Création d'un multi-arbre à partir d'un texte balisé : l'exemple de l'annotation d'un corpus d'oral spontané

Julie Belião
LPP - Université Paris Sorbonne Nouvelle (ILPGA) - CNRS - UMR 7018
MoDyCo - Université Paris Ouest Nanterre La Défense - CNRS - UMR 7114
julie@beliao.fr

RÉSUMÉ

Dans cette étude, nous nous intéressons au problème de l'analyse d'un corpus annoté de l'oral. Le système d'annotation considéré est celui introduit par l'équipe des syntacticiens du projet Rhapsodie. La principale problématique qui sous-tend un tel projet est que la base écrite sur laquelle on travaille est en réalité une transcription de l'oral, balisée par les annotateurs de manière à délimiter un ensemble de structures arborescentes. Un tel système introduit plusieurs structures, en particulier macro et micro-syntaxiques. Du fait de leur étroite imbrication, il s'est avéré difficile de les analyser de façon indépendante et donc de travailler sur l'aspect macro-syntaxique indépendamment de l'aspect micro-syntaxique. Cependant, peu d'études jusqu'à présent considèrent ces problèmes conjointement et de manière automatisée. Dans ce travail, nous présentons nos efforts en vue de produire un outil de parsing capable de rendre compte à la fois de l'information micro et macro-syntaxique du texte annoté. Pour ce faire, nous proposons une représentation partant de la notion de multi-arbre et nous montrons comment une telle structure peut être générée à partir de l'annotation et utilisée à des fins d'analyse.

ABSTRACT

Creating a Multi-Tree from a Tagged Text : Annotating Spoken French

This study focuses on automatic analysis of annotated transcribed speech. The annotation system considered has been recently introduced to address the several limitations of classical syntactic annotations when faced to natural speech transcriptions. It introduces many different components such as embedding, piles, kernels, pre-kernels, discursive markers etc.. All those components are tightly coupled in a complex tree structure and can hardly be considered separately because of their close intrication. Hence, a joint analysis is required but no analysis tool to handle them all together was available yet. In this study, we introduce such an automatic parser of annotated transcriptions of speech and present the corresponding framework based on multi-trees. This framework permits to jointly handle separate aspects of speech such as macro and micro syntactic levels, which are traditionnaly considered separately. Several applications are proposed, including analysis of the transcribed speech by classical parsers designed for written language.

MOTS-CLÉS : Arbres syntaxiques, unité illocutoire, unités rectionnelles, micro-syntaxe, macro-syntaxe, entassement.

KEYWORDS: Syntactic trees, illocutionary unit, microsyntax, macrosyntax, piles.

Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 3: RECITAL, pages 109–122,
Grenoble, 4 au 8 juin 2012.

1 Introduction

Le projet ANR Rhapsodie (Rhapsodie, 2012) a pour but de créer un corpus de trois heures de parole transcrite, annoté en prosodie et en syntaxe, qui serve de référence pour le français parlé. Une analyse et une annotation prosodique et syntaxique sont réalisées indépendamment l'une de l'autre sur l'ensemble du corpus, de manière à permettre une étude intono-syntaxique(Benzitoun *et al.*, 2009)(Benzitoun *et al.*, 2010)(Lacheret-Dujour *et al.*, 2011). Nous nous concentrerons ici sur la tâche d'exploitation informatique du corpus annoté syntaxiquement.

La problématique qui sous-tend le système d'annotation syntaxique de Rhapsodie est que le français parlé transcrit ne présente que peu de similitudes par rapport à la syntaxe de l'écrit pour pouvoir être traité directement par des parsers syntaxiques tels que FRMG (de la Clergerie *et al.*, 2009). Les transcriptions de l'oral sur lesquelles ont travaillé les syntacticiens ne sont ni ponctuées, ni segmentées et comportent un grand nombre de phénomènes propres à l'oral tels que les disfluences, les greffes (décrites en section 3.3), etc. Ce sont ces particularités inhérentes à la transcription du français parlé qui posent problème aux parsers classiques. Les syntacticiens de Rhapsodie ont donc développé un système d'annotation syntaxique centré sur le cas de l'oral (Benzitoun *et al.*, 2009)(Benzitoun *et al.*, 2010). Leurs travaux se basent sur ceux de l'école d'Aix (Blanche-Benveniste *et al.*, 1990) et sur la syntaxe de dépendance introduite par Tesnière (Tesnière, 1959). Ce système de balisage manuel dispose de suffisamment de souplesse pour rendre compte d'un grand nombre de phénomènes relatifs à la micro et à la macro-syntaxe. Cependant, aucune représentation informatique n'était jusqu'à présent disponible pour exploiter ce formalisme de manière automatisée. Dans cette étude, nous introduisons une telle représentation et montrons comment elle peut être mise à profit pour l'analyse de la parole transcrite annotée.

Dans un premier temps, nous présenterons brièvement les différents niveaux micro et macro-syntaxiques considérés en section 2. Ensuite, nous préciserons le système d'annotation utilisé en section 3. Enfin, la notion de multi-arbre sera discutée pour sa représentation informatique en section 4 puis exploitée dans le but de procéder à une analyse automatique de la parole annotée.

2 Phénomènes micro et macro syntaxiques

2.1 Unités rectionnelles

L'approche adoptée ici est une approche *"de bas en haut"* (Benzitoun *et al.*, 2010; Lacheret-Dujour *et al.*, 2011). Une Unité Rectionnelle (UR) est une unité construite autour d'une tête, qui n'est à priori syntaxiquement dépendante d'aucun élément de rang supérieur dans le texte. La rection est caractérisée par les contraintes imposées sur une position donnée en termes de parties du discours, de marques morphologiques et de possibilités de restructuration (commutation avec un pronom, effacement, passivation, clivage, etc.). Il est important de souligner le fait que les UR ne sont pas définies dans l'absolu. C'est toujours relativement à un texte donné que l'on peut affirmer raisonnablement que certaines constructions ne dépendent d'aucune catégorie du contexte. Les UR, unités micro-syntaxiques sont souvent considérées comme les unités significatives maximales et sont définies à la fois par leur connexité rectionnelle interne et par leur autonomie externe (Berrendonner, 2002) : *"La micro-syntaxe vise à décrire des constructions syntaxiques conçues comme des ensembles rectionnels complets" (Benzitoun* et al.*, 2010)*.

2.2 Unités Illocutoires

Parallèlement à l'UR, il y a l'Unité Illocutoire (UI) dont la délimitation est liée à la reconnaissance de la force illocutoire qui peut affecter un segment dans un texte. UR et UI sont des unités relativement autonomes qui ont leurs propres règles de formation et leurs propres combinatoires. L'UI fait partie de la macro-syntaxe et *"on appelle unité illocutoire une portion de discours comportant un unique acte illocutoire, soit une assertion, soit une interrogation, soit une injonction*". (Benzitoun *et al.*, 2010)

Les syntacticiens de Rhapsodie ont considéré que ces deux modules de l'analyse syntaxique sont complémentaires mais que la sortie de l'un ne constitue pas l'entrée de l'autre. Ainsi les UI sont constituées d'UR variées, allant de l'interjection à des constructions plus complexes à plusieurs enchâssements. Les UI peuvent donc combiner plusieurs UR, mais leurs frontières ne coïncident pas forcément entre elles.

Le principe d'annotation consiste à segmenter par une balise adéquate dès que l'on ne peut plus effectuer de rattachement micro-syntaxique à l'intérieur du texte. Dans le cadre de notre étude, ce travail est effectué manuellement.

Chaque UI se décompose en un certain nombre d'unités, prosodiquement marquées — c'est du moins l'hypothèse qui est faite : (Blanche-Benveniste, 1997)(Cresti, 2000) — que l'on appelle composantes illocutoires (CI). Ces unités sont nommées suivant leur position par rapport au noyau. Le noyau (kernel) est l'UR qui comporte la force illocutoire de l'UI. Les autres UR, dépourvues de force illocutoire, s'associent au noyau et sont appelées : prénoyaux (prekernel) à gauche du noyau, in-noyaux (inkernel) dans le noyau ou post-noyaux (postkernel) à droite du noyau.

3 Balisage de la transcription

Dans cette section, nous présentons le système de balisage manuel introduit dans (Benzitoun *et al.*, 2009)(Benzitoun *et al.*, 2010) et permettant d'annoter la transcription selon les niveaux de la micro et macro-syntaxes.

3.1 Balisage des UI

– Les UI sont délimitées par le symbole **//** qui est une marque de fin d'UI [1] :
a. ***on peut après passer le concours de l'agreg pour enseigner à l'université*** **//** *(échantillon M103-Corpus Rhapsodie)*
– Par défaut, le symbole **//**, qui marque la fin d'une UI, marque aussi la fin d'une UR. Cependant, UR et UI ne se correspondent pas toujours. Le symbole **//+** (le + indique de manière générale une relation de rection) indique que l'UR se poursuit après la fin de l'UI.
b. ***c'est un Chinois //+ très riche //*** *(échantillon D210-Corpus Rhapsodie)*
– La notion de parallélisme permet d'identifier la mise en paralèlle de deux UIs. Les parallèlismes sont annotés par le signe **//=**.
c. "oh" tout est relatif //= tout est relatif // (échantillon D009-Corpus Rhapsodie)

1. Pour des raisons pédagogiques, tous les exemples donnés dans cet article sont simples et ne comportent chacun qu'une partie des phénomènes étudiés afin de les mettre en évidence. Cela-dit, il est entendu que le formalisme présenté est opérationnel et a été testé pour plus de trois heures de français parlé spontané. La plupart du temps l'ensemble des phénomènes sont réalisés simultanément.

3.2 Balisage des marqueurs d'UR et de Composante illocutoire

3.2.1 Pré-noyau, post-noyaux, in-noyaux

– Le pré-noyau, annoté **<** (ou **<+** si relation de rection).
a. ***bien évidemment <*** *c'est vrai pour la peinture religieuse en Occident // (échantillon M202-Corpus Rhapsodie)*
b. ***au début <+*** *il n'y avait pratiquement pas d' informatique // (échantillon D005-Corpus Rhapsodie)*
– Les symboles **>** et **>+** signalent les post-noyaux :
a. ***^*** *et "euh" Charlot s'est accusé* ***> plutôt que de laisser la jeune fille s'accuser*** *// (échantillon M024-Corpus Rhapsodie)*
b. ***^*** *mais vous étiez auprès des femmes* ***>+ là-bas //*** *(échantillon D204-Corpus Rhapsodie)*
– L'in-noyau est annoté par les symboles **( )** et **(+)** :
a. une rallonge à venir ***(également)*** *dans le secteur automobile // (échantillon M206-Corpus Rhapsodie)*
b. le cri de Job ***(+ que nous avons entendu dans la première lecture)*** *retentit à nos oreilles // (échantillon M203-Corpus Rhapsodie)*

3.2.2 Introducteurs

– Une UI peut commencer par un ou plusieurs introducteurs. Ces éléments ont la fonction de préciser la nature de la relation entre l'UI qu'ils introduisent et d'autres UI dans le discours (notamment l'UI qui précède). On les annote par le symbole **^**.
a. ***^donc*** *c'est pas normal qu'ils arrivent en CP ne parlant pas français // (échantillon D002-Corpus Rhapsodie)*
b. ***^et*** *tu arrives à la fontaine place Notre Dame // (échantillon M001-Corpus Rhapsodie)*
– Sont annotés avec le même symbole les marqueurs d'entassement ou joncteurs comme *et, ou, mais,* etc :
c. { les uns | ***^et*** *les autres } (échantillon M203-Corpus Rhapsodie)*

3.3 Balisage des enchâssements et parenthèses

Une UI peut se trouver à l'intérieur d'une autre UI. On distingue deux cas, les enchâssements et les insertions.
– Le discours rapporté dans cet exemple, "*casse-toi pauvre con*" forme une UI. Par contre, "*il a dit*" n'est ni une UI complète, ni une UR complète. On considère donc que "*casse-toi pauvre con*" dans "*il a dit casse-toi pauvre con*" est régi par le verbe dire.
a. il a dit ***[casse-toi > pauvre con //]*** *//*
– La greffe est la réalisation d'une UI au sein d'une UI. *"Il s'agit du procédé qui consiste à remplir une position syntaxique à l'aide d'une autre catégorie que celle attendue"* (Deulofeu, 1999). Ces deux types d'enchâssement sont annotés par des crochets et un marqueur de fin d'UI **[//]** :
b. vous avez dit que ***[disons ma carrière pour simplifier //]*** *témoigne de ma bonne conduite // (échantillon D201-Corpus Rhapsodie)*
– L'enchâssement ne contient pas toujours une UI, en effet il peut aussi contenir des sous-composantes d'une composante illocutoire (CI). Ici il s'agit d'un enchâssement d'une proposition avec un pré-noyau mais qui n'est pas une UI :
c. ce qui fait que ***[au moment de la guerre < nous étions toujours en Bretagne]*** *// (échantillon D003-Corpus Rhapsodie)*

– On parle d'insertion d'UI chaque fois qu'une UI vient interrompre momentanément une autre UI. On utilise les parenthèses simples **( )** pour délimiter l'UI insérée.
*d. "euh" et sinon < les spécialités { les m~ | un { peu moins **(je sais pas si c'est ça qui vous intéress*
***//)** | petit peu moins } } prises < "bah" { { c'est les | c'est les } spécialités à risques //+ | { la gynéc*
obstétrique (par exemple) | la cancérologie } } // (échantillon D006-Corpus Rhapsodie)

3.4 Balisage des Entassements

Les entassement font normalement partie de la micro-syntaxe, l'entassement, aussi appelé pile (voir (Gerdes et Kahane, 2009)(Kahane et Pietrandrea, 2012)(Kahane, 2012)), est un dispositif de connexion syntaxique qui relie tous les éléments qui occupent la même position syntaxique à l'intérieur de l'UR. On utilise les symboles **{** et **}** pour marquer le début et la fin de la liste et **|** pour signaler le ou les points de jonction dans le prolongement des listes paradigmatiques et de l'analyse en grille proposées dans (Blanche-Benveniste, 1990). Les conjoints ne sont pas nécessairement des constituants micro-syntaxiques mais peuvent être des disfluences par exemple :

a. ^ et **{ la | la }** Loire est en bas // *(échantillon D003-Corpus Rhapsodie)*

4 L'arbre complet et son exploitation

La réalisation d'un balisage manuel permettant d'encoder de l'information macro et micro-syntaxique implique de réaliser un parsing de ce balisage afin de pouvoir l'exploiter. Il y a trois raisons importante pour cela :

1. La création à partir du balisage d'une multi-arborescence macro et micro-syntaxique dans la transcription afin d'en extraire des arbres topologiques et d'entassement.
2. Pouvoir à partir des différents parcours de cet arbre, fournir une version dépliée de la transcription afin de faciliter la tâche de l'analyseur syntaxique automatique.
3. À partir de l'arbre initial et des résultats du parser automatique, restituer l'ordre original des mots de la transcription et procéder à leur intégration dans la structure arborescente initialement annotée.

Dans le but de réaliser ces différentes tâches, il a dont fallu implémenter un algorithme capable de réaliser ces différentes tâches.

La grammaire de balisage développée ne rentrant pas dans les cadres classiques des grammaires non-contextuelles ou même des grammaires-contextuelles d'ordre *k*, il a été nécessaire de mettre au point un parser *ad-hoc* pouvant permettre l'analyse de l'intégralité des symboles du balisage. L'objectif de cette partie est de mettre en évidence l'utilité de réunir en tant qu'objets, l'information micro, macro-syntaxique et d'entassement dans un même arbre appelé multi-arbre. Pour des raisons de place, nous ne détaillons pas ici l'agorithme de parsage, décrit dans (Beliao et Liutkus, 2012), mais nous concentrons plutôt sur l'utilité de la représentation arborée qu'il produit.

On appellera arbre une structure de données qui peut se représenter sous la forme d'une hiérarchie dont chaque élément est appelé nœud. Dans notre cas, nous avons choisi d'implémenter un parser qui construit un arbre multiple ou englobant qui représente à la fois l'information micro et macro-syntaxique. Ainsi chaque nœud correspond à un type d'unité et est typé en tant qu'unité illocutoire, entassement, enchâssement, marqueur discursif et de manière générale tout typage donné par le balisage. Ainsi, cet arbre intègre toutes les informations contenues dans le balisage.

L'implémentation d'un tel arbre présente l'avantage de pouvoir être parcouru selon un point de vue macro et micro-syntaxique ou les deux en même temps, permettant différents traitements impliquant ou non toute l'information.

Nous allons présenter en sous-section 4.1 une série d'exemples qui nous permettront de constater que le balisage peut se représenter efficacement sous la forme d'un arbre. Nous présenterons ensuite en sous-section 4.2 l'opération de dépliage, qui consiste, à partir de l'arbre, à générer un ensemble de phrases susceptibles d'être traitées par un analyseur syntaxique automatique. Nous présenterons ensuite en sous-section 4.3 l'opération que nous avons évoquée plus haut et qui consiste à extraire du multi-arbre les noeuds désirés pour en obtenir un arbre particularisé. Nous verrons que l'obtention de l'arbre topologique et de l'arbre des entassements sont des cas particuliers de cette opération de projection. Nous évoquerons ensuite la phase de repliage en section 4.4, qui consiste à réintégrer au multi-arbre l'information donnée par le parser automatique. Enfin, nous montrerons en sous-section 4.5 comment le multi-arbre arbre peut être utilisé de manière naturelle pour convertir l'information d'annotation dans un format structuré tel que XML.

4.1 Exemples d'arbres

Considérons l'exemple suivant :

a. vous avez dit que [disons ma carrière pour simplifier //] témoigne de ma bonne conduite // (échantillon D201-Corpus Rhapsodie)

Cet exemple [2] peut être représenté comme indiqué dans la figure 1. En effet, on voit que cette phrase est composée de deux UI, la deuxième étant enchâssée dans le noyau de la première par une greffe. Cette deuxième UI contient un noyau.

```
|=> root:
    |=> iu:
        |=> ru:
            |=> kernel: vous avez dit que [...] témoigne de ma bonne conduite
                |=> graft:
                    |=> ui:
                        |=> ru:
                            |=> kernel: disons ma carrière pour simplifier
```

FIGURE 1 – Représentation macro de a.

Dans cet exemple, on n'a pas encore considéré le phénomène d'entassement. Considérons donc l'exemple suivant :

b. les fêtes y sont { plus | plus } nombreuses // (échantillon D101-Corpus Rhapsodie)

On voit que cette UI contient une UR de type noyau.

Cependant, certains de ses éléments : "plus, plus" sont entassés (disfluence) selon deux couches. On observe donc par le balisage que ce segment est un entassement inclut dans un noyau. Le typage noyau correspond à une information macro-syntaxique indépendante du typage des entassements qui relève de la micro-syntaxe. On peut assimiler ces deux informations à deux niveaux ou encore deux dimensions différentes du discours et plusieurs approches sont envisageables ici.

2. Pour des raisons didactiques les numéros d'identifiant des lexèmes ont été remplacés par les token-mots dans les exemples donnés.

La première approche consisterait à représenter de manière indépendante les dépendances macro-syntaxiques (noyaux, pré-noyaux, enchâssements, etc) et les informations d'entassements sous la forme de deux arbres "projetés". Elle présenterait l'avantage d'offrir directement au spécialiste une représentation pertinente selon le point de vue désiré. Ainsi, on obtiendrait deux arbres donnés en figure 2, un premier arbre contenant l'information macro et un deuxième arbre contenant l'information des entassements.

Information macro :

```
|=> root:
    |=> iu:
        |=> ru:
            |=> kernel: les fêtes y sont plus plus nombreuses
```

Information des entassements :

```
|=> root:
    |=> ui: les fêtes y sont [...] nombreuses
        |=> pile:
            |=> layer: plus
            |=> layer: plus
```

FIGURE 2 – Abres donnant l'information macro (haut) et l'information des entassements pour l'exemple b. (bas)

Cependant, comme on le verra en sous-section 4.2, certaines taches ne sont plus réalisables si une telle disjonction est faite car l'information portée par l'un des deux typage sera perdue.

Par conséquent, la deuxième approche consiste à intégrer l'ensemble de ces informations dans la même structure, c'est à dire de considérer dans le même arbre l'information macro et micro-syntaxique. Par exemple, on peut représenter l'UI considérée par l'arbre donné en figure 3 (l'idéal étant un graphe) :

```
|=> root:
    |=> iu:
        |=> ru
            |=> kernel: les fêtes y sont [...] nombreuses
            |=> pile:
                |=> layer:
                    |=> kernel: plus
                |=> layer:
                    |=> kernel: plus
```

FIGURE 3 – Représentation macro+entassement de b.

Cet arbre, dit "multiple" contient ainsi l'ensemble des informations contenues dans le balisage, ce qui est nécessaire pour certains traitements comme on le verra en sous-section 4.2. Cependant, il a l'inconvénient pour le syntacticien de ne pas représenter de manière conventionnelle l'information syntaxique. Cela dit, il est possible de ne garder de cet arbre que l'information micro ou macro-syntaxique de manière à obtenir des représentations plus conventionnelles comme on le verra en section 4.3.

Considérons un autre exemple un peu plus complexe :

c. il faut avoir un don spécial parce que [la psychiatrie < { c' est | c' est } quelque chose] // (échantillon D006-Corpus Rhapsodie)

L'ensemble peut être représenté sous la forme de l'arbre donné en figure 4 :

```
|=> root:
    |=> iu:
        |=> ru:
            |=> kernel: il faut avoir un don spécial parce que
            |=> embedded:
                    |=> prekernel: la psychiatrie
                    |=> pile:
                        |=> layer:
                            |=> kernel: c' est
                        |=> layer:
                            |=> kernel: c' est
                    |=> kernel: quelque chose
```

FIGURE 4 – Représentation macro+entassement+micro de *c.*

4.2 Dépliage

L'une des premières applications possibles de l'arbre est de procéder au dépliage du texte. On entend par dépliage du texte un réarrangement des entassements permettant ensuite une analyse syntaxique automatique par un programme informatique. En effet les structures d'entassement, d'enchâssement et de parenthèsage, particulièrement courantes à l'oral, ne sont pas analysables en l'état par les analyseurs syntaxiques qui sont calibrés pour l'écrit. Les parsers ne savent pas traiter les disfluences et font encore beaucoup d'erreurs sur les coordinations (difficulté avec le rattachement du deuxième conjoint). Le dépliage va donc explorer chaque chemin de l'entassement et donne une UR bien formée sans entassement (Gerdes et Kahane, 2009). Il est nécessaire de fournir des segments syntaxiques débarrassés de tout phénomènes de l'oral à ce type de programme.

Considérons le premier exemple suivant :

a. les fêtes y sont { plus | plus } nombreuses // (échantillon D101-Corpus Rhapsodie)

Le dépliage correspondant est donné figure 5.

```
Unrolled sentences:
les fêtes y sont plus nombreuses
les fêtes y sont plus nombreuses
```

FIGURE 5 – Dépliage correspondant à l'exemple a.

Pour obtenir ce dépliage, il nous a fallu prendre en compte les éléments micro-syntaxiques (ici l'entassement) et macro-syntaxiques (l'UI composée d'un noyau), pour ce faire le multi-arbre donné figure 3 est nécessaire.

Soit à présent l'exemple suivant :

b. ^ alors le petit fauteuil { que j'ai { là | à côté } | que je veux rhabiller } a toujours été appelé par mes parents fauteuil-crapaud // (échantillon D009-Corpus Rhapsodie)

Le multi-arbre obtenu est représenté sur la figure 6 et le dépliage qui en résulte sur la figure 7.

```
|=> root:
    |=> iu:
        |=> ru:
            |=> intro: alors
        |=> ru:
            |=> kernel: le petit fauteuil [...] a toujours
                                été appelé par mes parents fauteuil-crapaud
            |=> pile:
                |=> layer:
                    |=> kernel: que j'ai
                    |=> pile:
                        |=> layer:
                            |=> kernel: là
                        |=> layer:
                            |=> kernel: à côté
                |=> layer:
                    |=> kernel: que je veux rhabiller
```

FIGURE 6 – multi-arbre de *b.*

```
alors
le petit fauteuil que j'ai là a toujours été appelé par mes parents fauteuil crapaud
le petit fauteuil que j'ai à côté a toujours été appelé par mes parents fauteuil crapaud
le petit fauteuil que je veux rhabiller a toujours été appelé par mes parents fauteuil crapaud
```

FIGURE 7 – UR dépliées extraites du multi-arbre de b. et envoyée au parser automatique

Ici on constate que chaque alternative d'entassement (chaque couche qui compose la pile) est explorée, chaque couche de chaque pile va venir occuper le rôle syntaxique qu'elle doit occuper. On obtient donc autant d'alternatives qu'il y a de piles et de couches dans une pile. On notera également que les éléments de type introducteurs (ici : *alors*), marqueurs discursifs etc [...] sont "séparés" des autres UR car eux aussi peuvent perturber l'analyse syntaxique automatique. En aucun cas ils ne seront ignorés, ils seront réintégrés (cf section 4.4) aux autres UR après l'analyse en ligne du parser FRMG par un algorithme de "repliage" (cf (Beliao et Liutkus, 2012)).

De plus, il est important de voir que l'information d'entassement n'est pas suffisante à elle seule pour fournir le dépliage de l'arbre, mais que le multi-arbre complet est bien nécessaire à cette tâche. En effet l'information d'entassement est d'ordre micro-syntaxique et n'est pas suffisante pour la tâche de dépliage car on l'a vu plus haut, les marqueurs discursifs aussi peuvent poser problème et que l'on a donc besoin parallèlement de l'information macro. Considérons l'exemple suivant :

c. il faut avoir un don spécial parce que [la psychiatrie < { c' est | c' est } quelque chose] //
(échantillon D006-Corpus Rhapsodie)

L'UI contient un enchâssement dans lequel on remarque une UR de type pré-noyau et un noyau contenant un entassement. Si l'on ignore l'une de ces informations le dépliage ne serait que partiel. Si l'on considère seulement l'information en UR, le pré-noyau "la psychiatrie" sera effectivement extrait mais on obtiendra une UI contenant la disfluence "*c'est c'est*", ce qui ne manquerait pas de provoquer un problème au moment de passage dans le parser automatique. Une fois de plus le multi-arbre s'avère indispensable.

```
la psychiatrie
il faut avoir un don spécial parce que c' est quelque chose
il faut avoir un don spécial parce que c' est quelque chose
```

FIGURE 8 – Dépliage résultant de l'analyse de *c.*

Le dépliage obtenu figure 8 nous permet de constater que le pré-noyau “la psychiatrie” a bien été sorti des phrases générées, ce qui n’aurait pas été possible si l’information topologique avait été éliminée par la considération d’un arbre simple.

4.3 Projections

On a vu précédemment que l’arbre complet était nécessaire à l’opération de dépliage. En effet si l’on veut obtenir un tri des phénomènes à extraire provisoirement des UI et les multiples possibilités qu’offrent les piles, la concomitance de ces informations au sein du même arbre est indispensable.

Pour faire un parallèle géométrique, on peut difficilement conceptualiser un hypercube. L’opération de projection ou d’extraction consiste ainsi à diminuer le nombre de dimensions présentes dans le multi-arbre, de manière à se focaliser sur un point de vue particulier, cela revient à extraire l’arbre voulu, par la sélection des noeuds désirés.

Prenons l’exemple suivant :

a. ^alors ce que je souhaiterais faire de ma vie < c’est { devenir professeur d’italien à savoir certifié | donc "euh" enseigner { au collège | ^ainsi ^qu’au lycée } } // (échantillon M103-Corpus Rhapsodie)

Le multi-arbre obtenu est donné en figure 9 et l’arbre macro-syntaxique correspondant est donné figure 10. En revanche si l’on souhaite étudier uniquement les phénomènes d’entassement, on peut obtenir un arbre d’entassement par l’extraction des nœuds d’entassement uniquement, donnée figure 11. Les arbres projetés macro-syntaxiques et d’entassement sont obtenus comme leur nom l’indique par des projections des nœuds voulus sur le multi-arbre. Admettons que l’on veuille l’arbre d’entassement, il suffit de n’afficher que les nœuds relatifs à l’information d’entassement etc...

```
|=> root:
    |=> iu:
        |=> ru:
            |=> intro: alors, ainsi qu'
        |=> ru:
            |=> prekernel: ce que je souhaiterais faire de ma vie
        |=> ru:
            |=> kernel: c'est [...]
            |=> pile:
                |=> layer:
                    |=> kernel: devenir professeur d'italien à savoir certifié
                |=> layer:
                    |=> kernel: donc enseigner
                    |=> pile:
                        |=> layer:
                            |=> kernel: au collège
                        |=> layer:
                            |=> kernel: au lycée
        |=> ru:
            |=> discursivemarker: euh
```

FIGURE 9 – Multi-arbre résultant de l’analyse de *a.*

```
|=> root:
    |=> iu:
        |=> ru:
            |=> intro: alors, ainsi qu'
        |=> ru:
            |=> prekernel: ce que je souhaiterais faire de ma vie
        |=> ru:
            |=> kernel: c'est devenir professeur d'italien à savoir
                                certifié, donc enseigner, au collège, au lycée
        |=> ru:
            |=> discursivemarker: euh
```

FIGURE 10 – Arbre projeté de macro-syntaxe résultant de l'analyse de *a*.

```
|=> root:
    |=> pile:
        |=> layer: devenir professeur d'italien à savoir certifié
        |=> layer: euh donc enseigner
            |=> pile:
                |=> layer: au collège
                |=> layer: ainsi qu'au lycée
```

FIGURE 11 – Arbre projeté des entassements résultant de l'analyse de *a*.

Ces arbres projetés ont été obtenus par l'application d'un algorithme de regroupement des nœuds sur l'arbre complet. Pour des raisons de place, on ne rentrera pas dans les détails de cet algorithme ici [3].

4.4 Repliage des résultats du parser automatique

Nous avons vu que la phase de dépliage visait à simplifier le parsage par un analyseur automatique. Pour chaque dépliage possible d'une UI, une analyse automatique produit un ensemble de traits syntaxiques et un système de dépendance. Le *repliage* consiste à repercuter tous ces traits et liens de dépendance sur la transcription originale. Cette phase complexe est rendue possible par l'identification des éléments communs aux différents dépliages et par le fait que les données sont représentés comme des objets, pouvant avoir plusieurs attributs et liens entre eux.

Pour chaque lexème, on obtient ainsi autant de rôles syntaxiques et de liens de dépendance que le nombre d'UI dépliées dans lesquelles ce lexème apparait. L'avantage de cette phase est qu'elle permet de désambiguiser l'analyse syntaxique de certains lexèmes, par exemple en choisissant pour trait syntaxique (genre, nb etc...) de chaque mot celui qui apparait le plus de fois ou, en cas d'égalité, de choisir le dernier (critère de proximité). Ainsi, dans :

a. *"j'ai acheté { un | une } baguette //"*

le lexème baguette reçoit le trait *féminin,* malgré les segments dépliés contradictoires envoyés au parser automatique :

```
j'ai acheté un baguette
j'ai acheté une baguette
```

Une propriété intéressante de l'approche par dépliage/repliage est qu'elle permet — avec seulement des modifications mineures — de considérer plusieurs annotations différentes du même texte. En effet, chaque annotation différente produirait son propre lot de dépliages à analyser. Le repliage permettrait alors de rendre compte des ces différentes annotation et rendrait possible une plus grande robustesse en cas d'ambiguités dans les choix des annotateurs.

3. l'algorithme est consultable dans le rapport technique (Beliao et Liutkus, 2012).

Pour finir, la phase de repliage permet de visualiser l'ensemble des traits et dépendances ainsi construits directement sur la transcription originale.

4.5 Conversion en formats structurés

L'un des objectifs du traitement du balisage est l'obtention de données structurées. On cherche à générer à partir du corpus annoté l'ensemble des arbres topologiques et des arbres d'entassement possibles. Pour ce faire, l'équipe de recherche a opté pour une structure XML, ce format sert de format d'import-export pour le corpus annoté et la base SQL du projet. À terme, tous les résultats des différentes phases d'annotations syntaxiques du corpus sont donc appelés à être chargés dans une base de données SQL. Le tables relationnelles de la base sont enrichies à partir de ces fichiers XML. Le format XML des différentes phase d'annotaion sert également d'input au logiciel Vakyartha-Arborator (Gerdes, 2012) qui permet à l'équipe des syntacticiens de procéder à une phase de vérification et correction manuelle après les phases de projection, dépliage et repliage des données annotées.

La problématique qui se pose est donc de convertir des structures de données obtenues vers des fichiers structurés XML. La représentation du balisage sous la forme d'arbre permet d'effectuer cette tâche de manière triviale à partir du multi-arbre. Pour l'opération de projection de la micro ou de la macro-syntaxe, des algorithmes récursifs très simples permettent de convertir un arbre en format structuré de type XML.

Pour l'exemple a. on aura la représentation de la figure 12 pour la représentation topologique et la figure 13 pour la représentation de l'entassement.

a. ^ qui (donc) reste { toute seule | fort étonnée } // (échantillon M002-Corpus Rhapsodie)

```
<constree const_type="topology" id="a">
    <const type="iu">
        <const type="intro">
            <const const_type="lexeme" id="qui"/>
        </const>
        <const type="inkernel">
            <const const_type="lexeme" id="donc"/>
        </const>
        <const type="kernel">
            <const const_type="lexeme" id="reste"/>
            <const const_type="lexeme" id="toute"/>
            <const const_type="lexeme" id="seule"/>
            <const const_type="lexeme" id="fort"/>
            <const const_type="lexeme" id="étonnée"/>
        </const>
    </const>
</constree>
```

FIGURE 12 – Arbre topologique résultant de l'analyse de *a.*

```
<constree const_type="pile" id="a">
    <const type="pile">
        <const type="layer">
            <const const_type="lexeme" id="toute"/>
            <const const_type="lexeme" id="seule'"/>
        </const>
        <const type="layer">
            <const const_type="lexeme" id="fort"/>
            <const const_type="lexeme" id="étonnée"/>
        </const>
    </const>
</constree>
```

FIGURE 13 – Arbre d'entassement résultant de l'analyse de *a.*

Après le passage des UI dépliées dans l'analyseur automatique on procède au repliage des UIs dépliées et on obtient —après modification de certains traits et ajout de certains liens— un arbre de dépendance au format XML. Pour l'UI de exemple a. nous obtiendrons alors l'arbre de dépendance XML de la figure 14.

a. les fêtes y sont { plus | plus } nombreuses //

```
<dependency id="dep33" markupIU="les␣fêtes␣y␣sont␣{␣plus␣|␣plus␣}␣nombreuses␣//">
    <link depid="plus" func="dep" govid="nombreuses" id="func402"/>
    <link depid="plus" func="dep" govid="nombreuses" id="func403"/>
    <link depid="les" func="dep" govid="fêtes" id="func404"/>
    <link depid="fêtes" func="sub" govid="sont" id="func405"/>
    <link depid="y" func="ad" govid="sont" id="func406"/>
    <link depid="sont" func="root" id="func407"/>
    <link depid="nombreuses" func="pred" govid="sont" id="func408"/>
</dependency>
```

FIGURE 14 – Arbre de dépendance obtenu après repliage des UI dépliées résultant de a.

Le multi-arbre n'est pas généré en format XML dans le cadre du projet, il n'est utilisé que comme structure relais permettant l'ensemble des traitements nécessaires à la réalisation des tâches de dépliage et de projection.

5 Conclusion

Dans cette présentation on a proposé une systématisation informatique du système d'annotation du corpus Rhapsodie pour son exploitation par un parser FRMG. Cette proposition allie une représentation sous forme d'arbre, adaptée au formalisme souhaité, et les différents algorithmes permettant de mettre en œuvre cette proposition. Dans cette étude, nous nous sommes concentrés sur la présentation de cette représentation et sur les différents traitements qu'elle permet. La présentation des traitements informatiques correspondants fait l'objet d'un rapport technique indépendant.

Il a été vu que l'implémentation dans un multi-arbre de la totalité de l'information encodée dans le balisage est nécessaire pour certains traitements, tels que le dépliage, montrant qu'une exploitation conjointe des niveaux micro et macro-syntaxiques est parfois nécessaire. Ce multi-arbre peut aisément être projeté — ou particularisé — pour ne plus inclure qu'un sous ensemble des informations qu'il contient.

Un grand nombre de points évoqués dans cette étude peuvent faire l'objet de travaux ultérieurs. Tout d'abord, il est possible d'étendre la présente étude au cas où plusieurs annotations sont

disponibles pour la même transcription. Ensuite, il ne semble pas que la notion d'arbre, limitée au cas où chaque noeud n'a qu'un seul père, permette de rendre compte de tous les liens de dépendance envisageables. Un graphe, plus général, pourrait être plus adéquat dans ce but.

Références

Beliao, J. et Liutkus, A. (2012). Rapport technique provisoire des algorithmes utilisés pour le parsing d'un corpus de francais oral annoté. Rapport technique, HAL : halshs-00682283 version 1.

Benzitoun, C., Dister, A., Gerdes, K., Kahane, S. et Marlet, R. (2009). annoter du des textes tu te demandes si c'est syntaxique tu vois. *The 28th Conference on Lexis and Grammar*, Arena Romanistica 4, Presses de l'Université de Bergen:16–27.

Benzitoun, C., Dister, A., Gerdes, K., Kahane, S., Pietrandrea, P. et Sabio, F. (2010). Tu veux couper là faut dire pourquoi. propositions pour une segmentation syntaxique du francais parlé. *Actes du Congrès Mondial de Linguistique Francaise*, La Nouvelle Orléans.

Berrendonner, A. (2002). Morpho-syntaxe, pragma-syntaxe et ambivalences sémantiques. *Andersen, N. Nolke (éds). Macro-syntaxe et macro-sémantique. Actes du colloque d'Aarhus*, pages 23–41.

Blanche-Benveniste, C. (1990). Un modèle d'analyse syntaxique 'en grilles' pour les productions orales. *Anuario de Psicologia Liliane Tolchinsky (coord.) Barcelona*, vol. 47:11–28.

Blanche-Benveniste, C. (1997). Approches de la langue parlée en francais. *Paris : Ophrys*.

Blanche-Benveniste, C., Bilger, M., Rouget, C. et van den Eynd, K. (1990). Le français parlé. études grammaticales. *Paris, CNRS Éditions.*

Cresti, E. (2000). Corpus di italiano parlato. *Florence, Accademia della Crusca.*

de la Clergerie, E., Sagot, B., Nicolas, L. et Guénot, M.-L. (2009). Frmg : évolutions d'un analyseur syntaxique tag du francais. *11th International Conference on Parsing Technologies (IWPT'09)*.

Deulofeu, J. (1999). *Recherches sur les formes de la prédication dans les énoncés assertifs en français contemporain (le cas des énoncés introduits par le morphème que)*. Thèse de doctorat, Université Paris 3.

Gerdes, K. (2012). Arborator : A tool for collaborative dependency annotation. *http ://arborator.ilpga.fr/vakyartha/*.

Gerdes, K. et Kahane, S. (2009). Speaking in piles : Paradigmatic annotation of french spoken corpus. *Proceedings of the Fifth Corpus Linguistics Conference, Liverpool.*

Kahane, S. (2012). De l'analyse en grille à la modélisation des entassements. *(à paraitre) Hommage à Claire Blanche-Benveniste, Presses de l'université de Provence.*, in S. Caddeo, M.-N. Roubaud, M. Rouquier, F. Sabio éds.

Kahane, S. et Pietrandrea, P. (2012). Typologie des entassements en francais. *In Actes de la conférence Linx*.

Lacheret-Dujour, A., Kahane, S., Pietrandrea, P., Avanzi, M. et Victorri, B. (2011). Oui mais elle est où la coupure, là ? Quand syntaxe et prosodie s'entraident ou se complètent. *Langue française, Paris-Larousse*, 170:61–80.

Rhapsodie (2012). Site du projet rhapsodie, corpus prosodique de référence en francais parlé. *http ://rhapsodie.risc.cnrs.fr*.

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Librairie C. Klincksieck.