacune de ces observations. Nous allons voir à quel point il est nécessaire d'éviter d'encoder les analyses directement dans les éditions.

*a. Les analyses viennent après l'édition.* Nous pensons qu'édition et analyse sont des objets distincts. Préserver cette distinction sépare en outre le travail de l'éditeur de celui du linguiste, et permet de partager des éditions de référence avec des scientifiques travaillant sur des sujets littéraires sans leur imposer une surcharge d'information qui ne les intéresse pas. Par ailleurs, l'ordre de surface (celui donné dans l'édition) peut être différent de l'ordre de la structure profonde (donnée par l'analyse), et on voit mal comment intégrer facilement et efficacement les deux ordres à une seule représentation hybride.

*b. Les analyses ne sont pas consensuelles.* Cela implique que tout le monde ne soit pas d'accord avec l'analyse proposée. Elle peuvent même être *fausses*. Encoder les analyses « à côté » du texte permet de les réviser, d'en proposer d'autres, ou même de les jeter sans altérer l'édition.

*c. Il y a une infinité d'analyses possibles.* Pour des raisons pratiques, il est inconcevable d'encoder plusieurs analyses arborescentes (ce qui n'implique pas exclusivement des analyses syntaxiques) sur un même texte : la surcharge de balises deviendrait très vite impossible à gérer. De plus, d'un point de vue technique, les hiérarchies multiples ne font pas bon ménage avec XML [3].

Il est donc obligatoire de séparer les données de l'analyse de celles de l'édition. Pour ce faire, on utilise généralement le principe du *pointeur*.

### 1.1.2 Mise en œuvre : le principe du pointeur

Pour ces raisons, Nancy Ide (1998) recommande que le balisage soit effectué comme suit :

3. Cela est dû aux contraintes définitoires de la structure `Element`, voir XML REC : §3.

> This strategy yelds, in essence, a finely linked hypertext format, where the links specify a semantic role rather than navigational options. That is, links signify the location(s) where markup contained in a given annotation document *would appear* in the document to which it is linked. As such, annotation information comprises remote or « stand-off » markup that is virtually added to the document to which it is linked.

Le principe du *pointeur* est extrêmement courant en informatique. Vu de manière très abstraite, le pointeur est un élément (au sens le plus général du terme) qui n'a pas d'autre valeur que celle de renvoyer à une autre unité. En quoi cela consiste-t-il concrètement dans un corpus employant XML ? Traduit en XML et adapté aux recommandations de la TEI, le concept s'applique comme suit. Chaque unité minimale de l'analyse (p. ex. : les mots) porte un identifiant unique (attribut `@id`), qui les distingue des autres unités de tous ordres. Prenons comme exemple l'adresse d'une charte du treizième siècle :

> ··**A** tos ceaz ki ces preſenſ **l**etres veront ⸗ oront [...] (Document 1277-03-23 : 1)

pourrait par exemple être encodé comme suit (encodage TEI).

```
<lb id='_1234'/>
<c type='ponct' id='_381'>··</c>
<w id='_1'><c id='_1236'>A</c></w>
<w id='_2'>tos</w>
<w id='_3'>ceaz</w>
<w id='_4'>ki</w>
<w id='_5'>ces</w>
<w id='_6'>pre&#x017F;en&#x017F;</w>
<w id='_7'><c id='_1243'>l</c>etres</w>
<w id='_8'>veront</w>
<w id='_9'><expan>et</expan></w>
<w id='_10'>oront</w>
```

Par la suite, n'importe quelle analyse, quelle qu'elle soit, pourra faire référence à ces unités sans les mentionner directement : il suffira de citer leur identifiant pour les retrouver. Pour ce faire, la TEI définit les éléments `ptr` et `link`, qui font office de pointeurs [4], et dont l'attribut `@target` permet d'établir la relation avec les données qu'ils ciblent. Par exemple

```
<link target='_1'/>
```

n'est qu'une référence à l'élément

```
<w id='_1'><c id='_1236'>A</c></w>
```

du texte encodé ci-dessus. On peut donc imaginer n'importe quel type d'annotation supplémentaire portant sur le pointeur ; par exemple, une mise en forme :

```
<em><link target='_1'/></em>
```

mais également une lemmatisation ou une analyse syntaxique.

4. Cf. TEI P4X : §14.1 et ensuite TEI P5 0.4.2 : §14.1. Pour simplifier, nous limitons notre exposé aux pointeurs dirigés vers le même document que celui où ils se trouvent. La recommandation TEI P4X : §14.2 définit également les éléments `xptr` et `xlink`, qui permettent l'hypertextualité. La TEI P5 0.4.2 recommande une syntaxe qui distingue les pointeurs hypertextuels des autres uniquement par le contenu de leur attribut `target` (une URI et non un simple identifiant interne).

## 1.2 Minimalisme des recommandations

Une fois accepté le principe des pointeurs, il pourrait sembler nécessaire de se mettre d'accord sur le type d'information qui sera encodé dans l'analyse. Cela permettrait de partager des analyses comparables.

À notre avis, il serait extrêmement dommageable qu'une association de linguistes définisse un ensemble d'étiquettes précis pour l'analyse syntaxique. En supposant que ces linguistes puissent tomber d'accord, ce jeu comporterait potentiellement trois défauts scientifiquement inacceptables.

Premièrement, il pourrait souffrir d'*inertie*. Une fois appliquées par le plus grand nombre, ces étiquettes seraient exploitées dans des logiciels *ad hoc*, ce qui aurait comme inévitable conséquence d'imposer l'emploi de ces étiquettes à tout utilisateur. De ce point de vue, si les logiciels ne sont pas génériques, ils contraignent obligatoirement l'analyse dans un cadre qui ne pourrait convenir à tous. Cela équivaudrait à figer la réflexion linguistique.

Deuxièmement, ce qui, à nos yeux, fait l'intérêt d'une analyse syntaxique, c'est sa faculté à mettre en évidence un *système*. Un jeu d'étiquettes choisi en consensus ou prévu pour être étendu a de fortes chances d'être éclectique, et donc *asystématique*.

Enfin, en supposant que le jeu d'étiquettes soit consistant, il s'en faudrait de beaucoup pour qu'il soit appliqué de la même manière par tous les participants au projet, dans toutes les conditions, à moins que ne soient établies d'interminables listes faisant « jurisprudence » (Habert *et al.* 1997 : 157).

En conséquence, nous pensons qu'il est nécessaire de limiter les recommandations à des considérations d'ordres méthodologique et technique, sans imposer une manière spécifique d'analyser le corpus. Il est possible de se mettre d'accord sur une façon de faire commune qui ne soit pas (trop) contraignante du point de vue de la liberté des modèles d'analyse.

Un format ouvert laissant toute latitude à l'utilisateur en ce qui concerne la définition sémantique des annotations, nous recommandons d'employer le mécanisme d'analyse syntaxique pour toute élaboration concernant la structure du texte, comme par exemple, les divisions diplomatiques d'une charte ou la progression rhétorique d'un texte argumenté.

## 1.3 Concessions à l'ergonomie

*a. Lourdeur des logiciels traitant les pointeurs.* Du point de vue (concret) de la pratique d'encodage et du partage, les pointeurs sont difficiles à gérer en XML. En fait, ce dernier étant basé sur le format *plain text*, il ne permet pas d'optimiser le principe [5].

Techniquement, le système des pointeurs oblige non seulement à stocker ces références en mémoire, mais en plus, il impose l'emploi du modèle DOM [6] pour accéder au document XML, ce qui signifie concrètement que tout le document doit être chargé en mémoire pour être traité. Par ailleurs, utiliser des références force les programmes à reconstituer les données pour livrer une visualisation satisfaisante à l'utilisateur humain.

En conséquence, les applications qui gèrent ce principe risquent d'être gourmandes en matière de mémoire vive et de nécessiter un très grand nombre d'opérations (ce qui

5. Contrairement aux bases de données relationnelles, où les données sont généralement compilées et peu hiérarchisées. Dans ces conditions, qui ne sont pas celles de nos corpus, les pointeurs sont très efficaces en termes de performances.

6. Voir DOM Level 1.

charge forcément le microprocesseur). L'emploi de références a également le désavantage de créer des fuites de mémoire si l'on n'y prend pas garde.

*b. Difficultés d'encoder les données sans interface.* Tous ces désavantages techniques compliquent la tâche de l'utilisateur. Les applications étant difficiles à concevoir, elles tardent à être disponibles. Tout qui veut annoter est ainsi le plus souvent contraint de manipuler directement les données XML dans un éditeur gérant ce format de fichier texte… Or, si l'on introduit l'abstraction liée aux pointeurs, l'encodage devient impossible à réaliser : `<link target='_1'/>` est loin d'être sémantiquement explicite.

La plupart des logiciels propriétaires, coûteux de surcroît, échouent à résoudre ce problème.

# 2 Proposition de recommandation

Nous proposons de définir ici deux formats : un pour le travail (2.1), qui fait la part belle à l'ergonomie, et un pour l'échange (2.2), plus abstrait, générique et permettant la validation de manière traditionnelle.

Nous baserons nos exemples sur le syntagme suivant (extrait de celui que nous avons présenté sous 1.1.2)

```
1 <w id='_3'>ceaz</w>
2 <w id='_4'>ki</w>
3 <w id='_5'>ces</w>
4 <w id='_6'>pre&#x17F;en&#x17F;</w>
5 <w id='_7'><c id='_1243'>l</c>etres</w>
6 <w id='_8'>veront</w>
```

et sur l'analyse syntaxique en constituants immédiats (à laquelle nous n'adhérons pas [7], mais qui constitue une base didactique satisfaisante) de ce syntagme :

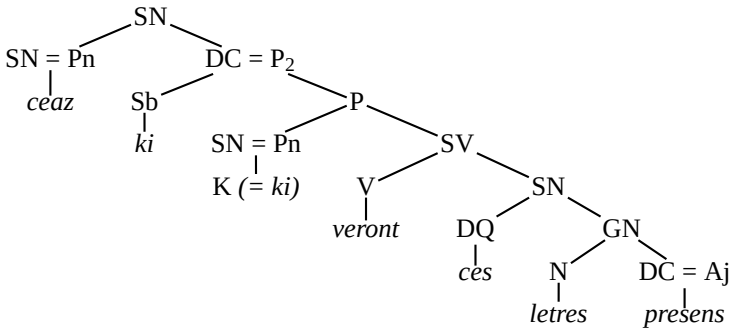


FIG. 1 – ACI

L'analyse consiste toujours : 1/ à isoler des segments (constituants) ; 2/ à les caractériser.

## 2.1 Format de travail

Le format de travail que nous proposons essaye de résoudre les problèmes posés par les pointeurs aux points de vue de la réalisation de logiciels de l'encodage manuel

7. En particulier, cette analyse est mimétique du français moderne, et la règle de réécriture P → SN + SV n'a aucun sens pour l'ancien français.

des données. Nous en donnerons d'abord une définition non formalisée[8], puis nous l'exemplifierons en l'appliquant à l'analyse ACI qui nous sert de support. Nous verrons ensuite quelles sont les implications techniques de ce format avant de réfléchir à une manière d'améliorer son ergonomie.

*a. Définition.* Nous travaillons sur un élément `div` dont l'attribut `@type` spécifie quel type d'analyse il contient. Cet élément inclut une hiérarchie de structures explicitée par des éléments `S` imbriqués (un seul élément à la racine). La hiérarchie XML est mimétique de la hiérarchie syntaxique telle que l'utilisateur la conçoit. Les nœuds terminaux de la structure sont des éléments `L`, qui ne sont que des pointeurs vers les mots du texte. Les éléments `S` reçoivent un nombre indéterminé d'attributs, de nom et de valeur laissés à l'appréciation de l'utilisateur. On minimise ainsi la quantité d'éléments et on simplifie l'arborescence en plaçant les informations dans des attributs.

*b. Application.* L'encodage de l'analyse en constituants immédiats présentée ci-dessus donnerait ceci (`@f` exprimerait la fonction et `@n` la nature[9]).

```
1  <div type='analyse syntaxique ACI'>
2   <S id='_S1' f='SN'>
3    <S id='_S2' f='Pn'>
4     <L target='_3'/>
5    </S>
6    <S id='_S3' f='DC' n='P2'>
7     <S id='_S4' f='Sb' n='K'>
8      <L target='_4'/>
9     </S>
10    <S id='_S5' f='P'>
11     <S id='_S6' f='SN' n='Pn'>
12      <L target='_4'/>
13     </S>
14     <S id='_S7' f='SV'>
15      <S id='_S8' f='V'>
16       <L target='_8'/>
17      </S>
18      <S id='_S9' f='SN'>
19       <S id='_S10' f='DQ'>
20        <L target='_5'/>
21       </S>
22       <S id='_S11' f='GN'>
23        <S id='_12' f='n'>
24         <L target='_7'/>
25        </S>
26        <S id='_S12' f='DC' n='Aj'>
27         <L target='_6'/>
28        </S>
29       </S>
30      </S>
31     </S>
32    </S>
33   </S>
34  </S>
35 </div>
```

On voit clairement que le système de pointeurs permet : d'une part de *multiplier* les références à un seul mot (lignes 8 et 12, l'attribut `@target` pointe vers l'élément vérifiant `@id == '_4'`), et, d'autre part, de *déplacer* les pointeurs, de sorte que leur ordre ne corresponde pas à celui des unités qu'ils représentent (p. ex., lignes 24 et 27).

*c. Implications.* D'un point de vue technique, cette manière de procéder a plusieurs implications. Tout d'abord, les hiérarchies syntaxiques comprennent un très grand nombre d'éléments. De ce fait, la longueur du nom des balises peut augmenter sensiblement la taille du document chargé en mémoire, surtout si le fichier code les caractères sur plus

8. C'est-à-dire que nous n'écrirons pas uns DTD ou un Schema de définition.
9. Quand celle-ci n'est pas automatiquement impliquée par la fonction.

d'un octet – à moins que le parseur DOM ne soit optimisé pour l'éviter (ce dont on ne peut être certain sans en voir et comprendre la source). Employer des noms brefs (un caractère) évite ce problème.

L'emploi systématique d'attributs a également des avantages :

1. on accède facilement aux attributs par la méthode de l'objet `Element` définie dans la spécification (DOM Level 1 : §1.2) : `getAttribute()` ;
2. on minimise la mémoire nécessaire au chargement du document – les structures qui représentent les nœuds de type `Attr` (attributs) ont moins de propriétés que les nœuds de type `Element` (DOM Level 1 : §1.2).

Mais il bloque certaines possibilités :

1. on ne peut valider facilement le document : chaque analyse nécessite l'importation d'une DTD appropriée ;
2. on ne peut pas non plus structurer l'information contenue dans l'attribut – ce qui est rarement nécessaire pour ce type d'analyse.

*d. Concession à l'ergonomie.* Si l'utilisateur n'a pas d'interface appropriée à sa disposition, il est possible de travailler sur des fichiers présentant unse structure de données explicitant la forme de la destination du pointeur :

```
<div type='analyse syntaxique ACI'>
 <S f='SN'>
  <S f='Pn'>
   <L target='_3' forme='ceaz' />
  </S>
  <S f='DC' n='P2'>
   <S f='Sb' n='K'>
    <L target='_4' forme='ki' />
   </S>
   <S f='P'>...</S>
  </S>
 </S>
</div>
```

Mais il faut à tout prix éviter de stocker ce genre de représentation dans les fichiers, surtout si l'édition est susceptible d'être révisée.

## 2.2 Format d'échange

Le format d'échange distingue la hiérarchisation de la caractérisation des constituants. Il fait appel au même système de pointeurs que le format de travail, mais ajoute une couche d'abstraction supplémentaire. La représentation prend la forme d'un élément, qui comprend les informations de segmentation et de hiérarchisation et les informations de caractérisation des segments en les distinguant.

### 2.2.1 Structure de la « racine » de l'analyse

L'élément qui contient l'analyse arborescente est nommé `anaGrp`. Il contient les éléments `anaChunk` (représentation arborescente des constituants) et `anaNotes` (annotation des constituants). Cette « racine » comporte un attribut `@type`, qui précise à quel type d'analyse on a affaire.

```
<anaGrp type='analyse syntaxique ACI'>
   <anaChunk>...</anaChunk>
   <anaNotes>...</anaNotes>
 </anaGrp>
```

### 2.2.2 Structure de l'élément `anaChunk`

Voyons tout d'abord comment les constituants sont représentés.

*a. Définition (non formalisée).* La structure de l'élément `anaChunk` n'est pas très différente de celle du `div` présentée ci-dessus (format de travail, 2.1) : `anaChunk` comprend un élément racine `seg` pourvu d'un identifiant (ce qui correspond au `S` de la structure de travail). Cet élément sert à représenter la racine de l'arbre syntaxique, ainsi que tous les autres nœuds de l'arbre syntaxique sont représentés à l'aide d'un élément de même nom. Les éléments terminaux (mots) sont rendus par des éléments `link` pourvus d'un attribut `@target`, qui identifie le mot sur lequel ils portent. Par contre, le contenu de l'analyse, spécifié par des attributs de `S` dans le format de travail est absent de cet arbre.

*b. Application.* Concrètement, une fois cette définition appliquée à l'exemple pris comme base, on obtient

```
<anaChunk>
 <seg id='_S1'>
  <seg id='_S2'>
   <link target='_3'/>
  </seg>
  <seg id='_S3'>
   <seg id='_S4'>
    <link target='_4'/>
   </seg>
   <seg id='_S5'>
    <seg id='_S6'>
     <link target='_4'/>
    </seg>
    <seg id='_S7'>
     <seg id='_S8'>
      <link target='_8'/>
     </seg>
     <seg id='_S9'>
      <seg id='_S10'>
       <link target='_5'/>
      </seg>
      <seg id='_S11'>
       <seg id='_S12'>
        <link target='_7'/>
       </seg>
       <seg id='_S13'>
        <link target='_6'/>
       </seg>
      </seg>
     </seg>
    </seg>
   </seg>
  </seg>
 </seg>
</anaChunk>
```

### 2.2.3 Structure de l'élément `anaNotes`

La caractérisation des `seg` est organisée dans l'élément `anaNotes`.

*a. Définition (non formalisée).* L'élément `anaNotes` contient une structure organisée de pointeurs qui servent à caractériser les éléments `seg` descendants de `anaChunk`. Au lieu d'enregistrer les données sous la forme d'attributs, comme dans le format de travail, le format d'échange évite de répéter les mêmes infomations, en ajoutant une nouvelle série de pointeurs (et donc un niveau d'abstraction). Pour comprendre comment nous avons mis cela en œuvre, repartons de la structure de travail ; soit l'extrait :

```
<S id='_S12' f='DC' n='Aj'>
 <L target='_6'/>
</S>
```

Deux analyses sont appliquées au constituant identifié par l'élément `S` : `@f = 'DC'` et `@n = 'Aj'`. Pour optimiser le format d'échange, nous proposons de classer les analyses par type (le type est rendu par le *nom* des attributs dans le format de travail), puis

par valeur (*valeur* des attributs). Il peut y avoir plusieurs types d'analyse mélangés, et chacun de ces types admet plusieurs valeurs. Par exemple, le type `@f` admet les valeurs `SN`, `SV`, `DC`, etc.

Il nous semble que la hiérarchie suivante convient à l'encodage de cette information :

```
<anaNotes>
 <anaTag type='f'>
  <anaValue type='DC'>
   ...
  </anaValue>
 </anaTag>
 <anaTag type='n'>
  <anaValue type='Aj'>
   ...
  </anaValue>
 </anaTag>
</anaNotes>
```

où l'on voit que l'élément `anaNotes` ne contient que des éléments `anaTag` (représentant le type d'analyse, spécifié par `@type`), comprenant à leur tour des éléments `anaValue` (représentant la valeur de l'analyse, exprimée par un attribut `@type` également).

En représentant les `seg` qui sont concernés par chacune des `anaValue` par un pointeur `link` descendant, on met en relation les segments et leur caractérisation.

*b. Application.* Si l'on applique ce principe à l'analyse qui nous sert de base de réflexion, on obtient :

```
<anaNotes>
 <anaTag type='f'>
  <anaValue type='DC'>
   <linkGrp>
    <link target='_S3'/>
    <link target='_S13'/>
   </linkGrp>
  </anaValue>
  <anaValue type='DQ'>
   <linkGrp>
    <link target='_S10'/>
   </linkGrp>
  </anaValue>
  <anaValue type='GN'>
   <linkGrp>
    <link target='_S11'/>
   </linkGrp>
  </anaValue>
  <anaValue type='N'>
   <linkGrp>
    <link target='_S12'/>
   </linkGrp>
  </anaValue>
  <anaValue type='P'>
   <linkGrp>
    <link target='_S5'/>
   </linkGrp>
  </anaValue>
  <anaValue type='Pn'>
   <linkGrp>
    <link target='_S2'/>
   </linkGrp>
  </anaValue>
  <anaValue type='SN'>
   <linkGrp>
    <link target='_S1'/>
    <link target='_S6'/>
    <link target='_S9'/>
   </linkGrp>
  </anaValue>
  <anaValue type='SV'>
```

```
42      <linkGrp>
43       <link target='_S7'/>
44      </linkGrp>
45     </anaValue>
46     <anaValue type='Sb'>
47      <linkGrp>
48       <link target='_S4'/>
49      </linkGrp>
50     </anaValue>
51     <anaValue type='V'>
52      <linkGrp>
53       <link target='_S8'/>
54      </linkGrp>
55     </anaValue>
56    </anaTag>
57    <anaTag type='n'>
58     <anaValue type='Aj'>
59      <linkGrp>
60       <link target='_S13'/>
61      </linkGrp>
62     </anaValue>
63     <anaValue type='K'>
64      <linkGrp>
65       <link target='_S4'/>
66      </linkGrp>
67     </anaValue>
68     <anaValue type='P2'>
69      <linkGrp>
70       <link target='_S3'/>
71      </linkGrp>
72     </anaValue>
73     <anaValue type='Pn'>
74      <linkGrp>
75       <link target='_S6'/>
76      </linkGrp>
77     </anaValue>
78    </anaTag>
79  </anaNotes>
```

#### 2.2.4 Implications.

Nous ne reviendrons pas sur les problèmes liés à l'abstraction qu'introduit l'ajout de pointeurs supplémentaires. Voyons plutôt ce que ce mécanisme apporte.

Tout d'abord, ce procédé permet de valider (au sens technique) les documents [10], puisque les différentes annotations ne sont plus encodées sous la forme d'attributs, mais sous forme de de données textuelles.

Ensuite, le procédé permet de séparer la segmentation de la caractérisation des segments. Au delà de l'analyse syntaxique, on peut imaginer qu'on éprouve le besoin d'ajouter une analyse d'un autre ordre, mais portant sur la même hiérarchie de constituants. Plutôt que de créer deux arborescences de pointeurs désignant les éléments du texte, le format sépare plutôt deux groupes de pointeurs se rapportant aux éléments de l'analyse.

Dans une perspective de partage, cela signifie également qu'il facilite la sélection du type d'information à transmettre. Dans une perspective hypertextuelle, le format permet de ranger les types d'analyse basés sur le même découpage des constituants en plusieurs fichiers, ce qui rend l'archivage et le suivi des versions très simple.

### 2.3 Déclarer le jeu d'étiquettes choisi

Le minimalisme de la recommandation ne doit cependant pas empêcher les utilisateurs et les développeurs de vérifier la validité, en particulier dans une perspective

10. Par contre, il ne permet pas de valider les analyses, par définition évolutives en ce qui concerne les langues anciennes.

d'échange. À cet égard, il semble intéressant de se pencher sur ce qu'offre le format TIGER-XML (König *et al.* 2003 : ch. V), dont les ingénieurs de la *Talbanken05*[11] parlent en ces termes :

> The encoding format we use to convey the phrase structure and dependency structure is TIGER-XML [...]. It is one attempt to create a more generic format for representing various corpora and treebanks. The format is designed to be theory-independent, but it is especially suited for treebanks using phrase structure annotation schema, where the encoding uses node labels (syntactic categories) and edge labels (grammatical functions) for creating the syntactical structure. (Nilsson *et al.* 2005 : 2)

Le format TIGER-XML permet en effet d'encoder n'importe quelle analyse syntaxique, pour peu qu'elle soit accompagnée d'une déclaration qui la définit. Cette déclaration constitue en une liste organisée des étiquettes choisies pour l'analyse. Adaptant le nom des balises Tiger-XML pour qu'elles soient cohérentes avec celles proposées plus haut, nous proposons le format suivant :

```
<anaDecl type='analyse syntaxique ACI'>
 <anaTagDecl type='f'>
  <anaValueDecl type='SN'>syntagme nominal<anaValueDecl>
  <anaValueDecl type='SV'>syntagme verbal<anaValueDecl>
  <anaValueDecl type='DC'>déterminant caractérisant<anaValueDecl>
  ...
 </anaTagDecl>
 <anaTagDecl type='n'>
  <anaValueDecl type='Aj'>adjectif</anaValueDecl>
  <anaValueDecl type='Pn'>pronom</anaValueDecl>
  ...
 </anaTagDecl>
</anaDecl>
```

L'élément `anaDecl` englobe l'ensemble de la déclaration. Il comprend une liste d'élements `anaTagDecl` qualifiés d'un attribut `@type`. Cette liste correspond aux types d'`anaTag` que l'analyse pourra employer dans l'élément `anaNotes`.

Les éléments `anaValueDecl` décrivent par leur `@type` les valeurs possibles pour les `anaValue` regroupés sous les `anaNotes/anaTag`, suivant les mêmes règles de hiérarchisation que ci-dessus (2.2). Le texte des `anaValueDecl` explicite le sens des abréviations employées comme `@type`.

# 3 Conclusion : du concret

Nous terminerons cette contribution en proposant une manière concrète de se servir de la recommandation présentée.

Nous préconisons que les annotations se fassent sur le format de travail et que le stockage emploie le format d'échange. Les programmes qui manipulent les données devraient commencer par convertir le format d'échange en format de travail, puis purger leur mémoire des données du format d'échange. L'opération inverse devrait être faite en cas de sauvegadre du travail.

On trouvera en annexe un script perl (`work2share.pl`, qui utilise le module `Ana`, également fourni) qui permet de faire des conversions simples d'un format à l'autre. Il peut être modifié facilement en cas de besoin. Nous fournissons également le script `buildtree.pl`, qui construit les pointeurs à partir d'un texte segmenté en unités `w`. La version informatique de cette contribution est accompagnée de fichiers destinés à

11. Banque de textes suédois arborés, `http://w3.msi.vxu.se/~nivre/research/Talbanken05.html`.

tester ces programmes : `test_work.xml`, `test_share.xml` pourront être convertis avec `work2share.pl` et `charte.xml` [12], dont on peut construire la liste des pointeurs avec `buildtree.pl`.

# 4 Annexe : logiciels

On trouvera ici la source le module de base, `Ana.pm`, qui étend le module perl `XML::LibXML` [13]. pour traiter les éléments définis ci-dessus. Nous incluons également les deux scripts `work2share.pl` et `buildtree.pl`, mentionnés dans la conclusion. Enfin, nous avons inclus le script shell `demo.sh` qui effectue une démonstration de ces programmes (nécessite un shell *bash* [14]).

## 4.1 Ana.pm

```
# Copyright (c) 2006 Université de Liège, Nicolas Mazziotta
# $Id$
# This program is free software; you can redistribute it and/or
# modify it under the same terms as Perl itself.
#
# Extension de XML::LibXML permettant de traiter les
# analyses syntaxiques CCFM.
#
############################################################
############################################################
package Ana;
############################################################
############################################################

use strict;
use XML::LibXML;

#====================================================#
sub  XML::LibXML::Document::AnaBuild
#====================================================#
# Construit la liste de pointeurs et la place à l'endroit
# voulu dans le document.
{
 my ($xml_doc, $target_nodes_xpath, $ana_node_xpath, $position, $mode, $formes) = @_;
 my $ana_node = XML::LibXML::Element->new("div");
 $ana_node->setAttribute("type", "analyse syntaxique");
 foreach ($xml_doc->findnodes($target_nodes_xpath)) {
  my $elt = XML::LibXML::Element->new("L");
  if ($formes) {
   my $text = $_->textContent;
   $text =~ s/\s+/ /g;
   $elt->setAttribute($formes, $text);
  }
  my $ref = $_->getAttribute("id");
  if ($ref) {
   $elt->setAttribute("target", $ref)
  } else {
   die "Tous les éléments qui sont ciblés par l'analyse doivent être pourvus d'identifiants!\n"
  };
  $ana_node->appendChild($elt);
 }
 $ana_node = &_work2share($ana_node) if $mode =~ /^s(h(a(r(e)?)?)?)?$/;
 $xml_doc->AnaAppend($ana_node, $ana_node_xpath, $position);
}

#====================================================#
sub  XML::LibXML::Document::AnaAppend
#====================================================#
# retourne le document modifié
# en collant le nom à l'endroit indiqué par le xpath
# et en ajustant sa position (before|after|firstChild|lastChild)
```

12. Ce document est une version simplifiée de l'édition de Document 1236-07, faite dans le cadre du projet *Khartês* (voir Mazziotta 2004 pour une présentation sommaire).
13. Module téléchargeable sur le site du CPAN, `http://www.cpan.org`.
14. Logiciel téléchargeable sur `http://www.gnu.org/software/bash/bash.html`.

```
{
 my ($xml_doc, $ana_elt, $xpath_locator, $position) = @_;
 $position = "lastChild" unless $position;
 foreach ($xml_doc->findnodes($xpath_locator)) {
  if ($position eq "firstChild") {
   $_->insertBefore($ana_elt, $_->firstChild) ;
  } elsif ($position eq "before") {
   $_->insertBefore($ana_elt, $_) ;
  } elsif ($position eq "after") {
   $_->insertAfter($ana_elt, $_) ;
  }
  else {
   $_->appendChild($ana_elt);
  }
  last;
 }
 return $xml_doc;
}

#=====================================================#
sub  XML::LibXML::Document::AnaGet
#=====================================================#
# Trouve l'analyse et la coupe de l'arbre
{
 my ($xml_doc, $xpath_locator) = @_;
 foreach my $ana ($xml_doc->findnodes($xpath_locator)) {
  return $ana->parentNode->removeChild($ana);
 }
}

#=====================================================#
sub  XML::LibXML::Element::AnaConvert
#=====================================================#
# Convertit l'analyse
{
 my ($base_ana, $direction) = @_;
 if (!$direction) {
  my $ana_name = $base_ana->nodeName();
  if ($ana_name eq "anaGrp") { $direction = "work" }
  else { $direction = "share" }
 } elsif ($direction =~ /^n(o(n(e)?)?)?$/) { return $base_ana }
 my $converted_ana;
 if ($direction =~ /^w(o(r(k)?)?)?$/) {
  $converted_ana = &_share2work($base_ana);
 } elsif ($direction =~ /^s(h(a(r(e)?)?)?)?$/) {
  $converted_ana = &_work2share($base_ana);
 }
 return $converted_ana;
}

#=====================================================#
sub  XML::LibXML::Element::AnaPurge
#=====================================================#
# détruit certains noeuds de l'analyse (utile pour
# enlever les noeuds qui explicitent les pointeurs)
{
 my ($ana_node, $xpath) = @_;
 foreach ($ana_node->findnodes($xpath)) {
  if ($_->nodeType == 2) {
   $_->parentNode->removeAttribute($_->nodeName)
  } else {
   $_->parentNode->removeChild($_);
  }
 }
 return $ana_node;
}

#=====================================================#
sub  _share2work
#=====================================================#
# (privé) convertit du format de travail au format
# de partage
{
 my ($share_ana) = @_;
 my $work_ana_node;
 my $work_ana_notes;
 my $type = $share_ana->getAttribute("type");
 foreach ($share_ana->findnodes("./anaChunk")) {
  $work_ana_node = $share_ana->removeChild($_);
  $work_ana_node->setAttribute("type", $type);
```

```
  $work_ana_node->setNodeName("div");
  last
 }
 foreach ($share_ana->findnodes("./anaNotes")) {
  $work_ana_notes = &__ana_struct_from_xml($_);
  last
 }

 foreach my $seg ($work_ana_node->findnodes(".//seg")) {
  my $ref = $seg->getAttribute("id");
  die "Tous les éléments d'analyse doivent être pourvus d'identifiants!\n" unless $ref;
  $seg->setNodeName("S");
  if ($ref and my $data = $work_ana_notes->{$ref}) {
   foreach (@$data) { $seg->setAttribute(%$_); }
  }
  foreach ($seg->findnodes("./link")) {
   $_->setNodeName("L");
  }
 }


 return $work_ana_node;
}

#=====================================================#
sub  _work2share
#=====================================================#
# (privé) inverse de _share2work
{
 my ($work_ana_node) = @_;
 my $anaGrp_node = XML::LibXML::Element->new("anaGrp");

 my $type = $work_ana_node->getAttribute("type");
 $type = "" unless $type;
 $anaGrp_node->setAttribute("type", $type);

 # contiendra l'arbre final
 my $tree_node = $work_ana_node->cloneNode(1);
 $tree_node->removeAttribute("type");
 # contiendra les données d'annotation, puis sera sérialisé en XML
 my $annotations_struct = {};

 $tree_node->setNodeName("anaChunk");

 foreach my $seg_node ($tree_node->findnodes(".//S")) {
  # on change le nom des noeuds
  $seg_node->setNodeName("seg");
  my $ref;
  # on extrait la valeur de chaque attribut
  foreach ($seg_node->findnodes("./attribute::*")) {
   my $name = $_->nodeName;
   my $value = $_->value;
   if ($name eq "id") {
    $ref = $value;
    next
   }
   # on range tout dans le hashref qui structure les annotations
   push @{$annotations_struct->{$name}{$value}}, $ref;
   $seg_node->removeAttribute($name);
  }
  foreach ($seg_node->findnodes("./L")) {
   $_->setNodeName("link");
  }
 }
 foreach ($tree_node->findnodes("./L")) {
  $_->setNodeName("link");
 }

 my $anaChunk_node = $tree_node;
 $anaGrp_node->appendChild($anaChunk_node);

 my $anaNotes_node = &__ana_struct_2_xml($annotations_struct);
 $anaGrp_node->appendChild($anaNotes_node);

 return $anaGrp_node;
}

#=====================================================#
sub  __ana_struct_from_xml
#=====================================================#
# (privé) Convertit les données de l'analyse (format de
```

```
# partage) en structure de données perl.
{
 my ($struct_node) = @_;
 my $struct = {};
 foreach my $tag_node ($struct_node->findnodes("./anaTag")) {
  my $tag = $tag_node->getAttribute("type");
  foreach my $value_node ($tag_node->findnodes("./anaValue")) {
   my $value = $value_node->getAttribute("type");
   foreach my $ref ($value_node->findnodes("./linkGrp/link/attribute::target")) {
    push @{$struct->{$ref->value}}, {$tag => $value};
   }
  }
 }
 return $struct;
}


#=====================================================#
sub  __ana_struct_2_xml
#=====================================================#
# (privé) inverse de __ana_struct_from_xml
{
 my ($struct) = @_;
 my $anaNotes_node = XML::LibXML::Element->new("anaNotes");
 foreach my $tag (sort keys %{$struct}) {
  my $anaTag_node = XML::LibXML::Element->new("anaTag");
  $anaTag_node->setAttribute("type", $tag);
  foreach my $value (sort keys %{$struct->{$tag}}) {
   my $anaValue_node = XML::LibXML::Element->new("anaValue");
   $anaValue_node->setAttribute("type", $value);
   my $linkGrp_node = XML::LibXML::Element->new("linkGrp");
   foreach my $ref (@{$struct->{$tag}{$value}}) {
    my $link_node = XML::LibXML::Element->new("link");
    $link_node->setAttribute("target", $ref);
    $linkGrp_node->appendChild($link_node);
   }
   $anaValue_node->appendChild($linkGrp_node);
   $anaTag_node->appendChild($anaValue_node);
  }
  $anaNotes_node->appendChild($anaTag_node);
 }
 return $anaNotes_node;
}

1;

# vim:ts=2 sw=2

```

## 4.2 work2share.pl

```
#!/usr/bin/perl -w
# Copyright (c) 2006 Université de Liège, Nicolas Mazziotta
# $Id: not versioned$
# This program is free software; you can redistribute it and/or
# modify it under the same terms as Perl itself.
#
# Effectue les conversions du format de travail au format
# d'échange et inversement

use strict;

# Init

my ($name,$execpath,$suffix);

use File::Basename;

BEGIN {
  # find where the program has been installed
  my $filename = $0;
  eval {$filename = readlink $0 if -l $0};
  ($name,$execpath,$suffix) = fileparse($filename,qw/pl/);
 $execpath =~ s/\\/\//g if $^O =~ /^MSWin/;
}

use lib "$execpath/lib";
```

```
use XML::LibXML;
use Ana;
use Getopt::Std;

my %opt;
getopts('l:t:p:d:hf:', \%opt) || die ;

my $usage = <<USAGE;
$0 -l xpath [-t xpath] [-p position] [-d direction] [-h] [-f xpath] fichier à convertir
Options:
  -f activée, cette option recherche tous les noeuds correspondant
     et les efface AVANT la conversion (utile pour enlever les formes);
     par exemple ".//L/attribute::forme"
     (attention, notez la présence du contexte au début du chemin)
  -d direction de conversion: "none", "work" (vers format de travail)
     ou "share" (vers format de partage). "none" ne convertit pas;
  -h affiche le présent message et sort
  -l chemin du noeud où se trouvent les annotations du
     document à convertir, sous la forme d'un xpath;
     par exemple: '/text/annotations[1]'
  -p position où insérer les annotations par rapport au
     chemin indiqué par -t: "before", "after", "firstChild"
     ou "lastChild" (par défaut)
  -t chemin où insérer les annotations converties;
     par exemple: './back/div[2]' (/ par défaut)
USAGE

if ($opt{h} or !$opt{l}) { die $usage; }
if (!$opt{d}) { print STDERR "Aucune direction spécifiée par '-d'.
Je vais essayer de deviner la direction avec le nom des balises.\n"; }
if (!$opt{t}) { print STDERR "Aucune destination spécifiée par '-t'.
Je mets l'analyse dans la racine du document.\n";
 $opt{t} = '/*';
}

my ($origin_xpath, $target_xpath, $position, $direction, $xpath_to_delete)
   =
   ($opt{l}, $opt{t}, $opt{p}, $opt{d}, $opt{f});

foreach (@ARGV) {

 my $parser = XML::LibXML->new();
 my $doc = eval {$parser->parse_file($_)};
 die $@ if $@;

 eval {
  my $base_ana = $doc->AnaGet($origin_xpath);
  $base_ana->AnaPurge($xpath_to_delete) if $xpath_to_delete;
  my $converted_ana = $base_ana->AnaConvert($direction);
  my $new_doc = $doc->AnaAppend($converted_ana, $target_xpath, $position);
  print $new_doc->toString(1);
 };
 die $@ if $@;

}



__END__

# vim:sw=2 ts=2

```

## 4.3 buildtree.pl

```
#!/usr/bin/perl -w
# Copyright (c) 2006 Université de Liège, Nicolas Mazziotta
# $Id: not versioned$
# This program is free software; you can redistribute it and/or
# modify it under the same terms as Perl itself.
#
# Construit un arbre syntaxique constitué de pointeurs à partir
# d'un texte segmenté.

use strict;

# Init

```

```
my ($name,$execpath,$suffix);

use File::Basename;

BEGIN {
  # find where the program has been installed
  my $filename = $0;
  eval {$filename = readlink $0 if -l $0};
  ($name,$execpath,$suffix) = fileparse($filename,qw/pl/);
 $execpath =~ s/\\/\//g if $^O =~ /^MSWin/;
}

use lib "$execpath/lib";
use XML::LibXML;
use Ana;
use Getopt::Std;

my %opt;
getopts('f:l:t:p:m:h', \%opt) || die ;

my $usage = <<USAGE;
$0 -l xpath [-t xpath] [-h] [-p] fichier à convertir
Options:
  -f intègre un attribut avec la forme graphique
     des noeuds qui seront analysés. L'argument est le nom
     de l'attribut; par exemple "forme";
  -h affiche le présent message et sort
  -l chemin des noeuds qui feront l'objet de l'analyse,
     sous la forme d'un xpath;
     par exemple: '/text/w'
  -m mode de création: "work" (défaut) ou "share"
  -p position où insérer les annotations par rapport au
     chemin indiqué par -t: "before", "after", "firstChild"
     ou "lastChild" (par défaut)
 -t chemin où insérer les annotations converties;
     par exemple: './back/div[2]' (/ par défaut)
USAGE

if ($opt{h} or !$opt{l}) { die $usage; }
if (!$opt{t}) { print STDERR "Aucune destination spécifiée par '-t'.
Je mets l'analyse dans la racine du document.\n";
 $opt{t} = '/*';
}
if (!$opt{m}) { print STDERR "Aucun mode de création spécifié par '-m'.
Je crée un arbre de travail.\n";
 $opt{m} = "work";
}

my ($origin_xpath, $target_xpath, $position, $mode, $formes)
   =
   ($opt{l}, $opt{t}, $opt{p}, $opt{m}, $opt{f});

foreach (@ARGV) {

 my $parser = XML::LibXML->new();
 my $doc = eval {$parser->parse_file($_)};
 die $@ if $@;

 eval {
  $doc->AnaBuild($origin_xpath, $target_xpath, $position, $mode, $formes);
  print $doc->toString(1);
 };
 die $@ if $@;

}



__END__

# vim:sw=2 ts=2

```

## 4.4 demo.sh

```
#!/bin/bash
cd '../texts'

echo "
```

```
Conversion du document charte.xml
---------------------------------"
echo ". Construction de l'arbre"
../bin/buildtree.pl -l './/w' -f 'forme' charte.xml \
| xmllint --format - > 'demo/charte.anabuilt.xml'
echo ". Exportation pour le partage"
../bin/work2share.pl -l '/charte/div' -d 'share' -f './/L/attribute::forme' \
'demo/charte.anabuilt.xml' > 'demo/charte.shareable.xml'

echo "
Conversion des tests
--------------------"
echo ". Exportation pour le partage"
../bin/work2share.pl -l '/test/div[2]' -d 'share' 'test_work.xml' \
| xmllint --format -> 'demo/test_work.shareable.xml'
echo ". Exportation pour le travail"
../bin/work2share.pl -l '/test/anaGrp' \
'test_share.xml' > 'demo/test_share.workable.xml'


echo "
Syntaxe des commandes
---------------------"
../bin/buildtree.pl -h
../bin/work2share.pl -h

echo "
J'ai terminé
------------
Si tout s'est bien passé, les fichiers xml contenus dans
texts/demo sont identiques à ceux contenus dans l'archive présente
dans le même répertoire."
```

# Références

## Travaux cités


Apparao, Vidur, Byrne, Steve, Champion, Mike, Isaacs, Scott, Jacobs, Ian, Le Hors, Arnaud, Nicol, Gavin, Robie, Jonathan, Sutor, Robert, Wilson, Chris et Wood, Lauren and (1998). *Document Object Model (DOM) Level 1 Specification. Version 1.0. W3C Recommendation 1 October, 1998*, http ://www.w3.org/TR/REC-DOM-Level-1.

Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eva et Yergeau, François ($^{3}$2004). *Extensible Markup Language (XML) 1.0. W3C Recommendation 04 February 2004*, http ://www.w3.org/TR/2004/REC-xml-20040204/.

DOM Level 1 = Apparao *et al.* 1998.

Habert, Benoît, Nazarenko, Adeline et Salem, André (1997). *Les linguistiques de corpus*, Paris : Armand Colin (U Linguistique).

Ide, Nancy (1998). *The XML Framework and Its Implications for the Development of Natural Language Processing Tools*, http ://citeseer.ist.psu.edu/ide98xml.html.

König, Esther, Lezius, Wolfgang et Voormann, Holger (2003). *TIGERSearch 2.1. User's Manual*, http ://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/, dernière modification au 1$^{er}$ septembre 2003.

Mazziotta, Nicolas (2004). « Le texte dans tous ses états. Philosophie d'encodage du projet Khartês », dans Purnelle, Gerald, Fairon, Cédrick et Dister, Anne, éds, *Le poids des mots*, Louvain : Presses universitaires de Louvain : 793-803.

Nilsson, Jens, Hall, Johan et Nivre, Joakim (2005). *MAMBA Meets Tiger : reconstructing a treebank from Antiquity*, http ://w3.msi.vxu.se/ jni/publications.html.

Sperberg-McQueen, C. M., Burnard, Lou, Bauman, Syd, DeRose, Steven et Rahtz, Sebastian (2004). *TEI P4. Guidelines for Electronic Text Encoding and Interchange. XML-compatible edition*, http ://www.tei-c.org/P4X/.

Sperberg-McQueen, C. M., Burnard, Lou, Bauman, Syd, DeRose, Steven et Rahtz,

Sebastian (2005). *TEI P5. Guidelines for Electronic Text Encoding and Interchange. Version 0.4.1*, http ://www.tei-c.org/P4X/.
TEI P4X = Sperberg-McQueen *et al.* 2004.
TEI P5 0.4.2 = Sperberg-McQueen *et al.* 2005.
XML REC = Bray *et al.* 2004.

## Documents d'archives

Document (1236-07), *juillet*, Archives de l'État à Liège (Cathédrale Saint-Lambert à Liège), main inconnue.

Document (1277-03-23), *23 mars*, Archives de l'État à Liège (Collégiale Saint-Martin à Liège), main A.