Leçons de l'annotation linguistique en dépendances de corpus écrits du français

Bruno Guillaume, Guy Perrier

Leçons de l'annotation linguistique en dépendances de corpus écrits du français

Bruno Guillaume and Guy Perrier

Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

Résumé

Nous proposons ici quelques éléments de réflexion sur l'annotation linguistique de corpus, notamment en syntaxe et en sémantique. En s'appuyant sur notre expérience dans des projets d'annotation de corpus français, nous listons quelques-unes des difficultés pratiques qui sont récurrentes dans les différents projets. Nous présentons enfin des outils, basés sur la réécriture de graphes, qui permettent d'automatiser en partie les conversions entre différents formats de représentation linguistiques et qui peuvent être mis en œuvre pour le développement de corpus ou pour leur exploitation.

1 Introduction

Le fait de disposer de corpus contenant des informations linguistiques est nécessaire dans de nombreuses tâches de linguistique et de traitement automatique des langues (TAL). Depuis longtemps, les lexicographes utilisent des corpus de textes pour décrire et attester les différents sens des unités lexicales ; les grammairiens les utilisent pour recenser et décrire les constructions de la langue. Avec les algorithmes récents exploitant l'apprentissage, de nouveaux besoins de corpus sont apparus. En effet, ces algorithmes ont besoin de larges ensembles d'exemples associés à des annotations linguistiques. Un corpus est d'autant plus utile qu'il est représentatif de la langue ce qui implique en général plusieurs milliers ou dizaines de milliers de phrases. Ces corpus annotés sont également aujourd'hui beaucoup employés pour évaluer les outils ou les méthodes proposées en TAL ; on parle de corpus de référence que l'on suppose correctement annotés et le résultat produit par les outils est comparé à ces corpus de référence.

Une tâche d'annotation présuppose toujours la définition d'un format et d'un guide précis. La problématique de la réalisation d'un corpus par des annotateurs humains et les questions relatives notamment à la mesure de la qualité des annotations produites a été abondamment étudié. On pourra consulter l'ouvrage [For16] à ce sujet. Ici, nous allons nous concentrer sur les questions relatives aux choix des formats et à la présentation de méthodes et d'outils qui facilitent le travail avec ces différents formats.

Même si nous ne considérons ici que les corpus écrits, on est nécessairement confronté à une multitude de choix possibles pour les formats d'annotations. Un premier choix à faire est celui du niveau linguistique visé. Pour citer les niveaux les plus couramment rencontrés, un corpus peut être annoté en catégories

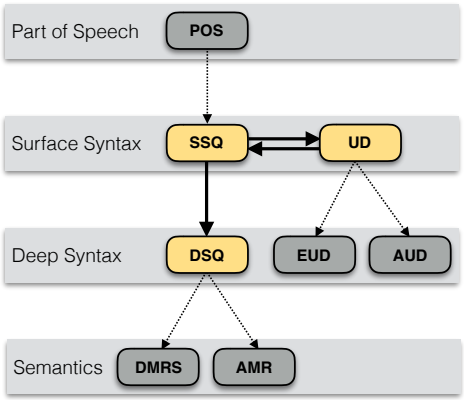


FIGURE 1 : Les formats linguistiques considérés

grammaticales, en syntaxe, en sémantique, en structure du discours... Ensuite, pour un niveau donné, plusieurs théories linguistiques existent dans la littérature. En syntaxe par exemple, deux traditions principales cohabitent : l'analyse syntagmatique et l'analyse en dépendances ; en sémantique ou en structure du discours, l'éventail des propositions théoriques est encore beaucoup plus large. Pour finir, même avec un cadre linguistique théorique fixé, de nombreuses options sont encore à choisir avant de pouvoir annoter en pratique un corpus. Par exemple, l'annotation en catégories grammaticales implique le choix d'un ensemble d'étiquettes et d'une définition de ces étiquettes qui va permettre en pratique d'annoter les données. On décrira plus loin d'autres exemples de choix à effectuer pour l'annotation en syntaxe de dépendances.

Ici nous nous intéressons uniquement à la syntaxe mais tout en gardant en tête la sémantique. Et c'est bien l'annotation en sémantique qui motive la *syntaxe profonde* que nous allons décrire. La syntaxe profonde est une abstraction de la syntaxe, telle qu'on la connaît habituellement et qui sera appelée *syntaxe de surface*. Nous nous placerons ici dans le cadre de la syntaxe en dépendances. L'annotation en dépendances est plus proche de la sémantique. Elle permet notamment de retrouver plus directement les relations prédicat argument. Par ailleurs, les structures syntaxiques en dépendances sont de plus en plus utilisées car elles permettent de construire de outils automatiques efficaces. Depuis quelques années, le projet *Universal Dependencies*[1] (UD) propose un cadre général dans lequel de nombreuses langues sont annotées (une centaine de corpus dans plus de 60 langues). Ce cadre, tout en restant au niveau de la syntaxe, prend en compte des aspects sémantiques.

Le livre [BGP18] traite en détail de ces questions. La figure 1 décrit les différents niveaux linguistiques et les conversions (internes à un niveau ou entre niveaux) qui sont considérés dans ce livre et en trait plein celles qui le seront ici.

Les trois formats qui seront utilisés dans la suite sont :

[1] http://universaldependencies.org/

- SSQ : c'est le format de surface utilisé pour le corpus Sequoia. Il a été mis au point dans le cadre de l'annotation en dépendances du corpus French TreeBank (FTB) [CS12] qui a suivi l'annotation en constituants du même corpus [ACT03].
- DSQ : c'est l'annotation en syntaxe profonde associé à SSQ [CPG+14].
- UD : c'est le format utilisé dans l'annotation de corpus en français dans le projet Universal Dependencies.

Les autres formats du schéma de la figure 1 (qui ne seront pas détaillés ici) sont : POS (l'annotation en partie du discours, suivant les catégories utilisées dans l'annotation du FTB) ; EUD (Enhanced UD qui est une proposition de représentation en syntaxe profonde du projet UD [SM16]) ; AUD (une autre proposition de syntaxe profonde inspirée de DSQ [CGPS17]) ; DMRS (Dependency Minimal Recursion Semantics [CFPS05], une représentation sémantique qui modélise, entre autres, les portées des quantificateurs et de certains adverbes) et AMR (Abstract Meaning Representation [BBC+13], un autre formalisme sémantique qui met le focus sur la représentation des relations prédicats-arguments).

Tous ces formats sont ceux que nous avons utilisé dans le travail d'annotation que nous avons mené sur des corpus du français. Pour ce qui est du niveau sémantique, nous avons choisi les formalismes de la DMRS et de l' AMR parce que les annotations se prêtent facilement à une représentation sous forme de graphes. Cela ne veut en aucun cas dire que ces formalismes seraient meilleurs que d'autres pour capturer la sémantique des langues.

Dans la section 2, nous détaillons quelques-unes des difficultés pratiques qui se posent lors de l'annotation de grand corpus. La section 3 décrit plus en détail les 3 formats considérés dans l'article et les problèmes de conversion entre ces formats. Enfin, la section 4 décrit quelques outils que nous proposons pour faciliter les tâches d'annotation et de maintenance de corpus annotés.

2 Difficultés linguistiques pour l'annotation de grands corpus

À partir de notre expérience dans les deux projets Sequoia et UD, nous donnons ici un aperçu de quelques difficultés d'ordre linguistique qui se posent pour annoter des corpus. Nous n'abordons pas ici la segmentation en phrases (qui peut également poser des difficultés) et nous supposerons que les phrases se présentent isolément.

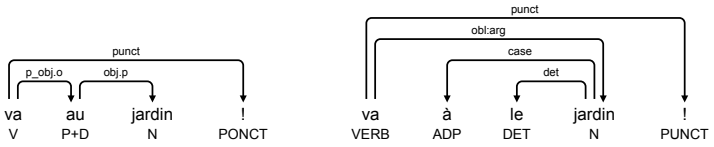
2.1 Segmentation en mots

Une première difficulté dans l'annotation syntaxique d'un corpus brut provient de la segmentation en mots. Si on peut toujours considérer qu'un espace indique une séparation entre deux mots et que les signes de ponctuation sont autonomes, le traitement des tirets (on pourrait vouloir un seul mot pour *porte-monnaie* et deux pour *vient-il*) et des apostrophes (un mot pour *aujourd'hui* et deux pour *l'image*) est beaucoup moins clair. Les deux formats qui nous intéressent font cette distinction et annotent les tirets ou les apostrophes en fonction de

leur usage, ce qui est plus naturel mais qui impose que la segmentation d'une nouvelle phrase dépend d'information lexicale. Il arrive que de l'information syntaxique soit nécessaire : par exemple, pour distinguer les deux usages de *rendez-vous* comme nom ou comme verbe suivi d'un pronom. Une solution qui permettrait de faire cette étape sans information linguistique serait de faire une segmentation maximale en tokens en utilisant les espaces et la ponctuation. Par exemple, *aujourd'hui* se décomposerait en 3 tokens : *aujourd*, *'* et *hui*. Ensuite, ce serait des relations de dépendances qui coderaient le fait que ces 3 tokens ne forment qu'un seul mot du point de vue linguistique.

2.2 Amalgame

En général, un mot correspond à une entrée dans un dictionnaire ou *lexème*. Il se trouve que certains mots correspondent à plusieurs lexèmes, quand ils résultent de l'amalgame d'autres mots. Un exemple typique en français est la forme *au* qui est le résultat d'un amalgame *à+le*. Dès lors, comment le considérer dans l'analyse ? SEQUOIA propose une catégorie grammaticale spécifique `P+D` qui rend bien compte du double rôle de *au* mais il n'est pas possible de rendre compte du double rôle au niveau des relations de dépendances sans déroger à la structure d'arbre ; *au* est donc considéré comme une préposition à ce niveau. En revanche, UD étant destiné à un usage multilingue, il n'est pas souhaitable d'introduire de nouvelles catégories spécifiques à une langue ; le choix a donc été fait d'introduire explicitement les deux mots *à* et *le* dans ce cas. Les structures syntaxiques de la phrase *va au jardin !* sont données dans la figure ci-dessous (au format SEQUOIA à gauche et UD à droite) :



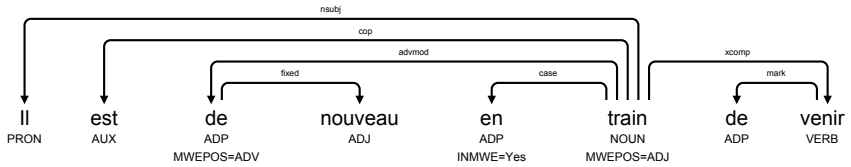
À noter que même lorsqu'une décision est prise pour *au*, d'autres cas sont plus délicats à annoter car ambigus. Par exemple *des* est un amalgame *de+les* dans *Max parle des enfants* et un simple déterminant dans *Max voit des enfants*.

2.3 Expressions multi-mots

Les expressions multi-mots (que l'on notera de l'acronyme anglais MWE pour *Multi-Word Expression*) sont des expressions qui ne respectent pas le principe de compositionnalité du sens. Elles posent des problèmes d'annotation car on peut les voir sous deux angles différents, soit comme s'il s'agissait d'un seul mot, soit comme un composé de plusieurs mots. Les formats d'annotation prévoient que chaque mot a une seule catégorie et un seul rôle dans la phrase. Dès lors, il est difficile de rendre compte des deux aspects d'une MWE.

Prenons l'exemple de la MWE *à peine*. Elle se comporte comme un adverbe mais elle composée d'une préposition suivie d'un nom. Pour rendre compte de ces deux aspects, une solution serait d'ajouter dans l'annotation aux deux nœuds représentant *à* et *peine*, un nœud représentant *à peine* mais cela sort du cadre des formats actuels qui interdisent l'ajout de ce type de nœuds. Il n'y a pas actuellement de consigne dans le projet UD pour annoter ces constructions.

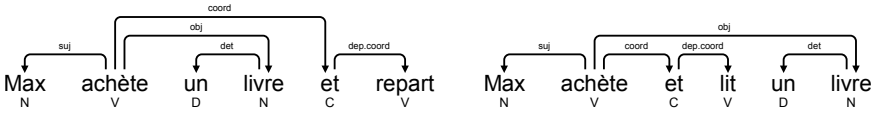
Pour le français, il a été proposé d'utiliser des traits pour indiquer ce double rôle. Dans l'exemple ci-dessous, le trait `MWEPOS` indique la catégorie de la MWE et il est attaché à sa tête.



On peut remarquer sur cet exemple qu'il y a deux types d'annotation pour les MWE : d'une part *de nouveau* qui n'a pas de structure syntaxique interne claire, qui est donc annotée avec la relation `fixed` et d'autre part *en train* dont l'annotation interne est régulière (avec la relation `case`) mais qui se comporte comme un adjectif du point de vue externe. Dans le premier cas, les relations `fixed` sont issues du mot le plus à gauche de la MWE vers les autres composantes de celles-ci. Dans le second cas, un trait `INMWE` indique les mots qui font partie de la MWE. La deuxième annotation n'est pas prévue dans le projet UD mais elle contient plus d'information et peut donc être traduite automatiquement dans le format attendu par UD (comme l'annotation de *de nouveau*).

2.4 Coordination

La coordination est également une source de nombreuses questions lors de l'annotation en syntaxe. On trouve une bonne synthèse de ces questions dans [GK15]. Dans les deux formats Sequoia et UD, le premier conjoint est la tête de la coordination. On ne peut alors plus distinguer la structure syntaxique d'une phrase où un élément dépend du premier conjoint de celle où il dépend de toute la coordination. Ainsi dans les deux phrases ci-dessous, les structures de graphe sont identiques et la relation `obj` relie *achète* et *livre* dans les deux cas, bien que *livre* soit objet de *achète* uniquement dans la première et de la coordination des deux verbes dans la seconde.



Les problèmes sont encore plus épineux lorsque la coordination vient avec une ellipse. Dans la phrase *Max mange une banane et Léa une pomme*, les syntagmes *Léa* et *une pomme* sont respectivement sujet et objet d'un verbe qui n'est pas présent dans la phrase. Si on écarte la possibilité d'ajouter un nœud fantôme pour rendre compte de cette ellipse, toutes les solutions envisageables introduisent des irrégularités dans les annotations. En plus, la plupart des formats d'annotation exigent que les structures syntaxiques soient des arbres, ce qui exclut par exemple les solutions qui explicitent le parallélisme entre éléments des deux conjoints [GK15].

2.5 Continuum entre les phénomènes

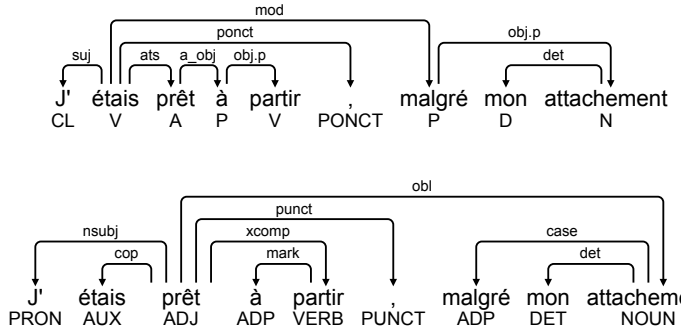
Un dernier exemple de problème récurrent en annotation syntaxique est la gestion du continuum entre différentes analyses possibles. Un cas typique est la distinction entre les arguments du verbe et les autres compléments qui ne font que modifier le verbe. Il faut se donner des critères précis tout en sachant qu'il n'est pas possible de trancher définitivement dans certains cas. De fait, en français, les compléments des noms sont encore plus difficiles à classer, si bien que dans les corpus existants, cette distinction n'est pas faite.

3 Les formats d'annotations syntaxiques

Indépendamment des difficultés évoquées plus haut, même pour des phrases relativement simples, il n'y a pas de façon canonique d'annoter la structure syntaxique et de nombreuses variantes sont possibles. Nous allons présenter ici, à titre d'exemple, deux formats de syntaxe de surface en montrant leurs différences et en étudiant comment il est possible de convertir un format dans l'autre. Nous présenterons ensuite un autre niveau syntaxique (appelé syntaxe profonde) dont le but principal et de proposer, toujours avec des relations syntaxiques entre unités lexicales, une représentation plus proche de la sémantique de la phrase. En syntaxe profonde, il n'y a notamment plus de contrainte d'arbre et une unité lexicale peut être la cible d'aucune relation (auquel cas on considère qu'elle ne fait plus partie de la structure), d'une relation (comme en surface) ou de plusieurs relations.

3.1 La syntaxe de surface

Les deux figures ci-dessous montrent les annotations dans les deux formats de syntaxe de surface Sequoia et UDde la phrase (1).



(1) J'étais prêt à partir, malgré mon attachement `[annodis.er_00474]`

Les principaux choix à opérer sont de deux types :

- Pour chaque syntagme, le choix de la tête détermine le mot qui sera la racine du sous-arbre de dépendance associé à ce syntagme. Dans les deux formats que nous considérons, des choix opposés ont été faits à ce sujet. En format Sequoia, les têtes sont les mots fonctionnels, alors que pour UD, ce sont les mots lexicaux. Par exemple, pour le syntagme prépositionnel

malgré mon attachement, la préposition *malgré* est la tête pour SEQUOIA et *attachement* est la tête selon UD. De même, la tête du syntagme *J'étais prêt à partir* est *étais* selon SEQUOIA et *prêt* selon UD.

- Le second choix porte sur le jeu d'étiquettes utilisé pour typer les dépendances. Les étiquettes désignent les fonctions syntaxiques (sujet, objet, modificateur...) mais celles-ci peuvent être définies plus ou moins finement. Un sujet est noté `suj` en SEQUOIA alors que UD précise par la relation `nsubj` le fait que le sujet est nominal (une autre relation `csubj` existe pour les sujets phrastiques). Pour le complément prépositionnel *à partir*, SEQUOIA note simplement `a_obj` (pour complément prépositionnel introduite par *à*), alors que UD utilise une relation `xcomp` qui précise à la fois que le complément est phrastique et qu'il y a un contrôle du sujet.

Ce n'est pas nécessairement un problème d'avoir différents formats d'annotation syntaxique si l'on peut passer de l'un à l'autre par une procédure automatique. Malheureusement, une annotation syntaxique dans un format ne contient pas forcément la même information que dans un autre format. Si elle est plus riche, il ne sera pas difficile de la convertir dans l'autre format. Par contre, si elle est plus pauvre, la conversion ne pourra être que partielle. Pour la compléter, on pourra parfois avoir recours à des lexiques qui fourniront les informations manquantes mais ce ne sera pas toujours suffisant et si la conversion reste ambigüe, elle devra être complétée par une annotation manuelle.

Pour convertir automatiquement l'annotation SEQUOIA en une annotation UD, dans le cas de l'exemple précédent, il faut changer la tête de la phrase de *étais* à *prêt* en déplaçant toutes les dépendances qui s'y rattachent. Il faut changer la tête des syntagmes prépositionnels, c'est-à-dire inverser les dépendances `obj.p` et changer les étiquettes en `case` si la tête du syntagme est un nom ou en `mark` si c'est un verbe ; il faut également déplacer les dépendances qui arrivent sur les prépositions. Tout ceci va pouvoir se faire automatiquement. La seule transformation qui nécessitera une information lexicale est le remplacement de l'étiquette `a_obj` par `xcomp` : un lexique indiquera que l'adjectif *prêt* régit un complément à l'infinitif introduit par la préposition *à* et dont le sujet est contrôlé par l'adjectif.

Dans le sens UD vers SEQUOIA la conversion est plus problématique. Il s'agit tout d'abord de transférer la tête de la phrase de *prêt* vers *étais* et la dépendance `cop` est remplacée par une dépendance `ats` en sens inverse. La difficulté est de savoir parmi les quatre autres dépendances issues de *prêt* lesquelles doivent être transférées sur *étais*. Pour les compléments, c'est au cas par cas qu'il faut regarder s'il faut déplacer le gouverneur de l'attribut du sujet vers la copule.

- Les dépendances `nsubj` et `punct` ne pose pas de problème : leur gouverneur est déplacé et leur étiquette remplacée.

- Pour *à partir*, on doit avoir recours à un lexique qui indique que l'adjectif *prêt* régit un complément à l'infinitif introduit par la préposition *à*. Cela implique qu'il n'y a pas à changer le gouverneur de la dépendance `xcomp` mais seulement son étiquette en `a_obj`.

- Pour le complément *malgré mon attachement*, aucun lexique n'a d'entrée pour *prêt* indiquant que l'adjectif régit un complément nominal introduit

par *malgré*, ce qui implique que ce complément est un modificateur. Mais la question est de savoir si c'est un modificateur de l'adjectif ou de la phrase. C'est seulement dans le second cas qu'il faudra déplacer le gouverneur vers la copule. Une transformation automatique ne sera donc pas correcte dans certains cas et une annotation manuelle est nécessaire.

Pour finir, il reste le changement de tête des syntagmes prépositionnels qui peut être fait automatiquement sans problème.

3.2 La syntaxe profonde, porte d'entrée vers la sémantique

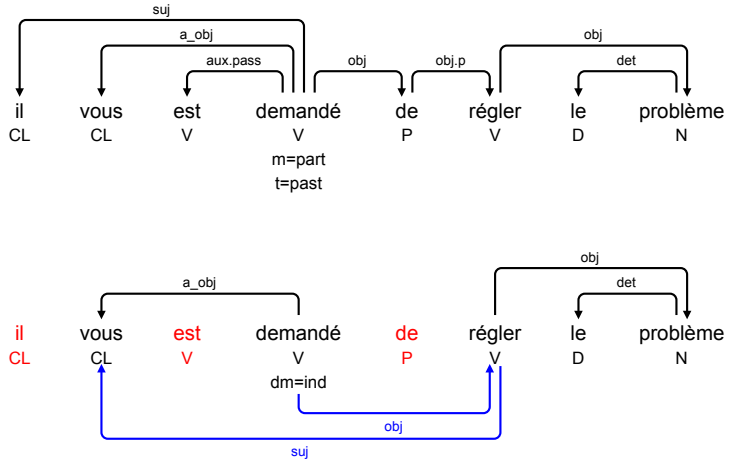
La formalisation de la sémantique de la phrase est beaucoup plus complexe que celle de la syntaxe, surtout si on veut prendre en compte tous ses aspects (typage fin des relations prédicats-arguments, portée des quantificateurs et des adverbes, temporalité et intensionnalité...). De nombreux formats ont été proposés ; chacun de ces formats modélisant une partie de ces différents aspects.

Dans notre approche, l'annotation sémantique d'un corpus ne se construit pas ex nihilo mais par transformation d'une annotation syntaxique de surface. Si on veut le faire de façon automatique, il n'est pas réaliste de concevoir une procédure spécifique pour chaque format sémantique. Un moyen de factoriser une partie du travail nécessaire consiste à utiliser un niveau intermédiaire commun, la *syntaxe profonde*. Plusieurs théories linguistiques, qui se situent dans le cadre des grammaires de dépendance, distinguent en syntaxe deux niveaux : la *syntaxe de surface* de la *syntaxe profonde* [SHP86, Mel88]. La syntaxe de surface est proche de la forme phonologique de la phrase en ce sens qu'elle fait intervenir tous les mots en prenant en compte leur ordre, et quand on parle de syntaxe sans préciser, c'est elle que l'on sous-entend. La syntaxe profonde est une abstraction de la syntaxe de surface vers la sémantique.

Avec Candito et Seddah [PCG+14], nous avons proposé une notion de *syntaxe profonde* qui s'inscrit dans le cadre du format d'annotation du corpus Sequoia. Dans cette proposition, la syntaxe profonde en dépendances présente trois caractéristiques :

- Elle ne considère que les mots lexicaux (noms, verbes, adjectifs et adverbes) et les mots grammaticaux qui ont un contenu sémantique propre : les pronoms, certaines prépositions et conjonctions.
- Elle exprime tous les arguments et modificateurs des mots lexicaux sous forme de dépendances.
- Elle neutralise les différentes diathèses du verbe et les constructions qui topicalisent ou mettent l'emphase sur un des éléments de la phrase, pour se ramener à une forme canonique, en général la construction sujet-verbe à la voix active.

La figure ci-dessous décrit les structures syntaxiques dans le format Sequoia de surface (que nous noterons SSQ par la suite comme dans la figure 1) puis dans le format Sequoia profond (que nous noterons DSQ).



(2) il vous est demandé de régler le problème `[annodis.er_00196]`

Certains mots grammaticaux ne font pas partie de la structure en syntaxe profonde : le pronom impersonnel *il*, l'auxiliaire du passif *est* et la préposition *de*. On en conserve néanmoins la trace en les faisant apparaître en rouge pour faciliter la lecture des figures.

On distingue les dépendances qui apparaissent uniquement dans la structure profonde de celles qui sont aussi présentes dans la structure de surface en représentant les premières en bleu et les secondes en noir. Ainsi, les sujets des infinitifs n'apparaissent pas en syntaxe de surface, mais lorsqu'ils sont présents dans la phrase, ils sont indiqués en syntaxe profonde. C'est le cas de *vous* qui est sujet profond de l'infinitif *régler*.

En surface, la dépendance `aux.pass` issue de *demandé* indique une construction passive impersonnelle. En syntaxe profonde, cette diathèse est neutralisée et la construction du verbe à l'actif est rétablie. C'est la syntaxe de la phrase « □ *vous demande de régler le problème* » qui est exprimée. Le seul changement que cela implique en termes de dépendance est que le sujet impersonnel est effacé. Pour garder suffisamment d'information en vue des transformations ultérieures, un trait morphosyntaxique `dm=ind` a été ajouté au verbe *demandé*. Il indique que le mode du verbe dans la structure profonde (*deep mood* en anglais, d'où le nom du trait `dm`) est l'indicatif.

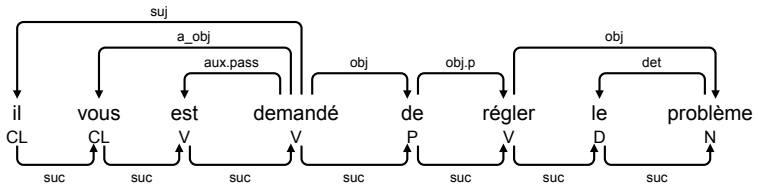
4 Outils pour le développement et l'exploitation des corpus annotés

Dans les formats de syntaxe de surface, il est requis que l'ensemble des relations forment un arbre : un des nœuds de la structure est la tête et n'a aucun gouverneur, tous les autres nœuds ont chacun exactement un gouverneur.

En revanche, en syntaxe profonde, cette contrainte n'existe pas et on a une structure de graphe dans lequel le même nœud peut être la cible de plusieurs relations. L'annotation en syntaxe profonde de l'exemple 2 présente une telle

structure : *vous* est la cible de deux relations. De plus, l'ensemble des relations peut contenir des cycles comme l'annotation en syntaxe profonde selon le format DSQ de l'expression *la personne attendue*, pour laquelle il y a une dépendance `mod` de *personne* vers *attendue* et une dépendance inverse `obj`. Au niveau sémantique également, les structures forment souvent des graphes.

En fait, même au niveau de la surface, la structure est plus riche que l'arbre des relations syntaxiques car les nœuds sont ordonnés. Une façon de rendre explicite cet ordre est d'utiliser une relation de précédence `suc`, ainsi l'annotation de surface de l'exemple 2 (en surface) devient :



Enfin, pour exprimer les interactions entre les différents niveaux de la langue, il peut être utile de disposer d'un même cadre pour les représenter [KL15].

Toutes ces constatations font que les graphes sont un cadre formel uniforme adapté pour représenter les différents niveaux d'annotations d'une phrase. Les graphes sont des structures de données qui sont bien étudiées mathématiquement. Les transformations présentées dans la figure 1 s'apparentent donc à des transformations de graphes. Pour décrire ces transformations de graphes, nous utilisons la réécriture de graphes. Un système de réécriture de graphes est un ensemble de règles qui décrivent des transformations élémentaires et qui sont appliquées successivement pour réaliser une transformation plus globale. Chaque règle est formée d'une partie gauche qui décrit un motif qui doit être remplacé dans un graphe et d'une partie droite qui indique par quoi ce motif doit être remplacé. La réécriture de graphes est un modèle adapté pour décrire ces transformations :

- Elle est suffisamment expressive pour décrire n'importe quelle transformation qui pourrait être décrite par un programme.
- Elle permet de décomposer une transformation globale en une suite de transformations élémentaires, chacune ne faisant intervenir qu'un petit nombre de nœuds. Le fait de considérer des transformations élémentaires facilite le développement et la maintenance des systèmes de transformation. Cette décomposition permet enfin de formaliser des connaissances linguistiques fines : un phénomène linguistique est en général un phénomène local qui se traduit directement par une règle de réécriture.
- Elle peut naturellement décrire des transformations non déterministes et permet donc de rendre compte d'ambiguïtés dans les conversions.

Le fait d'utiliser les structures de graphes et le modèle de calcul de réécriture de graphes nous permet de réaliser les opérations suivantes :

- recherche d'un motif donné dans un ensemble de graphes à la fois pour corriger l'annotation d'un corpus et pour exploiter une annotation donnée à des fins linguistiques,

- transformation d'un graphe en un autre graphe afin de convertir les annotations entre les différents formats qui nous intéressent.

Ces deux opérations sont en pratique très liées car les transformations utilisent des règles formées d'une paire (motif, transformation élémentaire) et l'application d'une telle règle nécessite la recherche du motif correspondant. La recherche de motifs est donc commune aux deux opérations. Nous avons développé un outil GREW[2] qui met en œuvre ces opérations.

4.1 Recherche de motifs dans un corpus annoté

La recherche de motifs peut être utilisée pour détecter les erreurs ou les incohérences que les corpus sont susceptibles de comporter. Elle peut également être utilisée pour faire l'étude d'un phénomène linguistique sur un corpus comme dans l'exemple détaillé plus loin.

Un motif est lui-même un graphe qui représente une construction linguistique et sa recherche consiste à identifier toutes les occurrences où le motif coïncide avec un sous-graphe d'un graphe du corpus. L'outil GREW-MATCH[3] permet ce type de recherche et le résultat est une liste des phrases où le motif a été trouvé. L'affichage met également en évidence la partie du graphe où le motif a été trouvé.

Dans la suite, nous utiliserons le corpus SEQUOIA (version 8.3) annoté en syntaxe de surface selon le format SSQ. Notre exemple consiste à étudier les dépendances à distance dans les propositions relatives [HPP+02, Bér12]. Le chemin de dépendances qui va de l'antécédent du pronom relatif jusqu'au pronom lui-même, en passant par la tête de la relative peut être plus ou moins long. Nous allons étudier ces différents types de chemins dans le corpus SEQUOIA en écrivant une série de motifs.

Notre démarche consiste à appliquer itérativement l'identification d'une construction qui caractérise une partie des solutions et la recherche des cas qui ne font pas partie de cette construction. À chaque étape, on ajoute des conditions qui permettent d'écarter les solutions qu'on ne souhaite pas retenir pour une analyse plus approfondie. On peut ainsi spécifier pas à pas les solutions que l'on souhaite conserver.

Étape 1 : toutes les relatives. La relation `mod.rel` est utilisée dans le format SEQUOIA pour relier l'antécédent d'une proposition relative à la tête de celle-ci. Le motif suivant permet donc de lister toutes les constructions avec subordonnées relatives :

```
pattern { ANT -[mod.rel]-> REL_HEAD }
```

Les identifiants `ANT` et `REL_HEAD` sont utilisés pour nommer les nœuds source (l'antécédent d'un pronom relatif) et cible (la tête de la relative) pour pouvoir y faire référence par la suite.

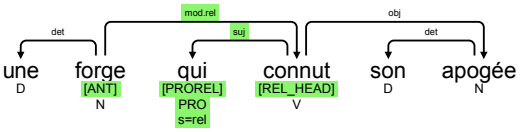
Cette recherche produit 539 solutions qui vont être ensuite classées.

[2] http://grew.fr
[3] http://match.grew.fr

Étape 2 : dépendance directe vers le pronom relatif. Pour éliminer les cas où le pronom relatif est un dépendant direct de la tête de la relative, on utilise un filtre (introduit par le mot-clé `without`) qui va écarter les cas suivants : ceux qui contiennent un nœud `PROREL` qui est un pronom relatif[4] et qui dépend directement de la tête `REL_HEAD`.

```
pattern { ANT -[mod.rel]-> REL_HEAD }
without { PROREL [cat=PRO, s=rel]; REL_HEAD -> PROREL }
```

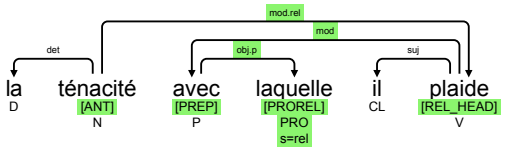
Ce motif est trouvé 83 fois ; on en déduit par différence que 456 des 539 occurrences présentent un lien direct vers le pronom relatif. La figure ci-dessous est l'un de ces 456 exemples.



Étape 3 : pronom relatif enchâssé dans un groupe prépositionnel. Pour distinguer ces cas parmi les 83 occurrences, on applique un filtre supplémentaire qui écarte les constructions où la tête de la relative `REL_HEAD` est reliée à son pronom relatif via une préposition `PREP`. Pour cela les lignes suivantes sont ajoutées au motif précédent :

```
without {
  REL_HEAD -> PREP; PREP [cat=P];
  PREP -[obj.p]-> PROREL; PROREL [cat=PRO, s=rel];
}
```

Avec ce troisième motif, 53 occurrences sont trouvées dans le corpus et il y a donc 30 (*i.e.* $83 - 53$) cas repérés à cette étape dont un exemple est donné ci-dessous :

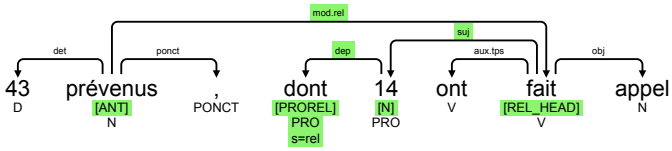


Étape 4 : chemin de longueur 3 via un nom ou pronom. Parmi les 53 occurrences, on peut distinguer les cas où le pronom relatif est complément d'un nom ou d'un pronom dépendant de la tête de la relative. Pour écarter ces cas et explorer d'autres exemples, on ajoute le filtre suivant où le nœud `N` joue l'intermédiaire entre le pronom relatif et la tête de la relative :

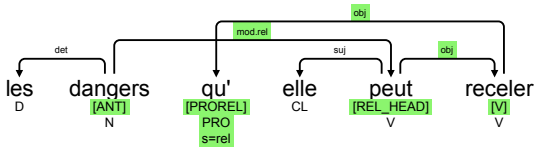
```
without {
  REL_HEAD -> N; N[upos=N|PRO|A]; N -[dep]-> PROREL;
  PROREL [cat=PRO, s=rel];
}
```

[4] Le trait `s=rel` permet de distinguer les pronoms relatifs des autres pronoms.

Avec cette nouvelle contrainte, on ne trouve plus que 29 occurrences du motif, c'est-à-dire que 24 des relations `mod.rel` sont repérées à cette étape ; en voici un exemple :



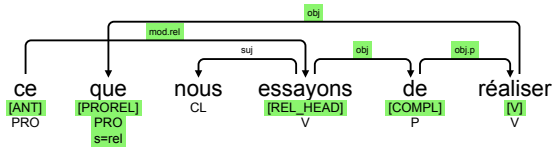
Étape 5 : chemin de longueur 3 via un verbe. Parmi les 29 restantes, il y a 8 occurrences où le pronom relatif dépend d'un verbe qui est un complément direct de la tête de la relative :



Le filtre pour les retirer :

```
without {
  REL_HEAD -> V; V [cat=V];
  V -> PROREL; PROREL [cat=PRO, s=rel];
}
```

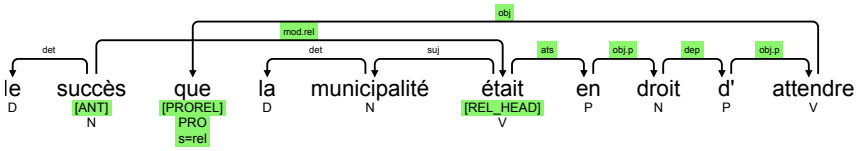
Étape 6 : chemin de longueur 4 via un verbe. Parmi les 21 solutions restantes, nous considérons les cas où le pronom relatif dépend d'un verbe qui est un complément indirect de la tête de la relative, comme dans l'exemple :



Le filtre ci-dessous introduit entre le verbe `V` qui gouverne le pronom relatif et la tête de la relative une préposition ou conjonction `COMPL`. Il permet d'identifier 8 cas de cette construction.

```
without {
  REL_HEAD -> COMPL; COMPL [cat=P|C];
  COMPL -[obj.p|obj.cpl]-> V; V [cat=V];
  V -> PROREL; PROREL [cat=PRO, s=rel];
}
```

Étape 7 : les autres cas. La recherche du motif avec les 5 filtres permet donc de trouver les 13 occurrences restantes que l'on peut explorer directement. 7 sont liées à l'annotation critiquable selon le schéma SSQ des amalgames *P+PRO* comme *auquel* ou *duquel*. Parmi les 6 autres, il y a 4 cas pour lesquels le chemin entre l'antécédent et le pronom relatif est de longueur 4 ; pour un cas la longueur est 5 et pour le dernier cas elle est de 6 dépendances (figure ci-dessous).



L'exemple sur les propositions relatives que nous venons de décrire montre comment affiner une recherche par filtrages successifs. Cette méthode s'applique évidemment à n'importe quelle construction qui peut être décrite par un motif. Sa mise œuvre est facile car l'outil est disponible en ligne sans installation par l'utilisateur.

4.2 Transformation d'annotations par réécriture de graphes

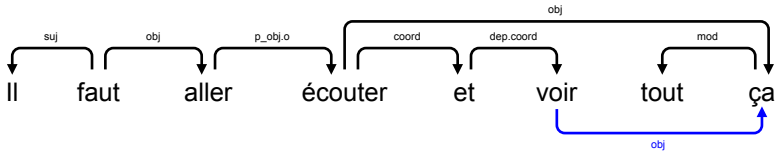
Il s'agit cette fois non seulement de reconnaître des motifs dans des annotations mais de remplacer les motifs reconnus par de nouvelles annotations. Le but peut-être de corriger des annotations ou de changer de format d'annotation.

La reconnaissance d'un motif dans un graphe peut être vue comme une fonction qui projette les éléments du motif sur le graphe en conservant la structure. Une telle fonction est une notion mathématique standard appelée *homomorphisme*. En revanche, il est difficile de décrire comment effectuer le recollement de la partie droite de la règle (ce qui doit remplacer le motif) avec le contexte et il n'existe pas de définition mathématique canonique de cette opération. Nous avons donc choisi de définir un système spécifique (appelée Grew) qui soit au plus près des besoins spécifiques au traitement des langues. Dans ce système, la partie droite est définie sous une forme opérationnelle comme une suite de commandes élémentaires (suppression ou ajout d'un nœud ou d'une arête, modification d'une étiquette...).

Avec Grew, un système de réécriture de graphes (ou GRS pour Graph Rewriting System) est donc composé d'un ensemble de règles. En pratique, le nombre de règles peut être important (quelques centaines pour les systèmes les plus évolués) et il est nécessaire de décrire comment ses règles doivent s'enchaîner. Ainsi, un GRS contient également la description d'une stratégie d'application des règles.

À titre d'exemple, considérons le système de réécriture que nous avons conçu pour transformer une annotation syntaxique de surface selon le format SSQ en une annotation syntaxique profonde selon le format DSQ. Ce système comporte 260 règles et nous allons illustrer l'une d'elles sur un exemple.

Dans la phrase *Il faut aller écouter et voir tout ça*, le syntagme *tout ça* (dont la tête est *ça*) est l'objet direct des deux verbes coordonnés et donc relié au premier élément de cette coordination (par convention dans l'annotation Sequoia). Dans la syntaxe profonde, on veut rendre compte de toutes les relations entre les mots sémantiquement pleins de la phrase ; les arguments sont donc distribués sur les différents éléments de la coordination pour rendre compte du fait que le syntagme *tout ça* est également objet du verbe *voir*. La transformation correspondante consiste donc à ajouter le lien bleu de la figure ci-dessous.



(3) Il faut aller écouter et voir tout ça. [annodis.er_00094]

La règle qui décrit cette transformation se présente ainsi :

```
rule obj-share {
  pattern {
    CONJ1 -[obj|D:obj]-> OBJ;
    CONJ1 -[coord]-> CC; CC -[dep.coord]-> CONJ2
  }
  without { CONJ2 -[D:obj]-> OBJ }
  without { CONJ1 << OBJ; OBJ << CONJ2 }
  commands { add_edge CONJ2 -[D:obj]-> OBJ }
}
```

Elle est formée de deux parties :

- La première partie (avec une clause `pattern` et un nombre quelconque de clauses `without`) décrit le motif à rechercher dans l'annotation. Le fonctionnement de la recherche du motif est identique à ce qu'on a vu précédemment (*cf.* 4.1).
- La partie `commands` présente la suite de commandes qui décrivent les modifications souhaitées sur le graphe.

Dans la règle `obj-share` ci-dessus, les nœuds `CONJ1` et `CONJ2` représentent les deux verbes coordonnés, le nœud `CC` la conjonction de coordination et `OBJ` l'objet direct des verbes coordonnés. La tête de la coordination est le premier conjoint `CONJ1` et `OBJ` dépend de celui-ci par une relation `obj` ou `D:obj`[5].

La première clause `without` est nécessaire pour éviter que la réécriture ne boucle. Le but de la règle est d'ajouter une relation `D:obj` entre `CONJ2` et `OBJ` et donc si cette relation existe déjà, la règle ne doit pas s'appliquer.

La seconde clause `without` bloque l'application de la règle quand le nœud `OBJ` est entre les deux conjoints de la coordination[6]. En effet dans ce cas, le complément n'est pas partagé. Par exemple, dans la phrase [annodis.er_00432] *Arturo et le sorcier soignent le lion et dansent avec les habitants.*, le syntagme *le lion* n'est pas partagé par le deuxième conjoint *dansent.*

Enfin, la partie `commands` est formée d'une seule commande qui ajoute une relation `D:obj` du second conjoint vers l'objet commun. Cette règle ne contient qu'une commande, mais en pratique, un nombre arbitraire de commandes est possible ; les commandes sont alors exécutées séquentiellement.

[5] Le préfixe `D:` (pour *deep*) est utilisé pour différencier les relations profondes des relations de surface.

[6] La syntaxe `X << Y` correspond au fait que `X` précède `Y` dans l'ordre des mots de la phrase.

5 Conclusion

Nous avons présenté ici des exemples d'annotation de corpus en syntaxe de surface et en syntaxe profonde ainsi que les problématiques de conversion entre ces différents formats. Pour aller plus loin dans l'analyse linguistique, l'étape suivante est l'annotation en sémantique. Les formalismes existants pour la sémantique utilisent en général soit des graphes, soit des formules de la logique ; mais on peut faire des conversions entre ces deux représentations. On peut donc considérer que l'annotation en sémantique consiste à construire un graphe. Il est donc naturel de se poser la question de la pertinence de l'utilisation de la même approche, à savoir la réécriture de graphes, pour construire les représentations sémantiques à partir de la syntaxe profonde.

Nous avons commencé à explorer cette question pour le cas de l'AMR (*Abstract Meaning Representation*) [BBC+13]. Il n'est pas difficile de construire la structure de graphe AMR car les relations prédicats/arguments sont déjà identifiées mais la partie plus difficile est d'identifier les concepts (il faut par exemple pour chaque verbe, identifier son sens précis dans une liste prédéfinie). L'approche par règle n'est pas bien adaptée à cette dernière tâche et il faudrait sans doute utiliser des méthodes basées sur l'apprentissage pour cela.

La MRS (*Minimal Recursion Semantics*) [CFPS05] a pour originalité de représenter de façon sous-spécifiée les relations de portée entre les quantificateurs et certains adverbes. La MRS est notamment utilisée avec le formalisme HPSG pour développer des grammaires à large échelle dans différentes langues qui combinent la syntaxe et la sémantique [BFO08]. Une remarque similaire à celle décrite plus haut pour l'AMR peut être faite pour la construction de structure DMRS : une partie de la transformation est mécanique et donc modélisable par des règles mais l'identification des contraintes sur la portée des quantificateurs ou des négations demandera une autre approche.

Remerciements

Les auteurs remercient Sylvain Kahane et le relecteur anonyme pour leurs relectures attentives.

Références

[ACT03] Anne Abeillé, Lionel Clément, and François Toussenel. *Building a Treebank for French*, chapter 10. Kluwer Academic Publishers, 2003.

[BBC+13] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013.

[Bér12] Lolita Bérard. *Dépendances à distance en français contemporain - Etude sur corpus.* PhD thesis, Université de Lorraine, 2012.

[BFO08] Emily M Bender, Dan Flickinger, and Stephan Oepen. Grammar engineering for linguistic hypothesis testing. In *Proceedings of the*

Texas Linguistics Society X conference : Computational linguistics for less-studied languages, pages 16–36, 2008.

[BGP18] Guillaume Bonfante, Bruno Guillaume, and Guy Perrier. *Application of Graph Rewriting to Natural Language Processing*. John Wiley & Sons, 2018.

[CFPS05] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. Minimal recursion semantics : An introduction. *Research on Language and Computation*, 3(2-3) :281–332, 2005.

[CGPS17] Marie Candito, Bruno Guillaume, Guy Perrier, and Djamé Seddah. Enhanced UD Dependencies with Neutralized Diathesis Alternation. In *Depling 2017*, Pisa, Italy, September 2017.

[CPG+14] Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric De La Clergerie. Deep Syntax Annotation of the Sequoia French Treebank. In *International Conference on Language Resources and Evaluation (LREC)*, pages 2298–2305, Reykjavik, Islande, 2014.

[CS12] Marie Candito and Djamé Seddah. Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012*, Grenoble, France, 2012.

[For16] Karën Fort. *Collaborative Annotation for Reliable Natural Language Processing : Technical and Sociological Aspects*. John Wiley & Sons, 2016.

[GK15] Kim Gerdes and Sylvain Kahane. Non-constituent coordination and other coordinative constructions as dependency graphs. In *Depling 2015*, pages 101–110, 2015.

[HPP+02] Rodney Huddleston, Geoffrey K Pullum, Peter Peterson, et al. Relative constructions and unbounded dependencies. 2002.

[KL15] Sylvain Kahane and François Lareau. Word ordering as a graph rewriting process. In *Formal Grammar*, pages 216–239. Springer, 2015.

[Mel88] Igor Mel'čuk. *Dependency Syntax : Theory and Practice*. Albany, N.Y. : The SUNY Press, 1988.

[PCG+14] Guy Perrier, Marie Candito, Bruno Guillaume, Corentin Ribeyre, Karën Fort, and Djamé Seddah. Un schéma d'annotation en dépendances syntaxiques profondes pour le français. In *TALN*, pages 574–579, Marseille, France, 2014.

[SHP86] Petr Sgall, Eva Hajicová, and Jarmila Panevová. *The meaning of the sentence in its semantic and pragmatic aspects*. Springer Science & Business Media, 1986.

[SM16] Sebastian Schuster and Christopher D Manning. Enhanced english universal dependencies : An improved representation for natural language understanding tasks. In *LREC*, 2016.