Révision de données phonétiques et acoustiques & transformation de dépendances de surface en dépendances profondes de Stanford

Cécile Robin

Révision de données phonétiques et acoustiques

&

Transformation de dépendances de surface en dépendances profondes de Stanford

Nom : Robin
Prénom : Cécile

UFR LLASIC

Mémoire de **master professionnel - 20 crédits** Mention : **Science du langage**

Spécialité : **Industries de la langue** - Parcours : **TALEP**

Sous la direction de **Olivier Kraif**

Jury : **O. Kraif, A. Bittar, G. Antoniadis, A. Tutin**

Année universitaire 2012-2013

UNIVERSITE Stendhal GRENOBLE 3

Déclaration anti-plagiat

DECLARATION

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

Signature : 

Remerciements

Je remercie très fortement toutes les personnes qui m'ont suivie et assistée pendant ce stage :

- les membres de l'équipe de Nuance : Paolo pour ses conseils et remarques pertinentes, Daniele pour m'avoir accueilli, formé, pour avoir pris en considération mes remarques. Merci aussi pour ta patience et ton suivi même à la fin du projet.

- les membres de l'équipe Celi Italie et en particulier Bea pour son accueil chaleureux

- les membre de Ho2s, anciennement Celi France, pour m'avoir intégrée à l'entreprise comme une membre de l'équipe à part entière, en particulier à Mathieu pour sa précieuse aide pour résoudre mes problèmes , à Sigrid pour son intérêt et son retour sur mon travail, à Luca pour m'avoir guidé dans l'organisation de mes idées fumeuses, mais également pour ses faux départs, ses sonneries de téléphone venues d'un autre monde qui rythmaient la vie au bureau, et enfin à André pour son suivi, son aide précieuse, ses conseils, ses relectures, qui m'ont permis d'aller plus loin dans mon travail.

- Merci à Olivier Kraif pour ses réponses à mes questions, sa réactivité et son aide, même dans l'urgence.

Je remercie Spartacus le ficus, Marcus le cactus, et Stefano l'italiano pour leurs bouffées d'oxygène quotidiennes.

Un merci particulier à Kader qui, sans le savoir, m'a donné un coup de booste incroyable. Merci à toutes les personnes qui ont fait semblant d'avoir compris ce à quoi consistait mon travail.

Merci aux amis pour la bonne humeur, à la famille pour le soutien depuis le début de ma scolarité, et à toutes les personnes avec lesquelles j'ai interagit durant ces 6 derniers mois, qui ont influé d'une manière ou d'une autre sur le choix des phrases-test de mes règles.

MOTS-CLÉS : vérification phonétique, frontières phonémiques, dépendances de Stanford, analyse syntaxique, transformation de graphe

RÉSUMÉ

Ce mémoire rapporte le travail effectué sur deux tâches associées à deux projets distincts. La première consiste en la vérification manuelle de données phonétiques et acoustiques constituant la base de phonème d'un synthétiseur vocal. Après une brève description du système de synthèse, nous réalisons un classement des erreurs phonétiques générées par le module de conversion graphème-phonème, en détaillant les différentes façons de les corriger (automatique ou non). Puis nous distinguons les types d'erreurs recensés au niveau des frontières phonémiques après segmentation du son, impliquant des déplacement de barrières pour une bonne correspondance son-phonème. Dans la deuxième partie, nous décrivons l'implémentation d'un module de règles dans un système de transformation de graphe, dans le but de convertir une analyse en dépendances de surface vers les dépendances profondes de Stanford, dans une version adaptée au français. Enfin, nous réalisons une brève évaluation des règles, et proposons des solutions pour améliorer la vitesse d'exécution.

KEYWORDS : phonetic check, phonemic barriers, Stanford dependencies, parsing, graph transformation

ABSTRACT

This paper describes the realization of two different tasks as part of two projects. The first one consists of the manual checking of phonetic and acoustic data that will form the phoneme base of a speech synthesizer. After a brief study of the speech synthesis model, we review and classify the phonetic errors generated by the grapheme-to-phoneme tool, comparing the possible ways of correcting them (either automatically or manually). Then, we distinguish categories of errors implying phonemic barriers shifting after sound segmentation, which is aimed at matching the sound with the phonemes of the transcriptions. In the second section, we describe the implementation of a module of rules for a graph transformation model to convert parses from shallow typed dependencies into the deeper Stanford typed dependencies, which we adjusted to the French language. Finally, we perform an brief evaluation of the module, and suggest some improvements in order to increase processing speed.

Sommaire

Introduction

L'entreprise *Holmes Semantic Solutions* SAS (anciennement CELI France) située à Grenoble est spécialisée dans le domaine du traitement automatique de la langue, qu'elle soit écrite ou orale. Elle propose une série de web services pour l'exploitation de données textuelles sous plusieurs aspects (lemmatisation, extraction d'entités nommées, classification de document, *clustering*, analyse de sentiments, etc.) et participe activement à des projets de recherche européens en collaboration avec des centres de recherche, des entreprises et des universités. L'objectif du stage effectué au sein de cet établissement pendant six mois est la réalisation de deux tâches sur deux projets différents, l'un du domaine de la parole, et l'autre de l'écrit.

Le premier relève d'un partenariat avec l'entreprise Nuance Communications pour le développement d'un système de synthèse vocale à destination d'un client. Dans ce but, un corpus de fichiers audio a été constitué. Il contient les phonèmes qui seront choisis par le synthétiseur selon leur nature et leur contexte en relation avec la phrase à synthétiser (système de sélection de phonème). Pour cela, les transcriptions des différents fichiers de phrases qui vont constituer une base de donnée pour la synthèse se doivent d'être précises et justes. De même, il est nécessaire que les frontières phonétiques dans le fichier audio soient parfaitement ajustées au phonème qui leur est associé dans la transcription. La tâche sera donc ici d'assurer une vérification et correction manuelle de ces données, indispensables pour assurer un maximum d'exactitude.

La seconde mission s'inscrit dans le cadre du projet de recherche ANR SYNODOS (SYstème de Normalisation et d'Organisation de Données médicales textuelles pour l'Observation en Santé[1]), dont l'objectif est l'analyse et l'extraction de données sémantiques à partir de dossiers patients pour l'aide à la décision médicale. La tâche à accomplir ici concerne le prétraitement textuel pour mettre en évidence les données potentiellement pertinentes avant l'extraction sémantique. Dans ce but, une analyse

[1] http://www.synodos.fr/, dernière consultation : 08/09/13

syntaxique en relations de dépendances profondes est nécessaire, et celles de Stanford[2] conviennent aux besoins de l'entreprise. L'objectif est donc d'adapter le système de relations de dépendances de Stanford à la syntaxe du français, et de réaliser la conversion vers ces relations à partir de la sortie de l'analyse en dépendances de surface de l'outil (nommé *Holmes*) fournit par l'entreprise.

[2] http://nlp.stanford.edu/software/stanford-dependencies.shtml, dernière consultation : 08/09/13

PARTIE 1

Vérification de transcriptions et frontières phonémiques pour un système de synthèse vocale

Chapitre 1 – Introduction

Le modèle de synthétisation par sélection de phonèmes est le plus utilisé à l'heure actuelle par les systèmes de synthèse vocale. Il implique de nombreuses phases de traitement du texte et du son, ainsi que la constitution d'un corpus audio analysé et annoté d'informations sur plusieurs niveaux (linguistique, extra-linguistique), adaptées au domaine d'utilisation. Le tout constituera au final la base de phonèmes qui sera utilisée pour la génération de la voix synthétique.

Le présent travail portera sur cette première phase de la méthode consistant à appliquer des couches successives de traitements au flux audio et au texte correspondant, incluant entre autres la transcription phonétique, la segmentation au niveau des frontières phonétiques, et des informations prosodiques sur l'intonation. Les traitements sur ces données sont effectués automatiquement. Le corpus audio annoté constituant la base de la voix de synthèse, les étiquetages se doivent d'être les plus justes possibles afin que la voix obtenue corresponde à une prononciation correcte. Une vérification et une correction manuelle de ces annotations sont alors nécessaires pour obtenir un résultat précis et fiable, qui tienne compte des particularités non-prédictibles de la langue.

Nous détaillerons dans une première partie le principe général du modèle de sélection de phonèmes et en particulier, cette première phase de constitution de la base de données audio pour la synthèse. En deuxième partie, nous décrirons le processus de vérification et de correction des transcriptions, puis celui des frontières phonétiques.

Pour des questions de confidentialité, nous ne pourrons dévoiler dans le présent document aucune information concernant les noms du projet et du client, ni les outils utilisés. D'autre part, les exemples servant à illustrer les explications ont été créés de toute pièce pour ce rapport et ne sont en aucun cas directement tirés des fichiers

réellement traités. En outre, l'alphabet de transcription employé ici sera celui de l'Alphabet Phonétique International[3].

Chapitre 2 – Fonctionnement du système

Le système de synthétiseur adopté ici se fonde sur une méthode courante de sélection de phonèmes, autrement dit le son en sortie est généré par concaténation de diphones à partir d'un texte donné en entrée. Les diphones sont des paires de phonèmes contigus, c'est-à-dire des unités de parole qui ne sont pas considérées comme des entités distinctes, mais comme deux unités reliées par une transition [Dobrišek *et al*., 1999]. Plusieurs étapes sont nécessaires à l'obtention d'un pareil système, et le principe général est exposé ici.

1. La création d'une base de diphones

Pour mener à bien la concaténation des unités de son, une base contenant toutes les associations de phonèmes existantes dans la langue (ici le français) doit être constituée. En effet, les phénomènes de coarticulation sont nombreux et doivent être pris en compte pour un rendu de voix plus naturel [Mariani, 2009].

1.1 Collection d'un corpus de phrases

Pour cela, des phrases sont définies afin de recouvrir tous les diphones que nécessite le système. Elles peuvent être adaptées au domaine général, ou bien à des utilisations dans des domaines spécifiques. Une fois les phrases déterminées, elles sont prononcées par une personne native de la langue concernée et enregistrées avec une qualité de son constante sur l'ensemble des sessions d'enregistrement, afin de préserver une unicité dans le son [Mariani, 2009]. L'ensemble des fichiers audio ainsi collectés constituera la base de sons d'où seront extraits les phonèmes pour la voix de synthèse.

1.2 Prétraitements des données audio et textuelles de la base

Pour rendre ces données exploitables, un certain nombre de prétraitements est nécessaire sur les fichiers audio et les fichiers textuels.

[3] Voir Annexe 1 (p.106-107) pour la liste complète des symboles utilisés

Une segmentation au niveau des *tokens*[4] suivie le plus souvent d'une analyse morphosyntaxique permet de préparer le texte pour une transcription graphème-phonème. Le choix a été fait ici de ne pas réaliser cette dernière analyse, se fondant donc uniquement sur les frontières des mots et les graphèmes pour la transcription phonétique. Une analyse prosodique peut ensuite être appliquée pour détecter les pauses (présence d'une virgule par exemple), les marques d'intonations (la dernière syllabe du mot la plupart du temps) ou encore d'autres données extralinguistiques, mais toujours en se basant sur le texte.

La transcription est effectuée à différents niveaux (phonétique, phonologique, prosodique, etc.) à l'aide d'un module se fondant sur un système de règles appliqué au texte. Nous obtenons alors une transcription phonétique couplée à des informations diverses telles que le marquage des débuts de mots, de l'intonation, de pauses, etc. Ces éléments serviront à réduire le choix de diphones possibles pour le passage à la synthèse, et rendre le résultat plus naturel, grâce à un contexte plus précis.

Par la suite, un module de segmentation du flux audio fractionne et aligne les fragments de son aux unités phonétiques et autres unités linguistiques des transcriptions. La durée de chaque phonème est alors déterminée.

Enfin, une évaluation automatique de l'intensité de l'intonation est réalisée afin de tenir compte des variations de la voix en fonction de contextes déterminés (question, étonnement, déclaration, etc.). Ces spécificités sont importantes dans les choix adoptés pour la synthèse [Calia, 2002].

C'est sur ces trois dernières étapes (transcription phonétique, segmentation et évaluation de l'intonation) qu'il est nécessaire de vérifier manuellement (ou semi-automatiquement) des données pour constituer une base contenant le moins d'erreurs possible, et ainsi obtenir une sortie plus proche d'une voix naturelle.

[4] Un token est une unité lexicale composée d'un ou plusieurs mots (exemple : « parce que »)

2. La synthèse

2.1 Prétraitement du texte à synthétiser

Pour procéder à la synthétisation d'un document textuel, plusieurs étapes sont nécessaires. Pour pouvoir parvenir au résultat acoustique, il est indispensable de préparer le texte à la transition vers la voix. Les choix de segments acoustiques dépendent directement de cette préparation préalable. Elle consiste aux mêmes étapes que celles pour la création de la base de phonèmes annotés (transcriptions phonétique, phonologique, prosodique, etc.). Ainsi, les annotations de la base de phonèmes, et celles du texte à générer peuvent être mises en parallèle pour choisir dans le flux audio les diphones les plus pertinents par rapport au contexte.

2.2 Choix des segments de son dans la base

Toutes les informations précisées dans la transcription du texte à générer ont leur importance dans la sélection des segments audio qui constitueront le signal sonore final.

Le choix de la chaine optimale de segments de son s'effectue en se fondant sur des critères de sélection des « meilleurs » candidats. Les unités visées ici ne sont pas des unités simples mais des séquences d'unités, les diphones, puis des séquences de diphones pour obtenir un résultat lié. Pour déterminer ces candidats, les phonèmes et le contexte de chaque diphone sont examinés grâce aux annotations insérées au préalable dans la base de phonèmes et dans le texte à générer. Toutes les informations permettant de choisir un diphone particulier plutôt qu'un autre sont importantes : sa position dans la phrase (après une pause, avant une pause, en fin de phrase interrogative, etc.), dans le mot (première syllabe, dernière syllabe : variation de pitch, ...), etc. Ainsi, les diphones de la base qui sont trop éloignés du contexte sont éliminés de la liste des candidats, et seront conservés ceux dont le calcul de ressemblance est le plus fort. A la fin, parmi les candidats de chaque diphone sont choisis ceux qui produiront un chemin moins couteux à parcourir, autrement dit la séquence de diphones successifs (soit pour une suite de phonèmes abcde la séquence ab bc cd de) dont le coût d'enchainement est moindre [Mariani, 2009].

Chapitre 3 – Révision de transcriptions phonétiques

Lors de la vérification et la correction des transcriptions phonétiques, plusieurs catégories de problèmes ont été détectées et catégorisées. Certaines erreurs ont présenté des régularités et ont pu aboutir à des adaptations ou à des créations de règles de phonétisation dans le module de transcription automatique. D'autres, en revanche, ont révélé un caractère imprédictible, la génération de nouvelles règles n'étant alors pas réalisable considérant les données mises à disposition.

1. Erreurs imprédictibles

Plusieurs catégories d'erreurs se rapportent à ce cas : les prononciations pouvant varier aléatoirement indépendamment du locuteur ou de la phrase, les prononciations dépendantes d'une certaine syntaxe ou de la catégorie morphosyntaxique du terme (données non renseignées dans le présent système), ou encore les prononciations directement dépendantes du locuteur et de ses habitudes.

1.1 Variations aléatoires

Ces variations de prononciation ne dépendent ni du locuteur, ni d'une quelconque syntaxe. Un même terme dans une phrase donnée peut être prononcée de diverses manières par un locuteur, sans contrainte particulière. Ces cas sont totalement imprévisibles si l'on ne s'appuie que sur du texte, il faut donc toujours les vérifier et les corriger manuellement si nécessaire. Les principaux types de variations repérés ici sont au nombre de trois.

Parmi elles, nous avons la consonne glottale réalisée au début d'un mot commençant par une voyelle. Elle est le plus souvent présente en tout début de phrase, ou bien lorsque le mot précédent termine également par une voyelle. Mais toutes les successions de mot-terminant-par-une-voyelle et mot-commençant-par-une-voyelle n'entrainent pas l'apparition d'une consonne glottale. Dans ce dernier cas, celle-ci marque le plus souvent une mise en évidence du mot auquel elle se rattache.

Ex : Il a achet**é** **u**n œuf => [il#a#aʃəte#ʔɛ̃#nœf]

D'autre part, certaines liaisons entre deux mots sont facultatives. La réalisation phonétique du locuteur n'est donc pas prévisible en s'appuyant uniquement sur le texte à générer à l'oral. De plus, le traitement des liaisons obligatoires par le système de transcription a pu parfois entrainer des « dommages collatéraux ». C'est la raison pour laquelle la correction consiste à parfois rajouter une liaison qui est établie par le locuteur mais non transcrite, mais aussi à supprimer une liaison transcrite qui n'est en réalité pas prononcée par le locuteur.

Ex : Il avai**t** appris => [il#avɛ#**t**apʁi]
=>[il#avɛ#apʁi]

Enfin, un « schwa » [ə] peut être présent ou non, soit à l'intérieur d'un mot, soit en fin de mot. Là encore, aucun signe se fondant uniquement sur du texte ne permet d'anticiper sa présence ou son élision, et par conséquent aucune règle n'a pu être produite. Il s'agit donc soit de le rajouter, soit de le supprimer, selon la réalisation du locuteur et non pas selon le standard choisi par le module de transcription automatique.

Ex : de la menth**e** fraiche => [də#la#mɑ̃t**ə**#frɛʃ]
=> [də#la#mɑ̃t#frɛʃ]
Ex : je t'apporterai => [ʒə#tapɔrt**ə**rɛ]
=> [ʒə#tapɔrtrɛ]

Certains types d'erreurs de transcription par rapport à la prononciation peuvent par ailleurs être corrigés à l'aide de règles fondées sur des informations supplémentaires, telles que les étiquettes morphosyntaxiques. Le présent système ne donne cependant pas de telles précisions dans les annotations.

1.2 Variations prédictibles via un autre degré d'analyse

Plusieurs cas de figure ont été regroupés dans cette classe. Deux d'entre eux portent sur des variations phonétiques directement liées à la catégorie morphosyntaxique du mot concerné, alors que le troisième nécessite non seulement une analyse morphosyntaxique, mais également la détection dans la phrase d'un élément discontinu qui lui est associé.

La prononciation de termes homographes qui ne sont pas homophones peut être parfois désambiguïsée grâce à leur catégorie : nom, adjectif, et/ou verbe. Ces cas-ci nécessitent une analyse supplémentaire au niveau morphosyntaxique afin de différencier leur réalisation phonétique, et donc leur transcription. Cependant, le module de transcription ne produit ici qu'un standard fixe qu'il faut alors adapter manuellement en fonction du contexte.

Ex : influent => verbe : [ɛ̃fl**y**]

=> adjectif : [ɛ̃fl**ɥɑ̃**]

Ex : flipper => verbe : [flip**e**]

=> nom : [flip**œʁ**]

En outre, une analyse morphosyntaxique pourrait permettre de diminuer les erreurs engendrées par la règle de liaison nasale (avec ou sans dénasalisation de la voyelle nasale). Selon cette règle, la liaison ne se fait généralement pas avec les voyelles nasales devant un mot débutant par une voyelle, excepté pour quelques cas particuliers.

Parmi eux, par exemple, il est spécifié que la liaison est effectuée sans dénasalisation de la voyelle après le déterminant « un » lorsqu'il est suivi d'une voyelle [Kalmbach, 2011]. Sans analyse morphosyntaxique, le système généralisera cette règle et provoquera des erreurs de prédiction pour « un » en tant que nom. De même, l'adjectif « bon » suivi d'un mot débutant par une voyelle crée une liaison nasale, avec dénasalisation de la voyelle. Lorsqu'il est un nom, la règle générale devrait s'appliquer : pas de liaison nasale [Kalmbach, 2011]. L'absence d'analyse morphosyntaxique peut ainsi provoquer quelques erreurs. Mais il a été constaté que le nombre de cas concernés est plutôt faible sur la quantité de données traitées.

Ex : **un** arbre => déterminant : [**œ̃#n**aʁbʁ]

un ou deux => nom : [œ̃#u#dø]

Ex : un **bon** ami => adjectif : [œ̃#b**ɔ̃#n**ami]

le **bon** arrive => nom : [lə#bɔ̃#aʁiv]

Une dernière catégorie de variations n'est pas prédictible par un système de transcription automatique : celles directement contraintes par le locuteur.

1.3 Variations dépendantes du locuteur

La prononciation de certains termes et expressions ne dépend pas toujours de critères morphosyntaxiques, mais parfois seulement du locuteur lui-même, de par ses origines, ses interactions avec d'autres locuteurs, son environnement quotidien, etc. Plusieurs prononciations d'un même mot sont alors acceptées «officiellement» comme standards et la transcription doit s'adapter au locuteur. Quelques exemples permettent d'illustrer ce cas.

Ex : but => [by**t**]
=> [by]
Ex : Metz => [mɛ**t**s]
=> [mɛs]
Ex : puzzle => [pœzl]
=> [pœz**œ**l]

Enfin, nous allons étudier plus en détails les erreurs détectées lors des vérifications pour lesquelles une correction automatique est possible.

2. Erreurs corrigibles automatiquement

A l'intérieur de ce groupe rassemblant les erreurs de transcription qui peuvent donner suite à des règles de phonétisation, nous pouvons déterminer deux sous-groupes : les erreurs nécessitant une adaptation ou création de règle de phonétisation, et celles qui peuvent être corrigées via l'utilisation d'une liste comportant le mot et sa transcription.

2.1 Adaptation ou rajout de règle de phonétisation

Le module de conversion graphème-phonème traite les assimilations qui existent entre phonème de fin de mot et phonème de début de mot sur des couples en opposition de voisement. Si le dernier phonème d'un mot est une consonne voisée (selon le standard), et le suivant une consonne non-voisée, alors il y aura assimilation, et les deux phonèmes seront prononcés non-voisés. L'inverse est également couvert. Ainsi, le module traite par exemple les cas suivants :

Ex : cette boîte => [sɛt] [bwat] : [sɛd#bwat] : voisement du [t]

Ex : page suivante => [paʒ] [sɥivɑ̃t] : [paʃ#sɥivɑ̃t] : dévoisement du [ʒ]

Cependant, l'assimilation [s]+[d]->[zd] n'avait pas été pris en compte dans la première version. Une nouvelle règle a donc été rajoutée pour inclure ce voisement du [s] en présence d'un mot débutant par [d] :

Ex : pince d'or => [pɛ̃s] [d] [ɔʁ] : [pɛ̃z#d#ɔʁ] : voisement du [s]

En outre, aucun phonème /ɲ/ n'avait été prévu, le phonème /nj/ ayant été utilisé pour remplacer les cas concernés. Il a été ajouté à la liste des phonèmes et permis de transcrire le graphème « gn » car le locuteur réalisait bien le /ɲ/ dans sa prononciation.

Ex : grognon => avant : [grɔnjɔ̃] ; après : [grɔɲɔ̃]

D'autre part, quelques erreurs se sont produites autour de la transcription du graphème « ng » (comme dans « meeting »). Celui-ci se transcrit par le phonème /ŋ/ dans l'Alphabet Phonétique International. Or, ce graphème a été trouvé sous différentes représentations dans les transcriptions : /ŋ/ seul, /ŋg/, et /ŋk/ lors de la présence de phénomènes d'assimilation. Le phonème /ŋ/ a été finalement choisi comme standard pour le graphème « ng », se fondant sur le choix des transcriptions répertoriées dans Le Petit Robert.

Ex : planning => avant : [planiɲg] ; après : [planiɲ]

Ex : parking petit => avant : [plaʁkiɲk#pəti] ; après : [plaʁkiɲ#pəti]

Cependant, certaines erreurs détectées dans les transcriptions sont liées à une réalisation particulière du mot, et ne peuvent donner lieu à une généralisation, donc une règle de phonétisation.

2.2 L'utilisation de listes pour la correction

Lorsque la réalisation phonétique de certains termes ne correspond pas aux règles de phonétisation qui ont été établies, il est parfois impossible de créer de nouvelles règles

sous peine de généraliser celle-ci à d'autres termes, eux correctement transcrits, impliquant de nouvelles erreurs. Dans ce but, l'utilisation de listes inventoriant ces mots ou expressions permet de traiter ces cas particuliers pour lesquels aucune règle ne peut s'appliquer. Plusieurs catégories sont concernées.

D'une part, les mots d'origine étrangère, prononcés au plus près du standard de la langue d'origine (tout en conservant les phonèmes du français), y sont traités. Les règles de phonétisation applicables ici ne sont alors pas adaptées au français, il n'est donc pas possible de généraliser ces réalisations.

Ex : high-tech => [ajtɛk]

D'autre part, dans ces listes sont regroupées certaines entités nommées. Nous détectons ici celles qui concernent la désignation de personnes (anthroponymes) via leur nom de famille, pseudonyme, nom d'organisation, etc., mais aussi des lieux (toponymes), ainsi que des marques, produits et entreprises diverses (ergonymes) [Daille *et al.,* 2000]. Ceci inclut alors des sigles (dont la prononciation peut se faire lettre à lettre ou non), des mots composés et des mots simples, dont la prononciation est spécifique au référent qu'ils désignent, et qui peut être différente des règles génériques établies dans le module de conversion.

Ex : CROUS => [kʁus] | ABS => [abeɛs]
Ex : Stendhal => [stɛ̃dal]
Ex : Duingt => [dɥɛ̃]

Ainsi, il a été démontré que, malgré des adaptations de règles au niveau du module de conversion graphème-phonème, que ce soit via des règles ou des listes, certaines erreurs ne peuvent être corrigées que manuellement. Ces annotations seront associées à la base de phonèmes pour la voix de synthèse. Il est donc nécessaire de vérifier que celles-ci s'accordent avec la réalisation du locuteur. Plus de 12.000 séquences ont ainsi été vérifiées et corrigées. Au niveau suivant, il est également essentiel que la segmentation des phonèmes au niveau de l'audio soit bien alignée avec les transcriptions correspondantes, afin d'éviter les erreurs et de renforcer le naturel de la voix obtenue.

Chapitre 4 – Révision des frontières phonémiques

Il est important que la segmentation du son au niveau des phonèmes soit précise. Une erreur à ce niveau-là peut entrainer des anomalies dans la sortie, lors de la génération de la voix de synthèse (deux voyelles prononcées à la suite lorsqu'une seule était attendue, un silence au milieu du mot, etc.). Dans ce but, un outil de segmentation automatique du son au niveau des phonèmes (en cohérence avec les transcriptions phonétiques) est utilisé. Afin de vérifier la bonne position des frontières sans devoir visionner chacune d'entre elles, une recherche automatique de caractéristiques prédéfinies permet de traquer des anomalies-types liées à la durée, aux variations du spectre, etc. Des catégories sont ainsi déterminées pour classer et cibler les erreurs potentielles, et réduire la quantité d'information à inspecter. L'une d'entre elles regroupe les phonèmes concernés par une durée inhabituellement grande ou courte, ou encore les voyelles à énergie basse, grâce à un seuil fixé. Sont également recherchées les erreurs dues à la présence ou l'absence de frictions, ou encore à la présence de bruit ou de silence à des emplacements inattendus. Enfin, une catégorie particulière est réservée aux cas concernés par une combinaison de ces problèmes, ou qui présentent une potentielle anomalie ne correspondant à aucune des celles précédemment cités.

1. Phonèmes courts, longs ou silencieux

Dans la classe répertoriant les phonèmes de durée spécialement longue ou courte, ainsi que les voyelles anormalement silencieuses, des sous-catégories ont été distinguées pour bien cerner les différentes situations.

1.1 Voyelles trop silencieuses

Ce sous-groupe permet de se concentrer sur un type de son, la voyelle. Lorsqu'elle est trop silencieuse, c'est-à-dire que l'énergie (visible sur le spectrogramme) est faible, il peut être question d'une anomalie de voyelle mal segmentée. La recherche de cette particularité a pointé généralement peu sur de véritables erreurs, mais surtout sur les phénomènes de coarticulation entre les consonnes alvéolaires (/d/, /t/, /n/, /s/, /v/) et les voyelles (en particulier /i/ et /y/) et les semi-voyelles. La frontière entre les deux n'est en effet pas très nette, et la voyelle est presque silencieuse (Figure 1). Le même constat

s'applique dans les cas de contact semi-voyelle/voyelle (exemple : /ji/, /ij/, /uw/), où la frontière est difficile à détecter.

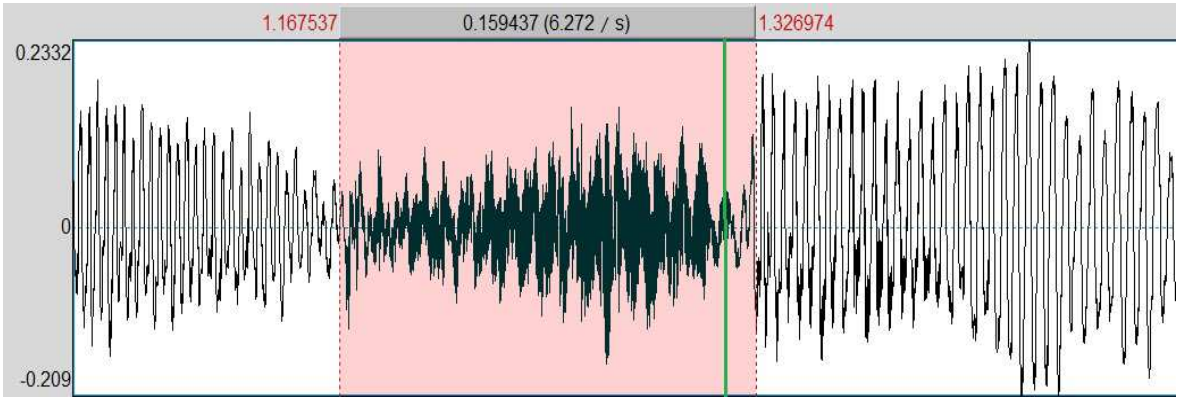


Figure 1 : *Spectrogramme de la syllabe [sj] (en rose), avec la frontière prédite par le système (simulation) en vert (outil : Praat)*

1.2 Phonèmes anormalement courts/longs

Des seuils limites sont déterminés en se fondant sur les caractéristiques de durée moyenne des phonèmes, au-delà desquels ces derniers sont considérés anormalement longs ou courts. Une vérification des frontières est alors nécessaire. Cependant, il a été démontré qu'en situation de parole spontanée, les voyelles et les consonnes ont une longueur significativement plus courte (excepté les semi-voyelles) [Pols *et al.,* 1996], ce qui est donc un phénomène attendu. Les phonèmes dont la durée passe au-dessus du seuil ont quant à eux été identifiés généralement en fin de phrase, car les voyelles sont prolongées et les consonnes enchainent sur un schwa final. Parmi les anomalies potentielles détectées, un grand nombre concernait ces cas spécifiques du langage spontané ou semi-spontané.

1.3 Voyelles comportant deux pics/sommets

Les voyelles sont caractérisées au niveau acoustique par la présence de formants, visibles sur le spectre du son sous forme de « pics » d'intensité dont l'amplitude diminue au court du temps. La présence de deux forts pics d'intensité pour une même voyelle peut mettre en évidence une erreur de segmentation entre deux voyelles qui se suivent. Le plus souvent, pourtant, cette observation portait seulement sur des voyelles de longue durée (Figure 2*)*.

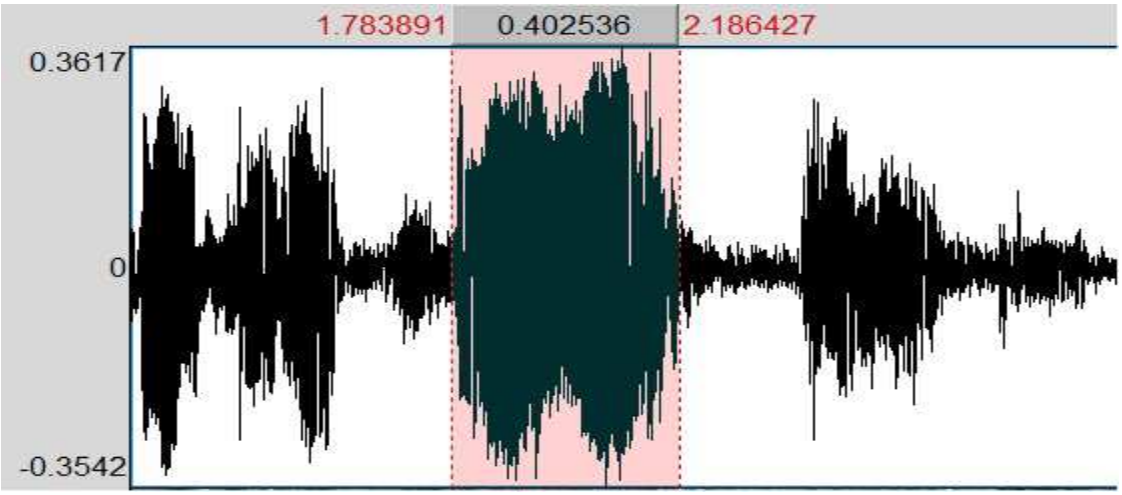


Figure 2 : *Spectrogramme de la voyelle [a:] (outil : Praat)*

2. Frictions et non frictions

Les fricatives ont des spécificités acoustiques qui ont permis de créer une catégorie d'erreurs (toujours potentielles) dédiée à ce mode d'articulation.

2.1 Absence de friction dans une fricative

Une caractéristique des consonnes fricatives, lorsqu'elles sont en contact avec une voyelle, est le phénomène de coarticulation engendré. En effet, dans le cas d'une voyelle suivie d'une fricative, la frontière entre les deux n'est pas nette, et parfois le système de segmentation automatique sépare les deux phonèmes avant la dernière partie de la voyelle. Cette imprécision de segmentation, qui implique la présence d'une partie sans friction dans la consonne fricative, est donc détectée et catégorisée dans cette classe d'anomalie (Figure 3).

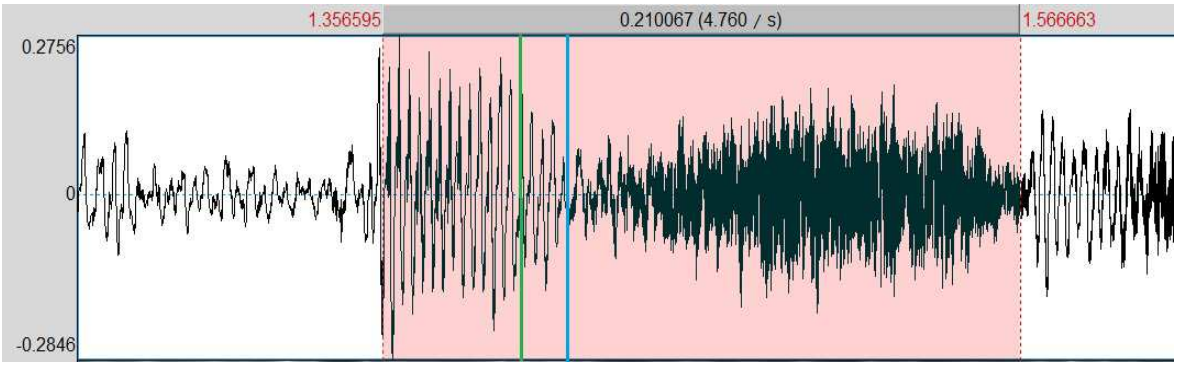


Figure 3 : *Spectrogramme de [es] (en rose), avec la frontière prédite par le système (simulation) en vert, et la correction en bleu (outil : Praat)*

2.2 Présence de friction dans une non-fricative

De même que les voyelles suivies d'une consonne fricative, le phénomène de coarticulation se produit également lorsqu'une consonne fricative est suivie d'une voyelle. Dans ce cas-ci, c'est la consonne qui s'approprie certaines caractéristiques de la voyelle et son spectre commence à prendre une légère périodicité, tout en gardant sa friction d'origine [Soli, 1981]. Ce phénomène est spécialement visible lorsque la consonne fricative est suive d'une semi-voyelle. Le système repère alors la présence de formants dans le spectre, et place la frontière phonémique avant la fin de la consonne (Figure 4).

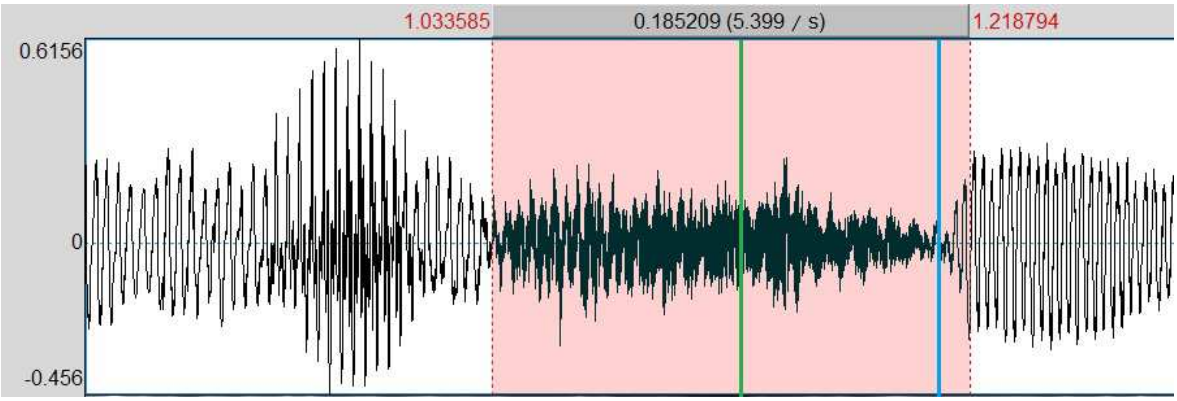


Figure 4 : *Spectrogramme de [sɥ] (en rose), avec la frontière prédite par le système (simulation) en vert, et la correction en bleu (outil : Praat)*

3. Bruits et silences

Les bruits et silences ne sont pas toujours bien détectés par le système de segmentation.

3.1 Silence dans un phonème et bruit dans du silence

La catégorie de « silence dans un phonème » détermine les frontières phonémiques mal placées lorsqu'un silence est suivi d'un phonème. Le principal cas de correction dû à cette mauvaise segmentation a lieu lorsqu'une pause sépare deux parties de phrases (virgule dans le texte), et que la deuxième partie commence par une voyelle ou une consonne fricative. Parfois, le système ne marque comme silence que le tout début de la pause, et regroupe sous le même phonème la partie principale du silence avec le son suivant.

Les phonèmes de fin de phrases font également fréquemment partie de cette classe d'erreurs, lorsque la segmentation se fait légèrement au-delà du phonème. Ces derniers ne requièrent en général que peu de corrections.

Quant au « bruit dans du silence », cette catégorie concerne tous les cas où le silence d'une pause (ou le silence de début de phrase) est associé à la fin du phonème précédent ou le début du phonème suivant. Cette recherche a généralement donné peu de résultats.

3.2 Bruit dans la phase d'occlusion d'une plosive

Les consonnes plosives sont constituées d'une partie d'occlusion, un silence avant un relâchement brutal d'énergie, le « burst ». Or, cette première partie de silence n'est pas toujours très nette, et le système place parfois la frontière sur la fin du phonème précédent (souvent une voyelle), et l'occlusive contient donc une partie de voyelle avant le début du silence (Figure 5).

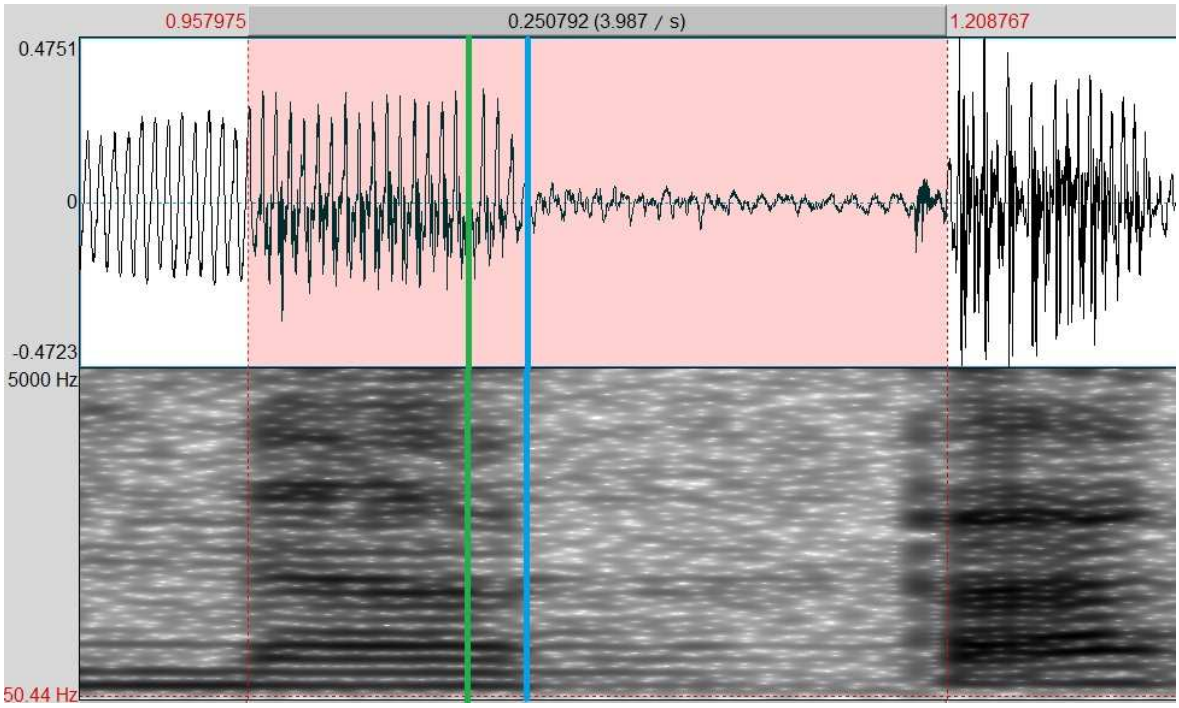


Figure 5 : *Spectrogramme de [ap] (en rose), avec la frontière prédite par le système (simulation) en vert, et la correction en bleu (outil : Praat)*

4. Problèmes combinés

Enfin, cette dernière catégorie regroupe toutes les anomalies potentielles qui, soit ne rentrent pas dans les classes précédemment définies, soit couvrent plusieurs d'entre elles. Sont principalement concernés par cette catégorie les phonèmes de fin de phrase, dont les consonnes sont suivies d'un schwa final. Celui-ci n'a pas été étiqueté au niveau phonétique car considéré comme une caractéristique de fin de phrase, donc toujours présent. D'autre part, dans la succession de voyelle-consonne en fin de phrase, la frontière entre les deux est parfois placée avant la fin de la voyelle car le phénomène de coarticulation est renforcé à ce niveau-là (Figure 6).

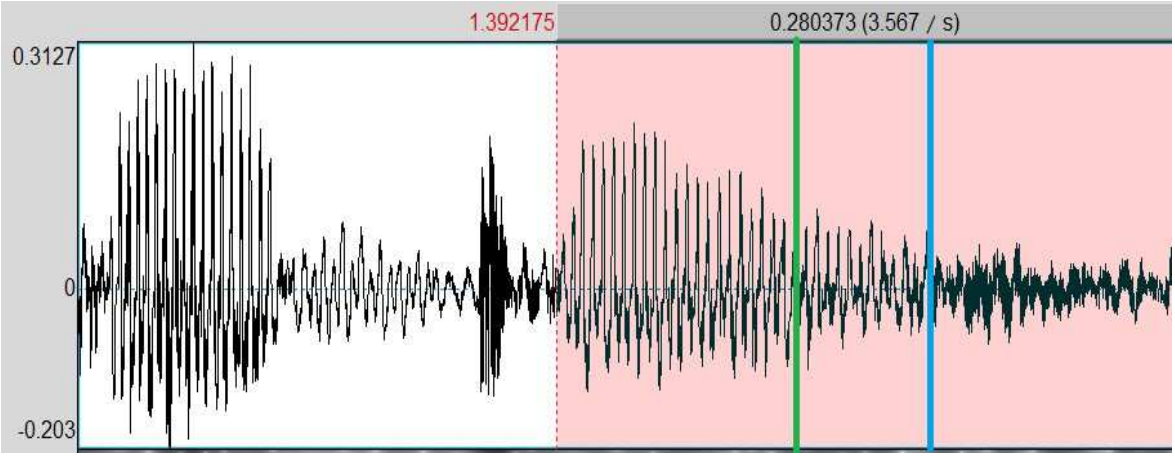


Figure 6: *Spectrogramme de [ɔʁ] (en rose), avec la frontière prédite par le système (simulation) en vert, et la correction en bleu (outil : Praat)*

Chapitre 5 – Conclusion

De nombreuses étapes constituent la création d'une voix de synthèse. Ainsi, pour chacune d'entre elles, des contrôles (manuels ou semi-automatiques) sont nécessaires à l'exploitation des données afin que celles-ci contiennent le moins d'irrégularités possible. Peu importe les choix adoptés, l'important est d'être cohérent tout au long des corrections. C'est ce qui permettra de garantir un maximum d'homogénéité dans la sortie finale.

PARTIE 2

Transposition d'une analyse en dépendances de surface vers les relations de dépendances profondes de Stanford

Chapitre 1 - Introduction

1. Contexte

Le projet SYNODOS[5] vise à exploiter les informations présentes dans les dossiers patients informatisés (DPI) pour de l'aide à la décision médicale ou la détection d'infections telles que les infections nosocomiales ou les cancers. L'objectif est de mettre à disposition une plateforme d'extraction sémantique de données médicales à partir de dossiers patients. Une recherche intelligente fondée sur un traitement linguistique profond permettra de dégager les données pertinentes avant de les organiser pour l'analyse. L'outil devra être accessible aux professionnels de la santé, non informaticiens, qui pourront créer eux-mêmes leurs règles métier pour diriger les prises de décision sur les informations extraites, en restant ainsi autonome. SYNODOS s'inscrit dans la continuité d'un ancien projet sur le domaine, TecSan ALADIN[6], qui a permis de réaliser un outil de détection de maladies nosocomiales, fournissant des résultats encourageants. Ainsi, SYNODOS permettra à terme d'aller plus loin dans l'extraction et le traitement sémantique des données des patients, en proposant un service web accessible à tout professionnel de la santé, et pouvant traiter de nombreuses catégories de maladies en fonction des règles métiers définis par ceux-ci.

2. Présentation de la tâche

Holmes Semantic Solutions (ou H2os) fournit un outil d'analyse de texte qui fonctionne grâce à une chaîne de traitements successifs (tokenization, détection de phrase, étiquetage morpho-syntaxique, analyse syntaxique en dépendances, …). Cet outil, nommé *Holmes* (*Hybrid Operable platform for Language Management and Extensible Semantics*), sera utilisé pour l'analyse et traitement des dossiers patient dans le cadre du projet SYNODOS. Cependant, l'analyseur en dépendance actuel produit des relations syntaxiques dites « de surface », c'est-à-dire qu'elles suivent l'ordre linéaire de la phrase, et ne permettent donc pas de joindre des termes liés par une fonction syntaxique cassant cet ordre. Les besoins du projet SYNODOS allant sur une analyse sémantique du texte, il est nécessaire d'avoir à disposition des relations syntaxiques

[5] http://www.synodos.fr/ dernière consultation : 08/09/13
[6] http://www.aladin-project.eu/ dernière consultation : 08/09/13

« profondes » afin d'opérer plus facilement et efficacement l'extraction des informations désirées lors du traitement sémantique.

Les relations de dépendance choisies par l'entreprise pour ce projet sont celles définies par le Stanford NLP Group[7] (les *Stanford Dependencies*, que nous nommerons *SD*), de licence libre, correspondent à une analyse syntaxique « profonde » et donc plus adaptée aux besoins du présent projet. Cependant, ces relations ne sont à l'heure actuelle disponibles que pour l'anglais et le chinois. Une mise en parallèle et adaptation de ces relations aux spécificités du français est donc dans un premier temps nécessaire pour définir le formalisme à implémenter. Dans un second temps, il s'agira de créer et mettre en place des règles de transition depuis les dépendances de surface actuelles de *Holmes* vers les dépendances profondes de Stanford afin d'obtenir un résultat suivant le modèle des SD.

Ce module de transition a été conçu en premier lieu pour le projet SYNODOS, mais son objectif à terme sera de s'intégrer à l'outil *Holmes* pour servir les projets à venir.

Dans le but de la réalisation de ce module, nous étudierons tout d'abord les principales caractéristiques de l'analyse en dépendances par rapport à l'analyse en constituants, puis nous nous intéresserons plus particulièrement aux relations de dépendances de Stanford par comparaison avec celles du système actuel de *Holmes*. Dans un second temps, nous décrirons les outils utilisés dans ce projet pour la conversion de dépendances (logiciel de transformation de graphes, et règles de transformation). Nous mettrons ensuite en confrontation le français et l'anglais au niveau des relations de dépendances afin de déterminer les différences, les problèmes rencontrés et les solutions trouvées, puis nous analyserons les règles de transformation en commentant les choix adoptés. Enfin, nous finirons par un test du système de règles, avec le calcul de la précision et du rappel et un commentaire du résultat.

[7] http://nlp.stanford.edu/software/stanford-dependencies.shtml dernière consultation : 08/09/13

Chapitre 2 - L'analyse en dépendance

Afin de bien cerner les spécificités des relations de dépendances de Stanford, nous allons tout d'abord exposer les caractéristiques des grammaires en dépendances, les comparer aux grammaires en constituants immédiats et déterminer le fonctionnement des parseurs adoptant ce système.

1. Présentation des grammaires de dépendance

1.1 Historique

La représentation en structure de dépendance a débuté très tôt. Les premières études s'en rapprochant datent du 8ème siècle avec l'apparition du principe de la hiérarchie gouverneur-gouverné en syntaxe arabe [Kahane, 2001]. C'est au début du 20ème siècle que les premiers travaux sur la théorie de grammaire en dépendances commencent à véritablement prendre de l'ampleur grâce à l'ouvrage de Tesnière [1934], qui a marqué un tournant dans l'usage de ces systèmes en Europe [Nivre, 2005]. Cependant, ce n'est que vers les années 80 que l'analyse en structure de dépendances elle-même s'est développée. En effet, le regain d'intérêt pour l'exploitation des lexiques et de la sémantique a amené les linguistes à se pencher davantage sur cette forme de représentation, plus adaptée à ces sujets. Effectivement, la représentation structurelle de dépendance s'accorde mieux à une analyse sémantique que la structure syntagmatique, ou analyse en constituants immédiats (comme nous le verrons dans l'explication détaillée de ces modèles en 1.4) [Kahane 2001]. De ce fait, elle s'adapte tout à fait aux besoins actuels du domaine du traitement automatique des langues. D'autre part, elle est parfaitement adaptée à l'analyse de langues dont l'ordre de la structure syntaxique est souple (typologie syntaxique)[8]. Par conséquent, elle permet de traiter des langues telles que le russe ou le grec [Debusmann, 2000].

[8] La typologie syntaxique d'une langue est liée à l'ordre de certains mots dans sa syntaxe : le sujet, le verbe et l'objet. Certaines langues sont donc dites de typologie SVO, SOV, etc. tandis que certaines n'ont pas d'ordre fixe.

1.2 Définition

On considère la définition de la phrase par Tesnière comme « un ensemble organisé dont les éléments constituants sont les mots. […] Les connexions structurales établissent entre les mots des rapports de dépendance. Chaque connexion unit en principe un terme supérieur à un terme inférieur. » [1959 : 2]

Dès lors, « on appelle grammaire de dépendance toute grammaire formelle qui manipule comme représentation syntaxiques des structures de dépendance. » [Kahane, 2001 : 1]

Autrement dit, une grammaire est dite de dépendance lorsque les relations entre les mots d'une phrase peuvent être modélisées par un arbre comportant plusieurs critères : un mot-racine en tête, les autres reliés directement ou indirectement à cette racine par des successions de relations de dépendances, elles-mêmes représentées par des arcs orientés (« *directed tree* ») [Kortes *et al.,* 2010]. Ces branches peuvent également être étiquetées de relations syntaxiques, on les dit alors « *typed* », et les nœuds marqués de traits (« *attributed* ») [MacCartney *et al.,* 2006].

1.3 Spécificités des grammaires de dépendance

Ce modèle met en avant les relations qui lient les mots entre eux, indépendamment de leur ordre d'apparition dans la phrase. Chaque relation est binaire et associe un mot, considéré comme le gouverneur (ou régissant) avec son dépendant (ou subordonné) qui est soit complément, soit modifieur de ce dernier. Dans une phrase, un seul mot ne dépend d'aucun autre, c'est la tête (ou la racine) de la phrase [Debusmann, 2000].

Dans les modèles de dépendance, les mots grammaticaux[9] peuvent également parfois ne pas être représentés, et un gouverneur ou un dépendant ne pourra pas composer un syntagme entier formé de plusieurs tokens. Il y aura toujours une décomposition hiérarchique de l'ensemble [Mel'čuk, 1988].

[9] Mot qui est caractérisé par sa fonction grammaticale mais qui ne transmet pas de sens par lui-même

Le graphe résultant comporte un ensemble de nœuds qui constituent les éléments lexicaux de la phrase, et un ensemble d'arcs représentant les relations de dépendances qui lient les gouverneurs aux dépendants. Plusieurs contraintes régissent cette structure selon Nivre [2005] : chaque nœud ne peut avoir qu'un père (« *single headed constraint* »), et, afin de conserver la structure en arbre, le graphe ne doit pas contenir de cycle (« *acyclicity constraint* »). Robinson [1970] insiste également sur cette condition de représentation graphique en arbre, pourtant pas toujours observée en pratique comme nous le verrons plus tard. Selon la terminologie de Tesnière [1959], ces types de représentations en dépendances sont nommés « stemma ».

Enfin, selon Mel'čuk [1988], l'arbre de dépendances doit remplir au maximum les conditions de simplicité et d'uniformité. Pour cela, les relations liant les mots doivent répondre à quatre critères : l'antisymétrie (si A est gouverneur de B, alors B ne peut pas être lui-même gouverneur de A), l'anti-réflexivité (aucun mot ne peut être complément ou modifieur de lui-même), l'anti-transitivité (si A est gouverneur de B, et B gouverneur de C, alors C ne peut pas être un dépendant direct de A), et enfin la relation de dépendance doit être étiquetée pour être dissociable des autres.

1.4 Analyse en constituant vs en dépendances (figuresFigure *7* et Figure *8*)

Même si le passage d'un formalisme à l'autre est possible, de grandes différences séparent l'analyse en constituants immédiats (ou PSG : *Phrase Structure Grammar*) de l'analyse en relations de dépendances (ou DG : *Dependency Grammar*). Alors que la PSG se focalise sur l'organisation syntagmatique de la phrase, c'est-à-dire qu'elle regroupe et relie les constituants par syntagmes et catégories grammaticales, la structure en dépendance, elle, associe directement des mots selon la relation syntaxique qui existe entre eux [Kahane, 2001]. La différence majeure se situe alors au niveau de cette absence de nœuds syntagmatique dans la DG au profit de mots [Nivre, 2005].

Au niveau de la représentation de ces modèles d'analyse, de nombreuses disparités apparaissent entre les « *dependency trees* » (DT) et les « *phrase-structure trees* » (PST). Mel'čuk [1988] en fait un examen détaillé que nous allons exposer ici.

Les PST traitent la phrase en terme de blocs syntagmatiques qui dérivent les uns des autres du plus général au plus spécifique (la plupart du temps, la phrase se sépare au

départ en deux blocs principaux : le syntagme nominal et le syntagme verbal, chacun contenant lui-même d'autres blocs, etc.). De ce fait, les PST ont une nature dite distributionnelle, l'ordre syntaxique ayant une répercussion directe sur la structure générée. Les DT, quant à eux, sont de type relationnel. Ils traitent la structure d'un groupe de mots en fonction des relations hiérarchiques qui existent entre ses éléments. Cependant, un pont est réalisable entre les deux représentations, car les constituants des PST peuvent toujours être décomposés hiérarchiquement, créant ainsi des dépendances.

Dans les PST, les relations syntagmatiques sont essentielles à la structure. En effet, les branches ne sont pas étiquetées, les feuilles (nœuds finaux) sont les mots de la phrase et tous les autres nœuds sont composés des relations syntagmatiques, plus précisément l'arbre part des constituants les plus généraux jusqu'aux plus précis (les relations morphosyntaxiques nom, adjectif, etc.). Par conséquent tous les nœuds, à l'exception des feuilles, sont non-terminaux (autrement dit divisibles en plus petites unités). Quant aux relations syntaxiques, elles ne sont pas explicitement indiquées dans l'arbre. Au contraire, pour la structure du DT, les données morphosyntaxiques ne sont pas des caractéristiques principales du modèle, mais peuvent y être rajoutées sous forme de traits sur les nœuds. L'accent est donc mis sur les relations syntaxiques, étiquetées sur les branches. Et les nœuds, contrairement au PST, sont tous terminaux (c'est-à-dire qu'ils ne peuvent pas se diviser en sous-unités).

Enfin, une certaine forme d'ordre linéaire est nécessaire au PST car il y est porteur d'information, tandis qu'il n'est pas du tout essentiel à la représentation en DT. Un arbre de dépendance n'est en effet pas toujours projectif[10], contrairement au PST.

[10] « Une structure est projective si pour chaque mot w tous les mots qui se trouvent entre w et ses subordonnés sont aussi dominés par w » [Karlov *et al*., 2012]

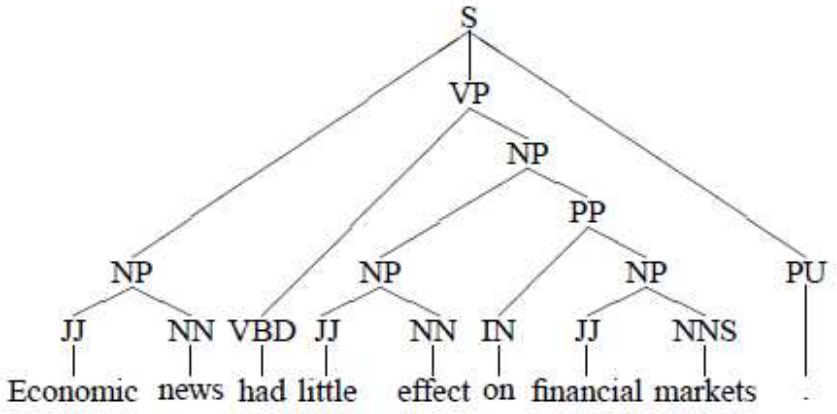


Figure 7 : *Structure en constituant pour une phrase en anglais du Penn Treebank [Nivre, 2005 : 2]*

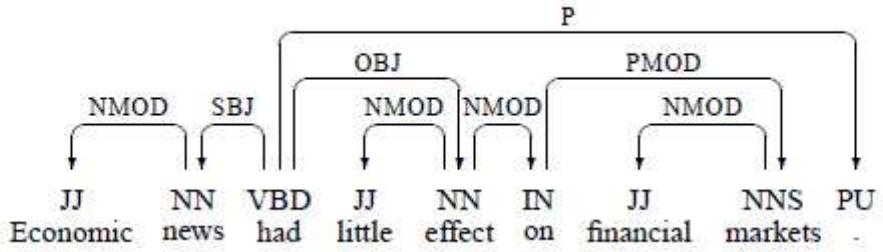


Figure 8 *: Structure en dépendances pour une phrase en anglais du Penn Treebank [Nivre, 2005 : 2]*

L'analyse sémantique met en évidence les relations entre les termes dont le sens est important pour le traitement futur souhaité. L'absence d'ordre linéaire syntaxique, les liens hiérarchiques entre les termes, les relations aux noms porteurs de sens et la structure gouverneur-dépendant (qui se rapproche des structures prédicat-argument de l'analyse sémantique) sont les caractéristiques qui font de l'analyse en dépendance la structure idéale pour un passage vers une analyse sémantique.

2. Parseurs en dépendances à base statistique

2.1 Les *treebanks*

Les parseurs statistiques fonctionnent tous sur le même modèle global : ils fondent leur apprentissage sur un ou plusieurs corpus déjà annotés manuellement sur plusieurs niveaux (morphologique, syntaxique,…). Ce type de corpus est nommé *treebank* car il constitue une « banque » d'arbres syntaxiques. Le plus connu et le plus complet

pour l'anglais est le *Penn Treebank*[11], et pour le français le *French Treebank*[12] [Abeillé *et al.* 2003]. L'objectif est ensuite de se calquer sur les phrases déjà annotées du *treebank* pour en déduire l'analyse de la phrase nouvelle, grâce à des modèles probabilistes classiques. Des probabilités sont évaluées pour chacune des analyses syntaxiques possibles de la phrase par comparaison avec le *treebank* (parmi un certain nombre de possibilités calculées), l'analyse obtenant le meilleur score étant celle renvoyée par le système [Charniak, 1997].

Les *treebanks* ont été développés à l'origine sur des modèles d'analyse en constituants. Aujourd'hui, ils s'adaptent aux nouveaux besoins, et des systèmes de conversions vers les systèmes en dépendances ont été développés. Le *French Treebank* a opéré cette transition vers des dépendances de surface, grâce à une méthode fondée sur la propriété de projectivité des arbres en constituants [Candito *et al*., 2010], et selon le schéma d'annotation défini par Candito *et al*. [2011]. Candito *et al*. [2011 : 3] expliquent ce choix de dépendances de surface (versus profondes) par la volonté de garder un « format [qui] soit un pivot convertible vers les différents standards […], et un pivot enrichissable ou convertible en dépendances plus profondes ». D'autre part, de nouveaux *treebanks* sont élaborés directement en relations de dépendances, comme le *Prague Dependency Treebank* [Böhmová *et al.*, 2001], ou le *Copenhagen Dependency Treebank* [Buch-Kromann *et al*., 2009].

2.2 Le parsing

Deux méthodes statistiques de *parsing* en dépendances sont aujourd'hui dominantes.

Le principe de la première est d'utiliser un algorithme permettant entrainer le système sur un *treebank* déjà analysé en dépendances ou qui aura été converti auparavant des constituants vers les dépendances. C'est le cas par exemple pour le français avec le *MaltParser*[13] ou le *MSTParser*[14], qui prennent en entrée des *treebanks* en dépendance (en l'occurrence, pour le français le *French Treebank*). Pour l'anglais, cependant, le *Penn Treebank* n'est pas directement disponible en arbres de dépendances.

[11] http://www.cis.upenn.edu/~treebank/ dernière consultation : 08/09/13
[12] http://www.llf.cnrs.fr/Gens/Abeille/French-Treebank-fr.php dernière consultation : 08/09/13
[13] http://www.maltparser.org/mco/french_parser/fremalt.html dernière consultation : 08/09/13
[14] http://mstparser.sourceforge.net/ dernière consultation : 08/09/13

Une méthode possible consiste à réaliser une première étape de conversion des arbres en constituants vers les dépendances, comme le modèle de Johansson et Nugues [2007], pour ensuite constituer un parseur par entrainement sur des arbres de dépendances. Il faut alors créer un modèle de conversion spécifique, adapté pour le système de dépendance souhaité.

La deuxième méthode consiste à réaliser l'apprentissage du *parser* directement sur un treebank en constituants immédiats, puis d'ajouter une étape permettant d'extraire les informations nécessaires pour constituer les relations syntaxiques de l'analyse en dépendances. C'est ce qui a été réalisé pour l'anglais avec le *Stanford Parser*[15] en dépendances.

Alors que le premier modèle nécessite seulement d'utiliser les ressources en dépendance disponibles (ou de se créer préalablement un *treebank* en dépendances en effectuant une conversion) pour constituer un *parser* opérationnel, la deuxième méthode implique une conversion en dépendance après chaque analyse en constituant du *parser*.

Cer *et al*. [2010] ont comparé cinq parseurs fondés sur le premier procédé et six sur le deuxième pour produire une sortie en dépendances de Stanford, sur l'anglais. Ils ont démontré que la méthode d'extraction à partir des constituants produit de meilleurs résultats que celle directement entrainée sur un *treebank* en dépendances. En effet, les ressources disponibles en constituants sont beaucoup plus nombreuses, l'apprentissage du *parser* est donc plus complet. Cependant, le traitement est bien plus lent : les algorithmes conçus directement pour le *parsing* en dépendance étant de 40 à 60% plus rapide, puisque le *parser* fournit directement le résultat, sans nécessiter un second traitement pour l'extraction des dépendances. Cependant, nous ne pouvons pas généraliser ces conclusions car l'évaluation a été réalisée sur de l'anglais. Le nombre et le type de ressources disponibles étant différents pour le français, les résultats peuvent diverger.

[15] http://nlp.stanford.edu/software/lex-parser.shtml dernière consultation : 08/09/13

Chapitre 3 - Stanford typed dependencies

La première version des *Stanford dependencies* (SD) a été mise à disposition en 2005. Depuis cette date et jusqu'à la dernière version de 2012, les relations ont été modifiées et améliorées dans le but de répondre au mieux au langage et aux besoins des utilisateurs [Manning *et al.,* 2012]. Voici la présentation de ce formalisme.

1. Objectif

Les SD sont des relations de dépendances élaborées par *The Stanford Natural Language Processing Group*[16] pour l'anglais, et adaptées actuellement seulement pour le chinois. Elles visent à fournir une représentation syntaxique qui se rapproche plus de la sémantique qu'une analyse en constituants ou en dépendances de surface, sous forme de relations hiérarchisées entre mots individuels. L'objectif principal de ce formalisme est de procurer une analyse épurée et exploitable par des utilisateurs non-spécialistes en linguistique (computationnelle ou non) ayant besoin d'accéder aux informations sémantiques de textes. Dans ce but, les noms des relations ont été définis spécialement pour se rapprocher des termes des grammaires traditionnelles [Manning *et al.,* 2008].

Le formalisme a été conçu pour extraire aisément l'idée générale de la phrase. Les mots porteurs de sens (les mots pleins) sont mis en évidence, et les relations sémantiques entre eux sont favorisées par rapport aux mots grammaticaux qui représentent une information moins pertinente pour l'utilisateur. Manning *et al* [2008 : 2] définissent 6 principes qui ont été suivis pour la conception du modèle :

> « *1. Everything is represented uniformly as some binary relation between two sentence words.*
>
> *2. Relations should be semantically contentful and useful to applications.*
>
> *3. Where possible, relations should use notions of traditional grammar for easier comprehension by users.*
>
> *4. Underspecified relations should be available to deal with the complexities of real text.*

[16] http://nlp.stanford.edu/software/stanford-dependencies.shtml dernière consultation : 08/09/13

5. *Where possible, relations should be between content words, not indirectly mediated via function words.*

6. *The representation should be spartan rather than overwhelming with linguistic details. »*

Aujourd'hui, ce modèle de relations est largement utilisé parmi les membres de la communauté du TAL en anglais, mais également par les professionnels du *text mining* dans le domaine biomédical [Cer *et al.,* 2010].

2. Description du formalisme

Les dépendances de Stanford étant typées, les arcs du graphe de dépendances comportent des étiquettes syntaxiques, et les relations sont donc des triplets constitués du nom de la relation, du gouverneur et du dépendant (ex : amod(meat,red)). La méthode du *Stanford parser* est, comme vue précédemment, fondée sur l'extraction de relations grammaticales à partir d'une analyse effectuée en constituants. La méthode d'extraction est composée de deux étapes : la première, l'extraction des relations, et la deuxième, l'annotation de ces relations [MacCartney, 2006].

La particularité du formalisme de Stanford, c'est l'organisation hiérarchique qui régit les relations de dépendance. La plus générique est la relation *dep* (*dependent*), de laquelle découlent toutes les autres relations, également hiérarchiquement organisées. Cette caractéristique implique une recherche de la relation sous-spécifié la plus proche de la situation lorsque le système ne parvient pas à déterminer la dépendance entre deux mots [MacCartney *et al.,* 2006]. Cela permet ainsi de s'approcher autant que possible du sens transmis par la relation, et implique un très bon rappel puisque toutes les relations sont analysées au moins en *dep.*

Le modèle de Stanford s'attache à donner une représentation plus sémantique de la phrase, c'est pourquoi il privilégie les mots pleins en tête de la construction, laissant les auxiliaires, compléments, verbes attributifs, etc. dépendre d'eux. De cette manière, la racine de la phrase n'est pas toujours un verbe, mais peut aussi bien être un adjectif, attribut [Manning *et al.,* 2012]. Les SD ne tiennent par ailleurs pas compte de la différence complément (ou argument)/modifieur qui est considérée plutôt inutile en pratique selon Manning *et al.* [2008]. Cependant, elles s'attachent à différencier les

relations internes des syntagmes nominaux (NP) et les distinguer plus finement car elles sont importantes dans les applications concrètes. Ainsi sont différenciés les « *numeric modifiers* » des « *noun compounds* » ou encore des « *abbreviations* », etc. [Manning *et al.*, 2008]

3. Formats de sortie

Les SD ont pour but de s'adapter au degré de sémantique désiré par l'utilisateur, fournissant une représentation plus ou moins profonde selon les options choisies. Ainsi, il est possible soit de conserver tous les termes du texte source en tant que nœuds, soit de sortir certains termes de la structure et les transformer en relations suivant le format souhaité [Manning *et al.,* 2008], comme nous allons l'illustrer par la suite. Le *Stanford Parser* propose cinq formats de sortie différents pour l'anglais : la représentation « *basic* », « *collapsed* », « *collapsed with propagation of conjunct dependencies*», « *collapsed preserving a tree structure* », et « *non-collapsed* » [Manning *et al.,* 2012 : 13-16].

La représentation « *basic* » utilise la majorité des dépendances qui seront citées dans la partie qui suit (Figure 9). Tous les termes sont représentés par des nœuds reliés par des relations typées, et la structure est celle d'un arbre. Elle possède donc toutes les propriétés qui découlent de ce type de structure qui est projectif.

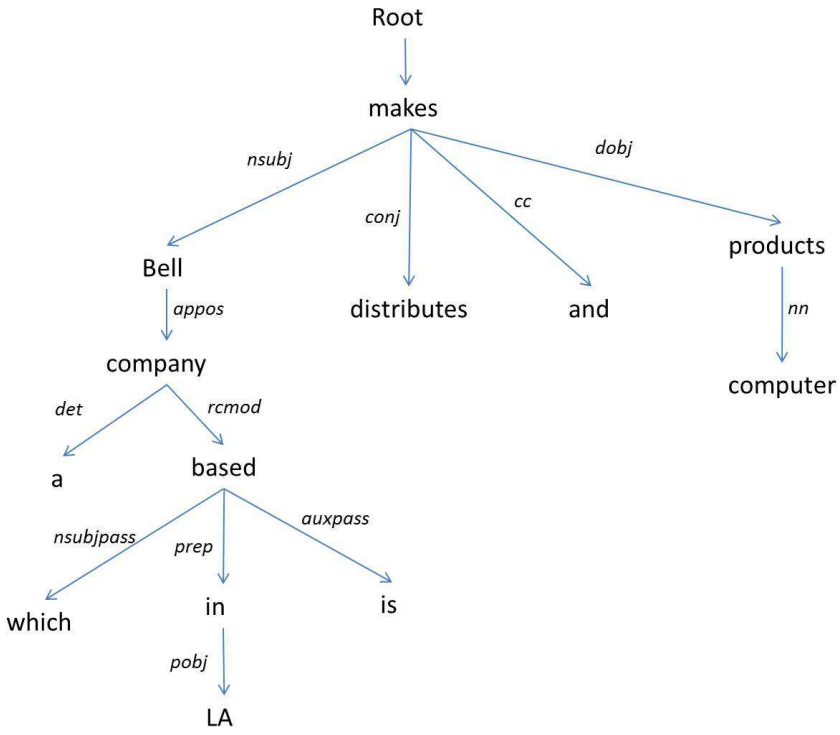


Figure 9 : *Analyse en dépendances de Stanford, format « basic » :*
« Bell, a company based in LA, makes and distributes computer products »

Le format « *collapsed* » élimine les relations impliquant certains mots grammaticaux comme les prépositions, les conjonctions ou encore les pronoms relatifs, au profit d'une relation entre mots pleins. Pour cela, le mot grammatical est ajouté dans le nom de la relation (ex : *prep_in*). De plus, la sortie est complétée de dépendances additionnelles, plus profondes, comme *ref* qui met en relation le pronom relatif et son antécédent, pour permettre d'établir des liens supplémentaires entre les mots pleins (et casser ainsi la structure en arbre) (Figure 10).

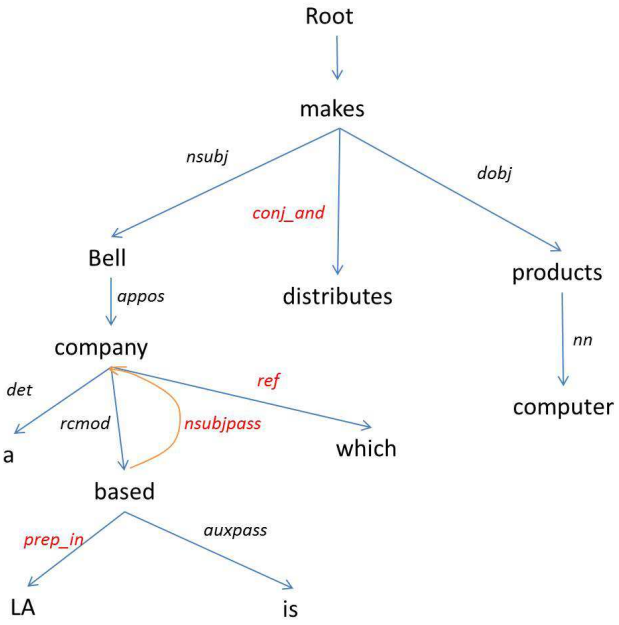


Figure 10 : *Analyse en dépendances de Stanford, format «* collapsed *» :*
« Bell, a company based in LA, makes and distributes computer products »

L'extension de ce formalisme à la « *propagation of conjunct* » permet en plus, lors d'une coordination de deux verbes dont l'objet et/ou le sujet sont dépendants des deux, de distribuer ces relations à chacun des deux verbes. La structure en arbre est alors cassée puisque ce sujet et cet objet ont chacun deux pères (Figure 11). Cependant, en pratique, pour l'instant ce mécanisme ne fonctionne que pour le cas du sujet car il est difficile de déterminer si l'objet doit se distribuer ou non.

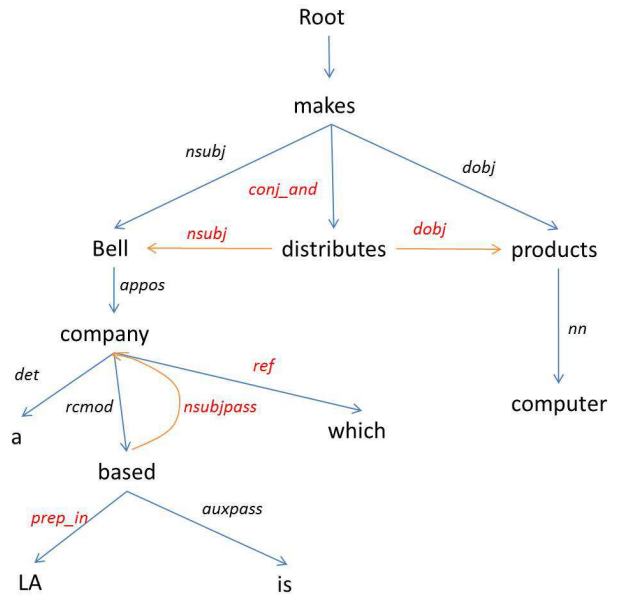


Figure 11 : *Analyse en dépendances de Stanford, format «* propagation of conjunct *» :*
« Bell, a company based in LA, makes and distributes computer products »

Les « *collapsed dependencies preserving a tree structure* » présentent toutes les caractéristiques des dépendances « *collapsed* », avec comme particularité la suppression des relations générées par le modèle « *collapsed* » qui cassent le principe d'acyclicité (Figure 12).

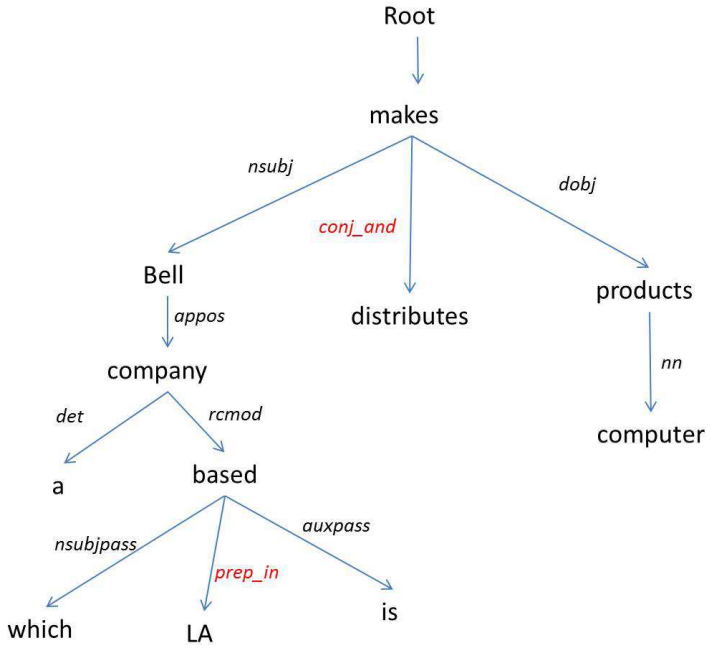


Figure 12 : *Analyse en dépendances de Stanford, format «* collapsed preserving a tree structure *» : « Bell, a company based in LA, makes and distributes computer products »*

Enfin, le modèle de « *non-collaped dependencies* » comporte toutes les relations des « *basic dependencies* » additionnées des dépendances supplémentaires (qui cassent la structure en arbre), mais sans celles dues au « *collapsing* » ou à la distribution d'arguments (Figure 13).

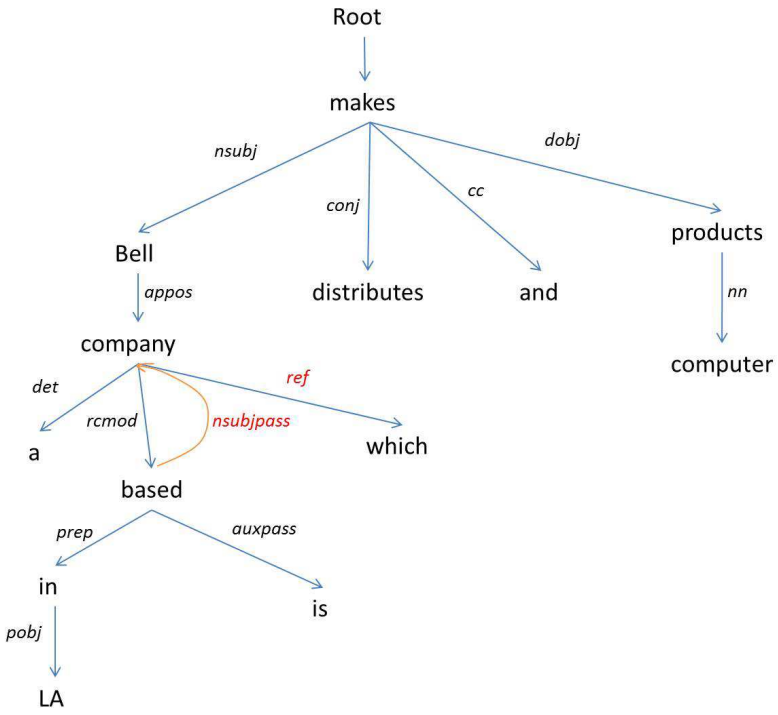


Figure 13 : *Analyse en dépendances de Stanford, format «* non-collapsed *» :*
« Bell, a company based in LA, makes and distributes computer products »

Dans le cadre du projet SYNODOS, le type de relations de dépendances adopté sera le « basic ». En effet, à ce niveau d'analyse, nous souhaitons conserver la structure en arbre car une analyse sémantiques sera ensuite réalisée. Cette étape sera donc l'intermédiaire entre la représentation de surface et la représentation sémantique. D'autre part, pour les mêmes raisons, il est encore nécessaire à ce degré d'analyse de conserver les relations entre chaque mot de la phrase, qu'ils soient mots pleins ou grammaticaux.

4. Les dépendances

La liste des dépendances actuelles de Stanford est exposée en Figure 14. En rouge sont mises en évidence les relations de dépendance supplémentaires du format « *collapsing* » (et non présentes dans le format « basic »). En outre, chacune de ces relations est expliquée en détail (exemple(s) à l'appui) par Manning *et al.* [2012] dans le manuel des dépendances de Stanford.

root - *root*
- dep - *dependent*
  - aux - *auxiliary*
    - auxpass - *passive auxiliary*
    - cop - *copula*
  - arg - *argument*
    - agent– *agent*
    - comp - *complement*
      - acomp - *adjectival complement*
      - attr – *attributive*
      - pcomp – *prepositional complement*
      - ccomp - *clausal complement with internal subject*
      - xcomp - *clausal complement with external subject*
      - complm - *complementizer*
      - obj - *object*
        - dobj - *direct object*
        - iobj - *indirect object*
        - pobj - *object of preposition*
      - mark - *marker (word introducing an advcl)*
      - rel - *relative (word introducing a rcmod)*
    - subj - *subject*
      - nsubj - *nominal subject*
        - nsubjpass - *passive nominal subject*
      - csubj - *clausal subject*
        - csubjpass - *passive clausal subject*
    - cc - *coordination*
    - conj - *conjunct*
    - expl - *expletive* (*expletive "there")*
    - mod - *modifier*
      - abbrev - *abbreviation modifier*
      - amod - *adjectival modifier*
      - appos - *appositional modifier*
      - advcl - *adverbial clause modifier*
      - purpcl - *purpose clause modifier*
      - det - *determiner*
      - predet - *predeterminer*
      - preconj - *preconjunct*
      - infmod - *infinitival modifier*
      - mwe - *multi-word expression modifier*
      - partmod - *participial modifier*
      - advmod - *adverbial modifier*
        - neg - *negation modifier*
      - rcmod - *relative clause modifier*
      - quantmod - *quantifier modifier*
      - nn - *noun compound modifier*
      - npadvmod - *noun phrase adverbial modifier*
        - tmod - *temporal modifier*
      - num - *numeric modifier*
      - number - *element of compound number*
      - prep - *prepositional modifier*
      - prepc - *prepositional clausal modifier*
      - poss - *possession modifier*
      - possessive - *possessive modifier ('s)*
      - prt - *phrasal verb particle*
    - parataxis - *parataxis*
    - punct - *punctuation*
    - ref - *referent*

sdep - *semantic dependent*
xsubj - *controlling subject*

Figure 14 : Liste des *Stanford dependencies* [Manning *et al.,* 2012 : 11-12]

Après avoir présenté le système de dépendance de Stanford, nous allons maintenant nous intéresser au système utilisé actuellement, *Holmes*.

Chapitre 4 - Le système actuel, *Holmes*

1. Présentation générale

Holmes est l'outil développé par *Ho2s* pour l'analyse de textes. Il applique sur le texte une succession de traitements, depuis le découpage en phrases jusqu'à l'analyse syntaxique, et prochainement l'analyse sémantique. La chaine de traitements est composée de phases nécessaires à l'analyse en dépendances, ainsi que des étapes supplémentaires permettant de mettre en évidence les informations pertinentes pour le domaine d'application et l'objectif du projet.

Tout d'abord, le texte est découpé en phrases, puis en *tokens*. Un étiquetage morphosyntaxique est ensuite effectué pour déterminer les parties du discours, additionnées d'attributs tels que le lemme, etc. Le *tagset* des catégories grammaticales[17] est celui conçu par Candito *et al.* [2008], composé des noms des catégories principales du *French Treebank* modifiés pour prendre en compte certaines caractéristiques des sous-catégories de ce dernier (par exemple « V » se décompose en « VINF », « VPP », etc.). Puis, des traitements spécifiques sont réalisés avant de passer au *parsing* en dépendances, si le domaine et l'objectif du projet les nécessitent. Ainsi, des « *gazetteers* » sont constitués. Ce sont des ressources spécifiant des caractéristiques propres à certains termes de par leur nature (auxiliaires, verbes modaux), leurs traits sémantiques (temporel, etc.) ou autre spécification à apporter pour un traitement futur optimisé. D'autre part, des expressions régulières au niveau des *tokens* peuvent être également appliquées avant l'analyse en dépendances. Enfin, le *parsing* est effectué sur le texte découpé et étiqueté.

[17] Annexe 2 (p.108)

2. L'analyseur en dépendance

Pour l'analyse morphosyntaxique, Holmes prend comme base le *MaltParser*[18] dans une version qui a été optimisée par l'entreprise, pour le présent système.

Le *MaltParser* est un système de *parsing* fondé sur un modèle statistique de *machine learning* (versus un traitement symbolique), non spécifique à une langue. A partir d'un *treebank* de dépendances dans une langue quelconque, et après apprentissage, ce système permet de créer un parseur adapté pour l'analyse de textes sur le modèle de relation du *treebank* donné en entrée. Ce modèle est personnalisable sous de nombreux angles, ce qui en fait sa flexibilité. En effet, plusieurs algorithmes sont mis à disposition pour le *parsing*, pour la phase d'apprentissage sur corpus, ainsi que pour le modèle de traits (qui permet d'utiliser les différents traits de l'étiquetage pour déterminer les choix en fonction du contexte) [Hall *et al.,* 2006].

Dans le cadre du français, le *treebank* choisi est celui du *French Treebank* (FTB) en dépendances de surface [Candito *et al.,* 2010]. La liste des relations est exposée en Figure 15.

[18] http://www.maltparser.org/mco/french_parser/fremalt.html

Relations pour gouverneurs verbaux (cf. schéma d'annotation FTB + ajouts)	
suj	Sujet
obj	objet
de_obj	SP argumental en de, non locatif
a_obj	SP argumental en à, non locatif
p_obj	autre SP argumental
ats	Attribut du sujet
ato	Attribut de l'objet
mod	Modifieur
auux_tps	auxiliaires de temps
aux_pass	auxiliaires du passif
aux_caus	verbe causatif (en cas de complexe causatif + inf)
aff	clitiques figés
Relations pour gouverneurs non verbaux	
mod	Modifieurs repérés structuralement (par exemple adjectifs épithètes), autres que les relatives
mod_rel	Relatives adnominales
coord	Relation portée par un coordonnant, avec comme gouverneur le coordonné immédiatement précédent
arg	Utilisé dans le cas de prépositions « liées » : dans *de Charybde en Scylla*, parallèlement au traitement de la coordination, *en Scylla* est dépendant de type *arg* de la première préposition *(de)*
dep_coord	Relation portée par un coordonné (sauf le premier), avec comme gouverneur le coordonnant immédiatement précédent
det	Relation portée par les déterminants
ponct	Relation portée par tout dépendant typographique, sauf pour les virgules jouant le rôle de coordonnant (qui porte la relation *coord*)
dep	Relation sous-spécifiée, pour les dépendants prépositionnels (pas de gestion de la distinction argument / ajout pour les gouverneurs non verbaux)

Figure 15 : *Liste des relations de dépendance du French Treebank [Candito et al., 2011 : 5]*

Maintenant que nous avons présenté les deux modèles de dépendances, nous allons les mettre en parallèle afin de bien appréhender leurs différences d'approche.

Chapitre 5 – comparaison des deux systèmes

Chaque système de dépendances est conçu dans un but précis. Nous avons vu que celles du *French Treebank* ont été imaginées de telle manière qu'elles ne restreignent pas leur utilisation à un type de tâche en particulier, mais qu'elles soient utilisées comme un pivot entre une analyse syntagmatique et une analyse en dépendances. Cela permet ainsi aux utilisateurs d'adapter ces relations en fonction de leurs besoins. Les SD ont quant à elles été pensées en vue de traitements sémantiques par des spécialistes comme des non-spécialistes de la linguistique computationnelle. Le but est donc de fournir des relations mettant en évidence les termes dont le sens est plein, tout en utilisant des noms compréhensibles par tous.

Cette différence d'objectif se répercute ainsi sur les types de relations définis par chacun des modèles, ce que nous allons examiner par la suite.

1. Surface vs profondeur

La principale différence qui sépare les deux modèles est l'opposition surface / profondeur. Cela implique dans les relations de Stanford une diminution des relations entre mots pleins et mots grammaticaux, au profit de relations plus sémantiques entre mots pleins. Cela se ressent à plusieurs niveaux.

Tout d'abord, contrairement à la sortie obtenue dans Holmes, les racines et têtes de propositions sont toujours représentées par le mot le plus porteur de sens. Cela signifie, par exemple, dans le cas de conjugaisons avec un ou plusieurs auxiliaires modaux, que le verbe non modal est en tête, et les auxiliaires y sont reliés par la relation *aux* (exemple 1). Le même raisonnement est suivi pour les cas d'attributs du sujet pour lesquels c'est ce dernier qui est gouverneur de la relation, appelée alors *cop* pour copule (exemple 2).

(1) He could eat. *=> Stanford : aux (eat, could)*

Il peut manger. *=> Holmes : obj(peut, manger)*

(2) He is tall. *=> Stanford : cop(tall, is)*

Il est grand. *=> Holmes : ats(est, grand)*

D'autre part, pour tous les cas de propositions relatives, subordonnées, etc., ce sont les mots-tête de chacune des propositions qui sont mis en relation au niveau de Stanford, lorsque la sortie de Holmes lie la première proposition au pronom / conjonction, puis ce dernier à la deuxième proposition (exemple 3).

(3) I don't know when you're coming.

=> Stanford : advcl(know, coming), advmod(coming, when)

Je ne sais pas quand tu viens.

=> Holmes : mod(sait, quand), obj(quand, viens)

Quant aux cas de propositions infinitives[19], lorsqu'un objet de la proposition principale est présent en sortie de *Holmes*, il est considéré dans Stanford le sujet de la proposition infinitive. De plus, une relation particulière *ccomp* relie les deux têtes de proposition, lorsque Holmes la traite comme un objet, un modifieur ou un attribut de l'objet (exemple 4).

(4) He let it fall. *=>Stanford : ccomp(let, fall), nsubj(fall, it)*

Il l'a laissé tomber *=> Holmes : obj(laissé, tomber), obj(laissé, l')*

La même idée s'applique pour les compléments d'objet indirect infinitifs, donc introduits par une préposition[20] « de » ou « à » (« to » en anglais), avec la relation *xcomp*, avec l'objet de la proposition principale dans Holmes qui est le sujet de l'infinitif dans Stanford.

(5) He asks me to come.

=>Stanford : xcomp(asks, come), aux(come, to), nsubj(come, me)

Il me demande de venir.

=> Holmes : obj(demande, de), obj(de, venir), aff(demande, me)

Enfin, pour la coordination simple (deux éléments coordonnés), ce sont les deux mots pleins coordonnés qui sont dans une relation avec le modèle de Stanford, tandis qu'il faut deux dépendances pour les relier avec la version de Holmes (exemple 6).

(6) I have a red and black sweater.

=> Stanford : cc(red, and), conj(red, black)

J'ai un pull rouge et noir.

=> Holmes : coord(rouge, et), dep_coord(et, noir)

[19] Plus d'explication sur la proposition infinitive : http://www.synapse-fr.com/manuels/PROP_INFI.htm dernière consultation : 08/09/13

[20] Pour plus d'information sur les compléments d'objet indirect infinitifs : http://research.jyu.fi/grfle/466.html dernière consultation : 08/09/13

2. Hiérarchie

Une différence notable entre les deux systèmes se situe également au niveau de la notion de hiérarchie qui permet (en théorie) au *parser* de Stanford de déterminer la relation sous-spécifiée se rapprochant au plus de la structure concernée, lorsque celle-ci n'est pas reconnue par le système. Pourtant, en pratique, peu de ces catégories supérieures apparaissent réellement (uniquement *dep, aux, advmod* et *npadvmod*) : celles qui ne sont pas définies dans le manuel de dépendances par Manning *et al.* [2012] n'ont jamais été aperçues dans les nombreuses analyses de test que nous avons effectuées. Le système de dépendances du FTB, cependant, n'est pas organisé d'une telle manière : les relations sont toutes au même niveau hiérarchique.

3. Niveau de détail

La granularité des relations diffère entre les deux systèmes de dépendances. Alors que Stanford propose 49 relations (sans compter les relations du format « *collapsed* »), celles du *French Treebank* sont au nombre de 20. Cette différence se fait nettement ressentir dans la transition de l'un à l'autre, les parties du discours de l'étiquetage devenant alors des attributs essentiels pour différencier les différent cas, comme nous le verrons au niveau de l'implémentation au chapitre 8.

Ce contraste est en effet très visible au niveau de la relation *mod* de Holmes, plus ou moins équivalente à la relation sous-spécifiée de Stanford *mod*, qui se divise, elle, en 25 relations (rappel Figure 14). Leurs étiquettes peuvent désigner la partie du discours du gouverné (ex : *amod* pour les adjectifs, *infmod* pour les infinitifs, *partmod* pour les participes passés et présents, *advmod* pour les adverbes, etc.), ou bien remplir des fonctions plus sémantiques comme par exemple *tmod* pour les expressions temporelles simples (ex : tomorrow, today, Thursday, month, etc.), *quantmod* lorsque le gouverné modifie une quantité (ex : Around 200 people) ou encore *num* pour désigner les noms qualifiés d'un nombre (ex : two eggs).

En outre, les dépendances du FTB caractérisent (en théorie) les compléments essentiels indirects du verbe, introduits par une préposition via les relations *a_obj* et *de_obj et p_obj* (ainsi que *aff*, cas que nous aborderons plus loin dans cette section). Celles de Stanford ne distinguent pas les compléments essentiels (exemple 7) des non-

essentiels (en cas de complément prépositionnel) (exemple 8), elles sont toutes réunies sous la relation *prep*.

(7) It is raining in Lyon. *=> Stanford : prep(raining, in), pobj(in, Lyon)*

Il pleut à Lyon. *=> Holmes : mod(pleut, à), obj(à, Lyon)*

(8) I'm going to Paris *=> Stanford : prep(going, to), pobj(to, Paris)*

Je vais à Paris. *=> Holmes : p_obj(vais, à), obj(à, Paris)*

Nous pouvons par ailleurs constater que l'objet de la préposition est représentée par une relation spécifique *pobj* (ex : pobj(on, chair)) lorsque la traditionnelle relation *obj* est utilisée dans le FTB.

Cependant, les SD opposent les objets directs (*dobj*) aux objets indirects (*iobj*) lorsqu'ils sont représentés par des pronoms personnels clitiques. Dans le FTB, au contraire, certains pronoms personnels objet (direct ou non) et pronoms réfléchis sont regroupés sous le même type *aff* pour affixe *(exemple 9).*

(9) He promised me a present. *=> Stanford : iobj(promised, me)*

Il m'a promis un cadeau. *=> Holmes : aff(promis, m')*

Toujours dans l'optique de se rapprocher de la sémantique, les relations de Stanford différencient par ailleurs les sujets de constructions passives (*nubjpass*) des constructions actives (*nsubj*), tous les deux traduits par *suj* dans le *French Treebank* (exemple 10*)*.

(10) My bike was stolen. *=> Stanford : nsubjpass(stolen, bike)*

Mon vélo a été volé. *=> Holmes : suj(volé, vélo)*

Nous avons opposé les deux systèmes de dépendances et constaté que de nombreuses différences les distinguent. Nous allons maintenant nous orienter sur la technique qui va permettre de les mettre en confrontation en vue de la conversion : la transformation de graphes.

Chapitre 6 – Méthode de conversion

1. La transformation de graphes

1.1 Généralités

Une méthode permettant l'enrichissement et la modification de relations de dépendances commence à se répandre dans le domaine du traitement automatique des langues : la transformation (ou réécriture) de graphes.

En effet, pour analyser et annoter des relations sémantiques ou des relations syntaxiques profondes pour le français, le FTB en dépendances de surface constitue une base très utilisée et sa structure en arbres, qui est un type spécifique de graphes, est donc idéale pour ce type de transformation.

Le principe de la réécriture de graphes est de modifier un sous-graphe dans l'arborescence générale pour qu'il corresponde à la structure finale voulue. Ces modifications peuvent s'appliquer au niveau des nœuds, des arcs, des attributs (au niveau des nœuds) et des types (nom de la relation au niveau des arcs). [Ribeyre, 2013]. La transformation de graphes permet donc beaucoup de liberté sur les types de transformations souhaitées. C'est pour cette raison que nous nous sommes tournés vers ce procédé de réécriture de graphe pour l'application de notre conversion en dépendances profondes.

Dans le domaine du TAL, deux outils ont été directement conçus dans le but d'enrichir une structure en dépendances de surface, soit avec des annotations sémantiques : Grew [Bonfante *et al.,* 2010] , soit avec des dépendances profondes : Ogre [Ribeyre, 2012].

1.2 Grew

L'outil Grew[21] a pour objectif d'annoter automatiquement au niveau sémantique un corpus analysé en dépendances, plus particulièrement le FTB [Abeillé *et al.,* 2003]. Le choix de ce formalisme s'est justifié par le format de structure sémantique adopté : la DMRS (*Dependency Minimal Recursion Semantics*) [Copestake, 2006] qui fournit des graphes de dépendances sémantiques cycliques.

[21] http://wikilligramme.loria.fr/doku.php?id=grew:grew dernière consultation : 08/09/13

Chaque règle de réécriture R est composée de trois éléments (Figure 16). Le premier élément est le « patron positif », qui est le graphe qui sera testé sur G, le graphe à transformer (1 sur la figure). Puis, les « patrons négatifs » (facultatifs) représentent les NAC (*negative application condition*) pour lesquels la règle ne s'appliquera pas, malgré le fait que le patron positif soit vérifié (2 sur la figure). Enfin, des « commandes » vont opérer les transformations sur les arcs, les nœuds, les attributs de G pour obtenir le résultat attendu (3 sur la figure). Au final, nous avons donc un système qui « cherche à apparier le patron positif de la règle […] avec une partie du graphe, et [qui] vérifie que l'appariement ne peut être étendu en un appariement pour aucun des patterns négatifs ». [Guillaume *et al.,* 2012 : 5]

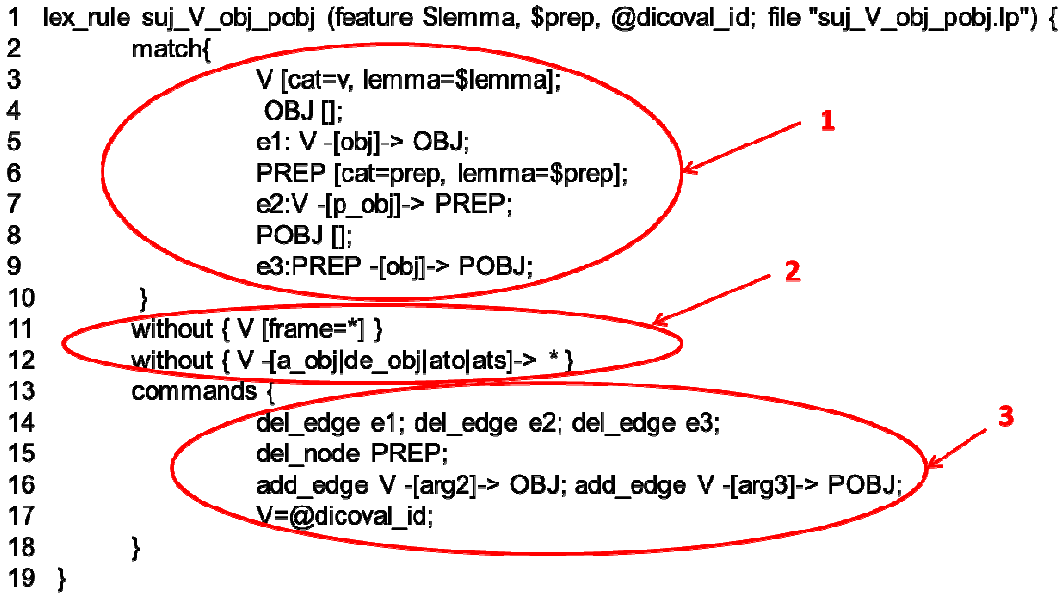


Figure 16 : *Exemple d'une règle qui « transforme les actants syntaxiques en actants sémantiques » [Guillaume* et al.*, 2012 : 5]*

Des règles sont organisées et appliquées par modules ordonnés pour éviter les interactions non souhaitées. Des paramètres supplémentaires peuvent également entrer en ligne de compte pour préciser des traits : des lemmes, etc. au niveau des patrons et des commandes. Enfin, des filtres sont créés pour éliminer les incohérences générées par la transformation, ou encore permettre d'écrire des règles simplifiées en supprimant les surgénération dans cette phase [Guillaume *et al.,* 2012]

Malgré toutes ces spécificités qui semblent adaptées à notre projet, ce système n'a pu être adopté car les seules versions disponibles sont pour Linux et Mac OS X, lorsque les ordinateurs de l'entreprise fonctionnent sous Windows.

1.3 Ogre

Le système Ogre [Ribeyre, 2012] présente un fonctionnement différent de celui de Grew que nous avons présenté en section 1.2. Le principe de base reste la réécriture de graphes, mais l'accent est mis sur l'utilisation d'une approche dite de « propagation de contraintes » au niveau des arcs. Il a été conçu dans le but d'insérer des relations de dépendances profondes à une analyse de surface. Les règles produites se veulent génériques afin de pouvoir s'adapter à différents analyseurs syntaxiques [Ribeyre, 2013].

Le langage créé spécialement pour cette transformation de graphe est nommé GrQL (Graph Query Language). A partir d'un patron constitué d'un graphe comportant nœuds, arcs et traits (au niveau des nœuds et des arcs), ce langage permet d'exécuter des commandes classiques du type ajout / suppression d'arc (ex : « add_node(F) » où F désigne la structure de traits), ajout / suppression de nœuds, etc., tout en autorisant la négation dans les motifs. L'ordre des règles est établi de manière automatique via un compilateur, les règles les plus spécifiques (Figure 17) étant appliquées avant les plus générales (Figure 18).

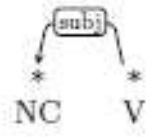


Figure 17 : *Motif plus spécifique [Ribeyre, 2012 : 35]*

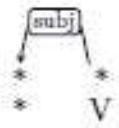


Figure 18 : *Motif plus général [Ribeyre, 2012 : 35]*

Le compilateur permet également de remonter les conflits potentiels à l'utilisateur, en cas de règles présentant des arcs en commun et des transformations

contradictoires. Par exemple, si deux règles de réécriture s'appliquent sur un même arc, l'une pour supprimer l'arc et l'autre pour en changer la cible, il y a un conflit à signaler [Ribeyre, 2012].

Ogre donne la possibilité d'ajouter des contraintes sur les arcs, permettant ainsi de traiter des cas plus complexes d'opérations, et de produire des règles génériques tout en en maitrisant le nombre. Ces contraintes se déclenchent après la première partie de réécriture de graphe, en fonction du contexte des arcs et des nœuds. Elles permettent d'ajouter des relations profondes à la structure et sont au nombre de quatre (un exemple est donné en Figure 19, avec une contrainte utilisée pour « faire remonter les arcs sortants de y » [Ribeyre, 2012 : 48]).

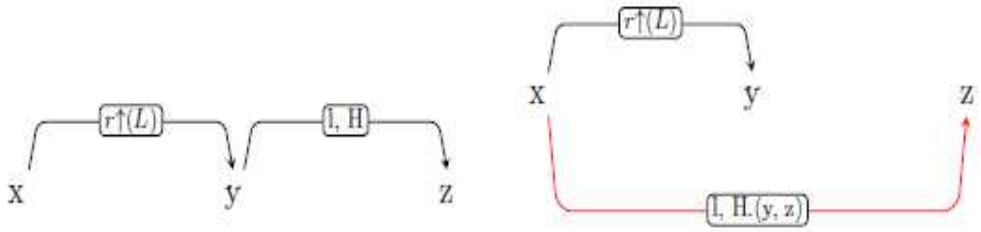


Figure 19: Contrainte Redirect up [Ribeyre, 2012 : 49]

Au final, un filtre est appliqué pour supprimer, parmi les relations créées, celles qui n'ont pas de signification linguistique.

Les paramètres que l'on peut spécifier, la possibilité d'utiliser la négation et les définitions des contraintes sont des éléments totalement adapté à notre projet. Cependant, ce système n'est actuellement pas mis à disposition par son auteur.

Les deux approches exposées brièvement ici vont donc dans le sens de notre projet mais ne sont pas exploitables pour le moment. Nous nous sommes alors tournés vers un outil de transformation de graphes non spécifique au domaine du TAL, que nous présentons dans la section 2).

2. *Outil utilisé*

Le choix s'est porté sur AGG[22] (*Attributed Graph Grammar*), un environnement de développement pour la transformation de graphes attribués[23] et typés[24].

2.1 AGG

AGG est un outil permettant la réécriture de graphes dont les arcs et les nœuds peuvent posséder des traits. Le modèle fonctionne selon un principe de modification de graphes via des règles. La grammaire consiste en un graphe initial (orienté et étiqueté) et une liste de règles qui décrit les transformations à apporter au graphe. Ces règles sont elles-mêmes formées de graphes indiquant le motif à transformer L (« left-hand side ») et le graphe après transformation R (« right-hand side ») reliés par un morphisme L → R. Le graphe initial, tout comme les graphes motifs, peut être attribué et typé, c'est-à-dire que leurs arcs et nœuds peuvent recevoir des traits. Un trait est défini par l'utilisateur, qui doit spécifier son type, son nom et sa valeur (par exemple *pos* (type) *= « NC* » (valeur), avec *pos*=PartOfSpeechAnnotation (nom)). La grande particularité d'AGG, et ce qui justifie principalement notre intérêt dans cet outil, est que le champ des types et attributs du motif peuvent être décrits par des objets Java et récupérés par une variable (sans être ainsi limité à une chaîne de caractère), ce que nous illustrerons dans la section 2.2. Cela laisse ainsi beaucoup plus de liberté dans la transformation et les motifs, et permet de traiter la disjonction,. La gestion des NAC[25] est également prévue par le système [Taentzer, 1999].

L'application AGG fournit une interface graphique (Figure 20) a été conçue à partir de l'outil de transformation de graphe, pour pouvoir définir et paramétrer, tout en les visualisant, les règles et les graphes (*left-hand side*, *right-hand side* et le graphe initial), puis de réaliser l'exécution soit de l'ensemble des règles, soit pas à pas. Les règles peuvent être organisées en couches (« *layers* ») pour spécifier l'ordre d'application, ou non.

[22] http://user.cs.tu-berlin.de/~gragra/agg/ dernière consultation : 08/09/13
[23] Lorsque des attributs sont précisés sur les arcs
[24] Lorsque le type des relations est spécifié sur les nœuds
[25] Conditions qui empêchent la règle de s'appliquer

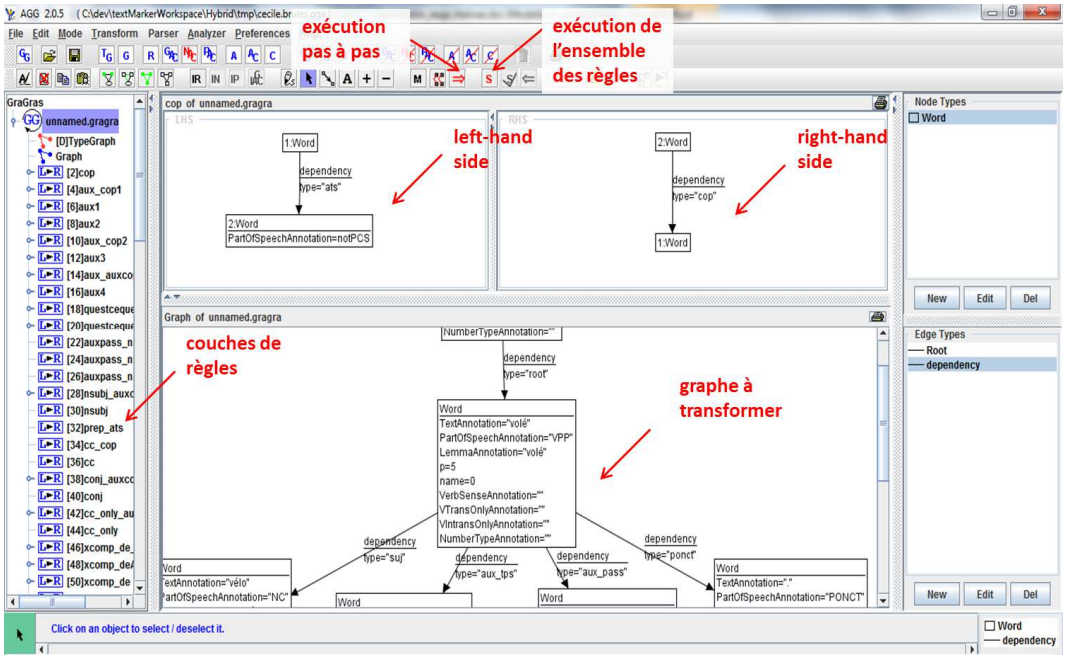


Figure 20 : *Interface d'AGG*

Cette interface ayant montré des limites au niveau de la rapidité et la facilité de création de règles, un formalisme spécifique a été développé par Luca Dini (Ho2s) pour permettre de définir ces règles via une syntaxe plus adaptée à nos besoins, règles nommées « *B-rules* » que nous présentons en section 2.2.

2.2 Syntaxe en règles

La syntaxe des *B-rules* est assez simple (Figure 21). Un commentaire en début de règle (obligatoire) décrit la règle qui va suivre (n°1 sur la figure). Puis, un numéro est attribué à la règle (n°2), pour définir l'ordre d'application. Un nom, unique, est également affecté comme identifiant de la règle (n°3). La règle est ensuite composée de deux parties principales, reprenant les *left-hand side* et *right-hand side* de AGG : L => R. La partie gauche, L, définit les nœuds du motif ([1] et [2] dans le cas de la Figure 21, n°4) auxquels des noms (ex : 1_NOM) peuvent être associés pour permettre une meilleure lisibilité, mais sans conséquence sur le traitement de la règle. Cette première partie comprend également la définition des arcs orientés (nœud 2 vers nœud 1, « type » désignant la relation, c'est-à-dire « suj » en Figure 21, n°5). La partie droite comporte les mêmes caractéristiques. Elle définit le graphe obtenu après transformation du motif, à travers la redéclaration des nœuds (et de leurs éventuels modifications : changement de trait, etc.) (n°6) et des relations qui les lient (n°7). Enfin, une phrase de test conclut la règle, en

donnant un exemple (le plus simple possible) du cas de figure représenté par la *B-rule* (n°8).

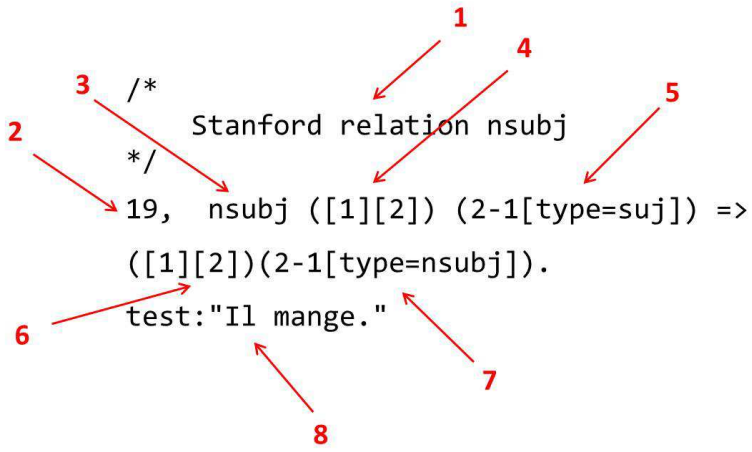


Figure 21 : *Un exemple simple de* B-rule *: le cas du sujet*

Plusieurs paramètres permettent d'enrichir ce modèle de base. Des attributs peuvent être définis (si déclarés) et ajoutés dans les descriptions des nœuds. Nous avons par exemple principalement utilisé « pos » (Part-Of-Speech) et « lemma », mais également « vtrans » (pour les verbes exclusivement transitifs), « vintrans » (pour les verbes exclusivement intransitifs), « numType » (pour les nombres en chiffres ou en lettres) et « vclass » (pour sélectionner les verbes modaux et/ou les auxiliaires). Ces attributs peuvent correspondre à une seule valeur (comme en Figure 22 où la partie du discours du nœud 2 doit obligatoirement être une conjonction de coordination, CC), ou bien plusieurs.

```
/*
   Stanford relation cc
   Handles Holmes relation coord
*/
21, cc([1_mange][2_et pos=CC])(1-2[type=coord]) =>
([1][2])(1-2[type=cc]).
test:"Il mange et part."
```

Figure 22 : *Cas simple d'utilisation d'attribut dans une* B-rule

Pour cela, il est possible de déclarer une variable (symbolisée par un « _ » suivi du nom de la variable) récupérant le résultat d'une méthode Java, définie au préalable dans

la librairie, et appelée dans la *B-rule* (ex : Figure 23, où la partie du discours du nœud 1 peut être soit un nom commun, NC, soir un nom propre, NP).

```
/*
   Stanford relation amod
   Handles noun + adj cases
   Other example : "Je vois Marie anxieuse."
*/
260, amod([1 pos=_NcNpp][2 pos=ADJ])(1-2[type=mod]) =>
([1][2])(1-2[type=amod])
:=
(  {TestStaticFunctions.regxp(NcNpp,"NC|NPP")} ).
test:"Il mange une pomme verte."
```

Figure 23: *Utilisation de méthode Java dans une* B-rule *(disjonction)*

La méthode Java utilisée principalement dans ce projet, nommée « regxp », permet de comparer les valeurs des attributs avec des expressions régulières. Cela nous permet une grande flexibilité. Nous pouvons ainsi traiter par exemple les disjonctions, comme en Figure 23, et des exceptions comme en Figure 24 (toutes les partie du discours, sauf un verbe à l'infinitif). Les « types » des relations peuvent également être soumis aux méthodes Java (Figure 24 en vert). Ces traitements doivent cependant être bien contrôlés pour éviter les surgénérations.

```
/*
   Stanford relations prep & pobj
*/
45, prep([1][2 pos=P][3 pos=_notVinf])(1-2[type=_multi] 2-3[type=obj])
=>
([1][2][3])(1-2[type=prep] 2-3[type=pobj])
:=
(
  {!TestStaticFunctions.regxp(notVinf,"VINF")}
  {TestStaticFunctions.regxp(multi,"dep|p_obj|a_obj|mod|obj|de_obj")}
).
test:"C'est la chaussure de David."
```

Figure 24 : *Utilisation de méthode Java dans une* B-rule *(exception)*

Lorsque les nœuds de la partie gauche de la règle contiennent des attributs, il n'est pas nécessaire de les répéter ensuite dans la partie droite, ils seront conservés. Cependant, le type est obligatoire pour les relations, et doit donc toujours être spécifié à gauche comme à droite. Enfin, si un nœud ou une relation ne sont pas répétés dans la partie droite, ils seront alors supprimés.

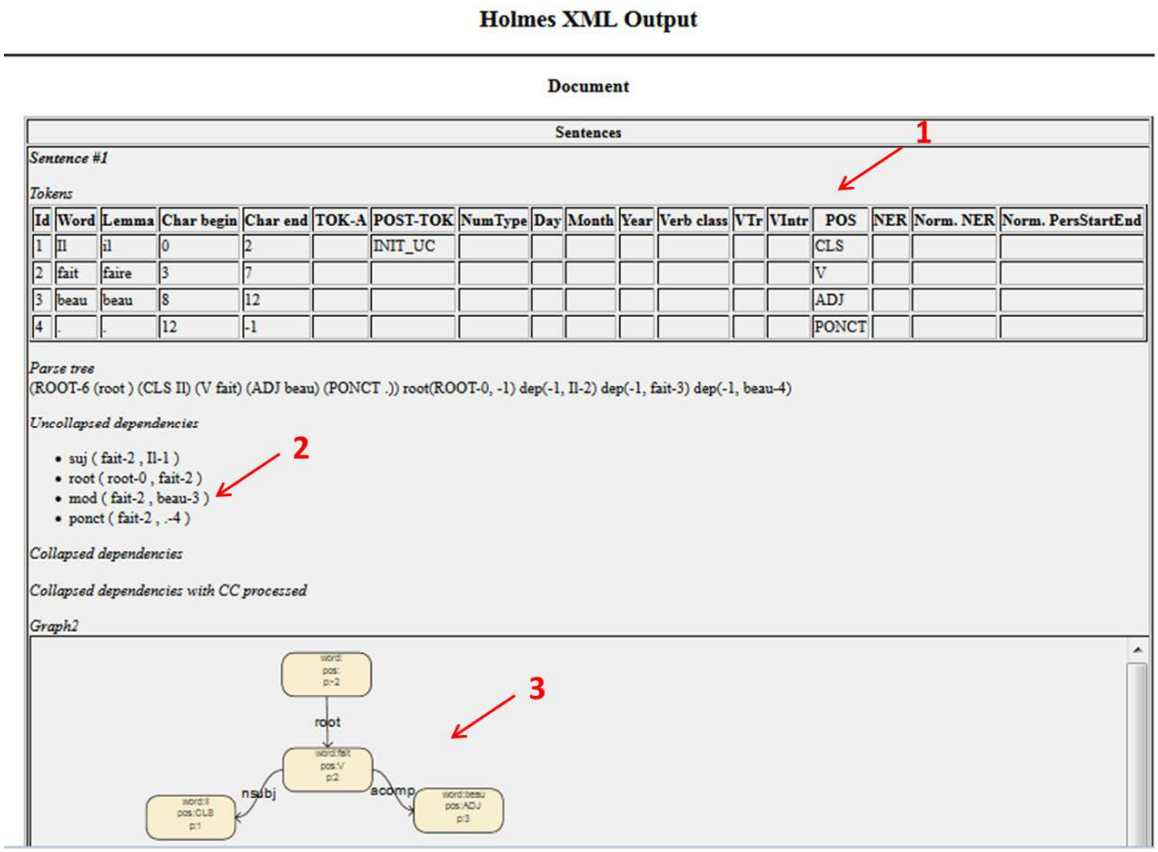
Ainsi, ce formalisme nous permet de conserver uniquement les fonctionnalités de AGG qui nous sont utiles, et ainsi faire du tri dans la multitude d'outils disponibles dans l'interface. La syntaxe se veut épurée et explicite, pour simplifier la lecture et l'écriture des règles. Ce formalisme nous permet donc un gain de temps et de clarté par rapport à l'interface d'AGG.

2.3 Application des règles

Deux formes d'exécution du module sont possibles.

La première permet de compiler les *B-rules* et de créer un fichier interprétable dans l'interface de AGG. Deux solutions sont possibles à ce niveau. Un seul fichier contenant l'ensemble des règles ordonnées peut être créé, en prenant un paramètre qui est le graphe initial, graphe que l'on veut transformer et sur lequel l'ensemble des règles seront appliquées. En seconde solution, un fichier par règle est généré, et la phrase choisie comme graphe initial sera la phrase de test incluse dans la *B-rule*. A partir de ces fichiers, nous pouvons alors faire la transformation via l'interface graphique de l'AGG (Figure 20), et ainsi pouvoir vérifier pas à pas la bonne exécution de règles en particulier.

La deuxième méthode consiste à lancer directement l'outil Holmes, qui fournit un document XML (Figure 25) contenant l'analyse morphosyntaxique (en 1), l'analyse en dépendances de surface (en 2), et le graphe transformé en dépendances profondes (en 3).



Holmes XML Output

Document

Sentences

Sentence #1

Tokens

Id	Word	Lemma	Char begin	Char end	TOK-A	POST-TOK	NumType	Day	Month	Year	Verb class	VTr	VIntr	POS	NER	Norm. NER	Norm. PersStartEnd
1	Il	il	0	2		INIT_UC								CLS			
2	fait	faire	3	7										V			
3	beau	beau	8	12										ADJ			
4	.	.	12	-1										PONCT			

Parse tree

(ROOT-6 (root) (CLS Il) (V fait) (ADJ beau) (PONCT .)) root(ROOT-0, -1) dep(-1, Il-2) dep(-1, fait-3) dep(-1, beau-4)

Uncollapsed dependencies

- suj (fait-2 , Il-1)
- root (root-0 , fait-2)
- mod (fait-2 , beau-3)
- ponct (fait-2 , .-4)

Collapsed dependencies

Collapsed dependencies with CC processed

Graph2

Figure 25 : *Document XML donnant le résultat de l'analyse de Holmes*

Pour transformer les relations du *FTB* vers celles de Stanford, il nous faut avant tout adapter ces dernières pour couvrir les spécificités du français et adopter des choix par rapport à ce qu'il est possible (ou non) de faire en fonction du langage, des dépendances sources, et des particularités du *parsing*.

Chapitre 7 – Adaptation des relations

L'anglais et le français ont chacune leurs particularités (par exemple les *phrasal verbs* : « *He fell* ***down »***, spécifique de l'anglais ou la construction « Qu'est-ce que » en français). Il est alors difficile, voire impossible de conserver toutes les relations prévues initialement pour l'anglais tout en respectant la structure syntaxique de la langue française. Des ajouts, modifications et suppressions de relations au passage du système actuel de Holmes à celui des SD sont inévitables, comme nous allons le voir dans les sections suivantes.

1. Ajustement des relations de Stanford

1.1 Ajouts et modifications

Au final, peu de relations ont été créées spécialement pour le français. En effet, suivant notre objectif final de transformation vers les SD, nous avons employé au maximum les relations développées par Stanford pour s'approcher au mieux des concepts véhiculés par celles-ci. L'avantage de travailler sur ces deux langues, c'est qu'elles sont de la même famille indo-européenne, et leur proximité typologique (SVO) permet de les rapprocher sous beaucoup d'aspects au niveau de la structure syntaxique, mais, bien entendu, pas tous.

Nous avons par exemple dû concevoir une relation pour traiter les cas de comparaison : la dépendance *compar*. En anglais les structures classiques de comparaison sont constituées d'un adverbe comparatif (« more », « less », « as », etc.) situé devant l'adjectif ou l'adverbe à comparer (comme le français : « plus », « moins », « autant », etc.), ou bien d'un adjectif comparatif (ex : « happier »). Dans Stanford, cette première partie est analysée comme *advmod* pour l'adverbe comparatif, et *amod* pour l'adjectif comparatif. Nous pouvons donc conserver l'interprétation de l'adverbe dans nos SD du français. La deuxième partie de comparaison en français est toujours introduite par la conjonction « que ». En anglais cependant, la conjonction n'est utilisé que lorsque la comparaison s'effectue avec une proposition (ex : « *He looks* ***stronger than*** *you do.* »), et est une préposition dans les autres cas (ex : *« She runs* ***faster than*** *you*. », « *He is* ***as fast as*** *you* »). Nous ne pouvons dans ce cas nous inspirer de l'analyse de l'anglais, qui relie les deux têtes de proposition, et implique la conjonction dans une dépendance *mark*, spécifique aux cas de propositions (Figure 26).

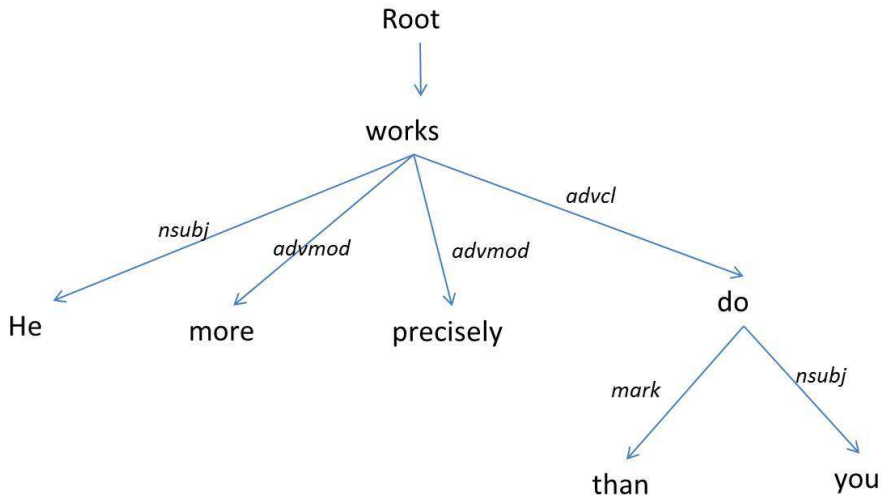


Figure 26 : *Analyse en SD de proposition comparative : « He works more precisely than you do. »*

Comme il n'existe pas de traitement « classique » pour les conjonctions, nous avons choisi de définir cette nouvelle relation, qui permet par ailleurs de transmettre la sémantique de la comparaison (exemple 11).

(11) Elle court aussi vite que moi. => Holmes : dep(vite, que), obj(que, moi)

=> Stanford : compar(vite, que), dobj(que, moi)

Cependant, elle ne couvre pas pour l'instant les propositions comparatives (ex : Il est plus grand que je ne le croyais), mais seulement les comparaisons avec syntagme nominal subordonné (ex : Il est plus grand que toi). En effet, l'analyse de Holmes est différente de celle de la comparaison avec un syntagme nominal (exemple 11), et de plus, elle est variable. Nous voyons par exemple que deux interprétations différentes sont données pour deux propositions comparatives : l'adverbe de comparaison dépend dans un premier cas de l'adjectif « grand » (Figure 27) et dans l'autre cas directement du verbe (Figure 28). Il nous faudrait alors rechercher toutes les interprétations pour s'adapter à tous les cas. A l'heure actuelle, la correspondance en dépendances profondes qui s'applique par défaut est celle des propositions subordonnées complément d'objet, introduites par la conjonction « que » (qui sera expliqué en détail en section 2.2).

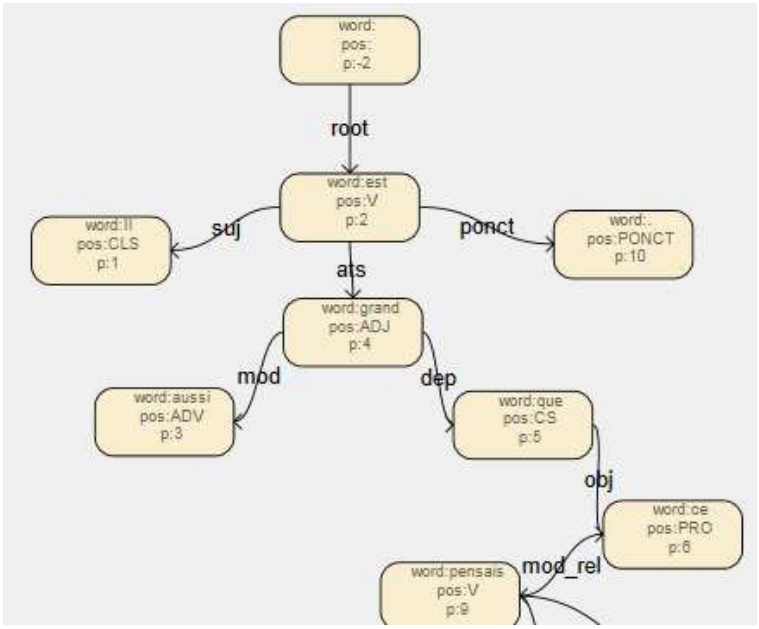


Figure 27 : *Analyse de Holmes pour « Il est aussi grand que ce que je pensais. »*

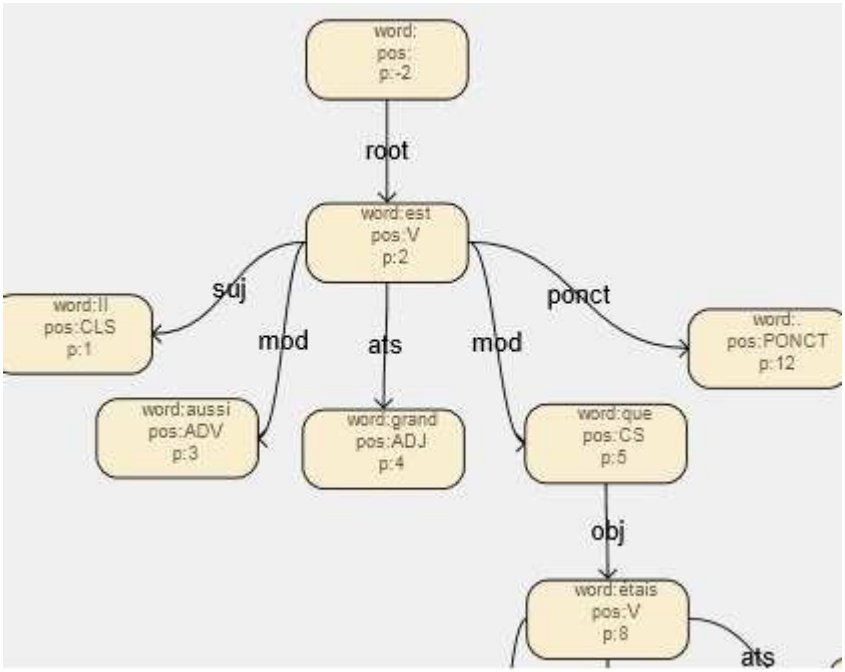


Figure 28 *: Analyse de Holmes pour « Il est aussi grand que tu l'étais à son âge. »*

Un autre cas de figure spécifique au français nous a poussé à former une nouvelle relation : la relation *refl*. En effet, le pronom réfléchi en anglais fonctionne soit comme objet direct, analysé en *dobj* (ex : *he cut himself*), soit comme objet indirect, *iobj* (ex : *He gives himself a present.*). Pour le cas du français, nous aurions également pu choisir de traduire celui-ci par une relation *dobj* ou *iobj* en fonction du contexte (ex : Il se connaît. ->

Qui connaît-il ? -> *dobj,* ; Il se demande pourquoi. -> A qui demande-t-il pourquoi ? -> *iobj*), mais cette méthode n'est pas toujours adaptée, par exemple pour les verbes pronominaux dits subjectifs[26] (ex : Il s'aperçoit de son erreur. -> *Qui aperçoit-il ? : sens différent). Nous avons décidé de représenter cette notion à travers une dépendance spécifique, *refl* (exemple 12).

(12) Il se base sur des faits. => Holmes : refl(base, se)

En ce qui concerne les modifications apportées aux relations existantes de Stanford, l'une porte sur une notion de relation qui ne nous semblait pas adaptée pour le français. En effet, Stanford contient des dépendances *xcomp*, *infmod* et *purpcl* qui traitent les cas de compléments (essentiels et non-essentiels) infinitifs introduits par « to » en anglais (exemples 13, 14 et 15).

(13) He asks me to come. => Stanford : xcomp(asks, come), aux(come, to)

(14) He jumped to escape. => Stanford : purpcl(jumped, escape), aux(escape, to)

(15) There are points to establish. => Stanford : infmod(points, establish),
aux(establish, to)

Une relation nommée *aux* est utilisée pour lier l'infinitif à « to ». Le terme « to » est un cas particulier dans Stanford. En effet, il est étiqueté avec une partie du discours qui lui est spécialement dédiée : TO. Or, en français, les prépositions qui encadrent ces cas de *xcomp*, *infmod* et *purpcl* sont multiples : « de » (ex : Il me demande de venir.), « à » (ex : Elle apprend à marcher. ; Il y a les vitres à nettoyer.) et « pour » (ex : Il sauta pour s'échapper). De plus, la relation *aux* étant déjà utilisée pour les relations avec les auxiliaires, ce terme nous a alors semblé inadapté ici. Nous avons donc pris la décision de traduire ces cas par la relation Stanford *prep*, qui relie un gouverneur à la préposition (quelconque) qu'il gouverne.

(16) Elle apprend à marcher. => Stanford : prep (marcher, à)

Enfin, la structure hiérarchique prévue par Stanford n'a pas pu être conservée entièrement (nous pouvons conclure qu'elle est d'ailleurs presque inexistante en

[26] Verbes pronominaux pour lesquels le pronom est totalement incorporé au verbe (ex : s'apercevoir, s'écrier, s'évanouir, se méfier, etc)

pratique, après le test de nombreuses phrases en anglais dans la version démo[27] de Stanford). En effet, les règles de transformation que nous avons élaborées s'appliquent à traiter des cas spécifiques. Stanford prévoit cependant des relations sous-spécifiées (autrement dit plus générales : *suj*, *comp*, etc rappel Figure 14). Dans les cas où le système ne reconnaît pas de relation spécifique adaptée, il cherchera la relation sous-spécifiée se rapprochant le plus possible du contexte. Notre système n'étant qu'une transformation de relations, nous ne pouvons analyser le contexte de la phrase pour en déduire la relation sous-spécifiée appropriée. Cela nécessiterait par ailleurs une recherche détaillée des éléments nécessaires pour lier une relation sous-spécifiée à un contexte. Les seules que nous avons conservées sont les dépendances *mod* et *dep*. En effet, les dépendances du FTB contiennent déjà une relation *mod* qui désigne tout modifieur autre que les relatives. Lorsqu'une relation *mod* à la sortie de Holmes n'est adaptée à aucune de nos règles de transformation, nous la laissons alors en tant que relation sous-spécifiée. En ce qui concerne la relation de Stanford la plus générale, c'est-à-dire *dep*, nous transformons toute dépendance de Holmes non transformée à la fin du traitement dans cette relation.

Ainsi, nous avons créé deux nouvelles relations *compar* et *refl*, changé la relation de Stanford *aux* en *prep* dans un contexte précis, et modifié la structure hiérarchique des SD. De nombreuses suppressions ont également été opérées.

1.2 Suppressions

Certaines relations n'ont pas pu être maintenues, la plupart pour des raisons d'adaptation à la langue. En tout, dix-huit (incluant les relations sous-spécifiées) d'entre elles n'ont pas été conservées : *arg, comp, attr, obj, subj, csubj, csubjpass, expl, abbrev, appos, predet, preconj, mwe, quantmod, npadvmod, prepc, possessive, prt*.

Par exemple, *predet* et *preconj* désignent les termes qui modifient le sens du déterminant/de la conjonction qu'ils précèdent. Ils possèdent une étiquette grammaticale spécifique : PDT (« all », « both », « half », « many », etc.) [28].

(17) All the boys are here. => Stanford : predet(boys, All)

[27] nlp.stanford.edu :8080/parser/index.jsp
[28] http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html

(18) Both the boys and the girls are here. => Stanford : preconj(boys, Both)

En français, le seul équivalent que nous avons pu mettre en correspondance est celui de « tout » (exemple 19), qui est analysé comme adjectif. Avant de répertorier des cas semblables, la relation de Stanford correspondante (*amod*) semble alors plus adaptée à ce cas spécifique.

(19) Tous les garçons sont présents. => Holmes : mod(garçons, tous)

=> Stanford : amod(garçons, tous)

D'autre part, certains phénomènes qui se produisent en anglais ne possèdent pas d'équivalent de même structure en français. Par exemple, *attr* désigne un complément de verbe copule, et plus précisément, ceux introduit par un pronom interrogatif (exemple 20) Or, l'équivalent de cet exemple en français serait « Qu'est-ce que c'est ? », l'analyse est donc différente.

(20) What is it ? => Stanford : attr(is, What)

En outre, la description du manuel des SD ne fournit pas assez de précisions sur les cas concernés par cette relation pour que nous puissions le traiter de manière exhaustive. La même observation s'applique aux dépendances *mwe* (expressions polylexicales), *quantmod* (modifieur de quantité) et *npadvmod* (syntagme nominal modifiant un adverbe). Nous pourrions par la suite déterminer les expressions françaises à considérer comme *mwe,* en se fondant sur les nombreux travaux de recherche disponibles (par exemple [Constant, 2012]). Par ailleurs, certaines expressions sont déjà rassemblées en un token après l'analyse de Holmes (ex : « à cause de »), il n'est donc pas nécessaire de lier les composants de cette expression polylexicale entre eux. Quant à *quantmod*, seul l'exemple « About 200 people » (quantmod (200, About)) est mentionné dans le manuel. Nous avons par la suite découvert, à travers nos tests, d'autre termes concernés par cette relation (comme « *nearly* » ou « *approximately* »), mais au moment de la rédaction de ce rapport le système de règle n'inclut pas encore ces cas. La recherche de la liste exhaustive des mots concernés est donc à envisager dans une prochaine version.

Certaines relations touchent des cas spécifiques à l'anglais, qui ne sont donc pas adaptables aux français. Par exemple, la dépendance *prt* lie le verbe à sa *phrasal verb particle* (exemple 21). Il n'y a alors aucun équivalent en français. De la même manière, la

relation *expl* est définie pour analyser « there is » et « there are » (exemple 22). L'équivalent français étant « il y a », considéré comme un seul token dans Holmes, la correspondance n'est donc pas réalisable.

(21) They shut it down. *=> Stanford : prt(shut, down)*

(22) There are two options. *=> Stanford : expl(are, there)*

Enfin, certaines relations ont dû être supprimées car l'analyse de surface de Holmes ne permettait pas de traduire les structures sans créer d'ambiguïté. C'est le cas de *abbrev* (abréviation qui modifie, entre parenthèses, le nom qui la précède, voir exemple 23) et *appos* (relation entre un syntagme nominal mis en apposition avec un autre syntagme nominal via des parenthèses ou virgules, voir exemple 24). La sortie de Holmes ne permet pas de déterminer un quelconque lien entre les parenthèses (ou virgules) et les syntagmes qui les entourent. C'est la relation Stanford *nn* (nom modifiant un autre nom) qui traitera alors ces cas d'abréviation et d'apposition.

(23) L'électricité de France (EDF) … *=> Holmes : mod(électricité, EDF)*

=> Stanford : nn(électricité, EDF)

(24) Son cousin, Paul, ... *=> Holmes : mod(cousin, Paul)*

=> Stanford : nn(cousin, Paul)

Comme nous venons de le voir, certaines relations n'ont pu être conservées, d'autres ont été créées ou encore modifiées. Pour les dépendances maintenues, la correspondance n'a pas toujours été aisée à mettre en place comme nous le verrons dans la section qui suit.

2. Difficultés et choix

2.1 Difficultés et choix liés au français

Parmi les difficultés amenées par l'adaptation des dépendances au français, nous avons l'exemple de la relation *xcomp* entre un verbe (ou attribut) et un adjectif/participe passé, en présence d'un complément d'objet (Holmes), considéré alors comme le sujet de l'adjectif/participe passé (exemple 25).

(25) Je le trouve bizarre. *=> Holmes : ato(trouve, bizarre), obj(trouve, le)*

=> Stanford : xcomp(trouve, bizarre), nsubj(bizarre, le)

Nous avons dû restreindre la règle aux phrases contenant seulement un complément d'objet clitique. En effet, l'ambiguïté de portée de l'adjectif (ou participe passé) ne permet pas de trancher sur l'interprétation à donner lorsque l'objet est un nom commun (ex : J'ai trouvé le chat gris : « Je l'ai trouvé gris, le chat. » versus « Le chat gris, je l'ai trouvé. » : adjectif épithète ou attribut de l'objet direct ?)

Nous avons dû créer une règle spécifique pour les suites de deux prépositions (dépendantes l'une de l'autre), assez courantes en français (ex : Il revient de chez lui.). Les seuls équivalents trouvés en anglais sont les combinaisons « except » + préposition.

Concernant la dépendance *neg* (indiquant une négation), nous avons déterminé les termes dont la sémantique marque la négation. Outre les traditionnels « ne »/ « pas » et « non », nous avons également considéré les mots « rien », « personne » et « que » (exemple 26).

(26) Il ne mange qu'une pomme. => Holmes : mod(mange, ne), mod(mange, que)

=> Stanford : neg(mange, ne), neg(mange, que)

L'ajout d'éléments supplémentaires dans cette liste sont à prévoir dans une prochaine version (incluant ainsi « plus », « jamais », « nullement », etc.).

Une recherche de termes répondant à la sémantique du temps a également été opérée pour la relation *tmod*, traitant les mots qui sont des modifieurs temporels impliqués dans une relation avec la tête de phrase. Une liste a ainsi été constituée en se fondant sur les termes de l'anglais concernés par cette relation (ex : jour, année, soir, printemps, etc.). Nous n'avons pas inclus « soirée » pour cause d'ambiguïté (ex : « Je te souhaite une bonne soirée » : fête ou fin de journée ?).

(27) Il est parti hier. *=> Holmes : mod(parti, hier)*

=> Stanford : tmod(parti, hier)

D'autre part, les cas particuliers de « est-ce que » et « qu'est-ce que » ont dû être « traduits » en SD. Pour cela, nous nous sommes fondés sur l'analyse en dépendances de surface. Pour « est-ce que », nous avons conservé la même structure que celle fournie par

Holmes en appliquant les règles de transformation classique après l'analyse de surface (ex : la relation SD *dobj* remplace la sortie Holmes *obj*, etc) . Cependant, pour « qu'est-ce que », nous avons réorganisé la construction pour suivre le motif des SD au niveau de la relation entre un verbe copule et son attribut. Ainsi, « qu' » est analysé comme modifieur du verbe « est » par Holmes (mod(est, que), voir Figure 29). Au niveau de Stanford, puisque « est » est un verbe copule, nous considérerons « qu' » comme la tête de proposition. Toutes les relations qui étaient liées à « est » seront alors les dépendants de « qu' » (Figure 30). De plus, « ce » sera considéré comme gouverneur de l'équivalent en SD de la relation *mod_rel,* c'est-à-dire *rcmod,* (au lieu du verbe de la proposition principale). Ainsi, la structure de proposition relative sera respectée, le verbe de celle-ci étant ainsi lié à un syntagme nominal (ici un pronom) et non pas un verbe.

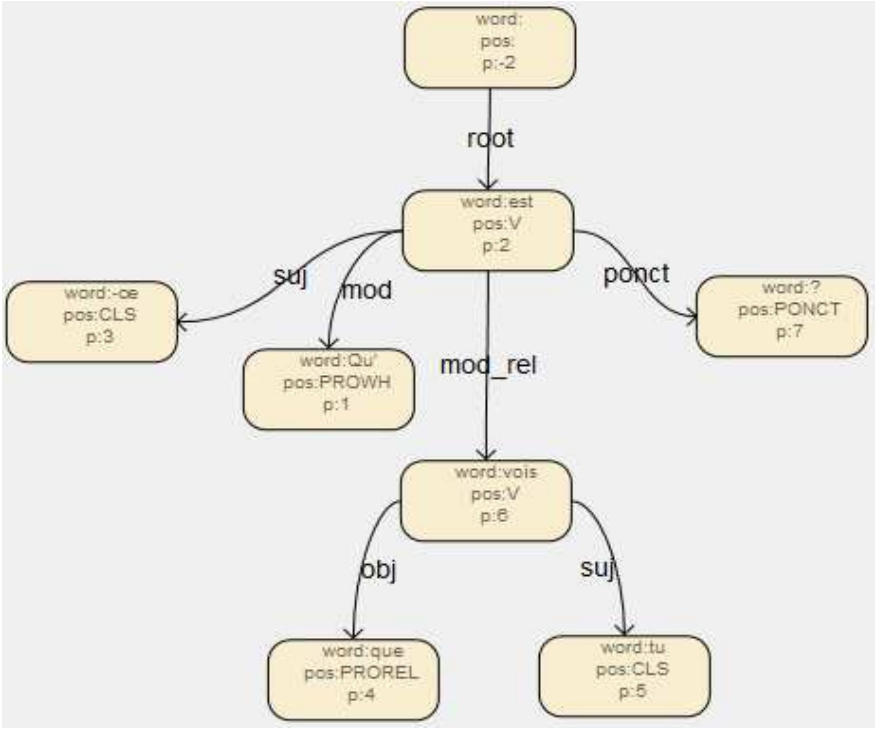


***Figure 29** : analyse Holmes « Qu'est-ce que tu vois ? »*

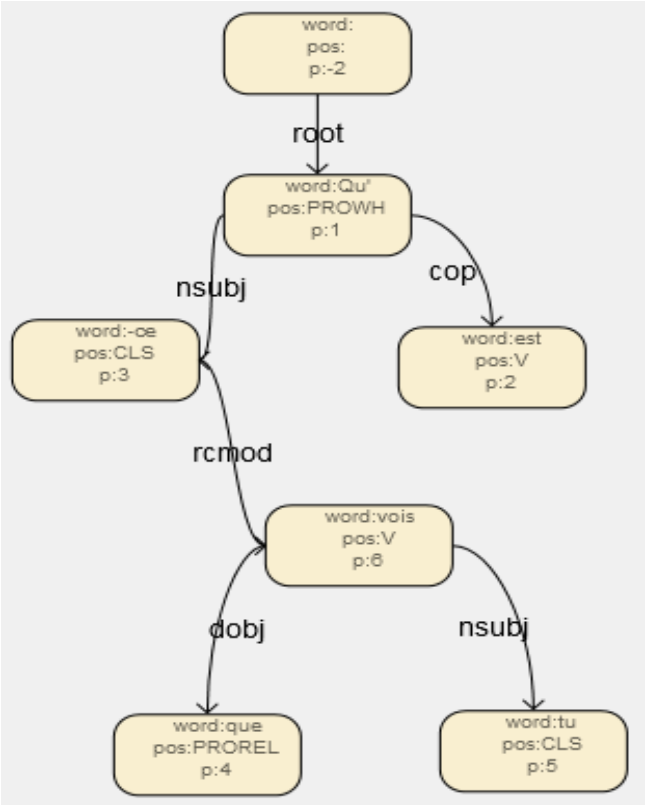


Figure 30 : *analyse après transformation « Qu'est-ce que tu vois ? »*

2.2 Difficultés et choix liés à la structure du FTB

Les différences de structures d'analyses entre les dépendances de surface du FTB et les dépendances profondes de Stanford entraînent un certain nombre de modifications à apporter.

Une restructuration des relations de Holmes a été par exemple effectuée dans le cadre des propositions subordonnées. En effet, quatre types de propositions sont différenciés par Stanford (autrement dit impliquant des dépendances différentes).

Pour les *relative clauses* (exemple 28), les *adverbial clauses* (exemple 29) et les *noun clauses* introduites par un pronom interrogatif (exemple 30), l'équivalent français analysé par Holmes possède une structure équivalente et nécessite uniquement un traitement au niveau des noms des relations,

(28) He is the man who lives in Japan.

=> Stanford : rcmod(man, lives), nsubj(lives, who)

(29) I don't know when it happened.

=> Stanford : advcl(know, happened), advmod(happened, when)

(30) I don't know who he is. *=> Stanford : ccomp(know, is), dobj(is, who)*

(31) I know that you are tall.

=> Stanford : ccomp(know, tall), complm(tall, that)

(32) I like sugar whereas you don't.

=> Stanford : advcl(like, do), mark(do, sugar)

L'interprétation diffère pour les deux autres : les subordonnées complément d'objet, introduites par « que » (*noun clauses* introduites par *« that » ou « whether »*, exemple 31) et les propositions adverbiales (*adverbial phrases,* exemple 32). En effet, en analyse profonde, pour ces deux cas, les têtes de propositions sont liées par une relation, et la tête de la deuxième proposition est dans une relation de dépendance avec le pronom/la conjonction. Dans Holmes cependant, si l'on considère les équivalents français, le lien est établi entre la tête de la première proposition et le pronom/conjonction, puis entre ce dernier et la tête de la deuxième proposition (exemples 33 et 34). Le niveau de dépendance a dû ainsi être adapté.

(33) Je dors parce que j'ai sommeil.

=> Holmes : obj(dors, parce que), obj(parce que, ai)

=> Stanford : advcl (dors, ai), mark(ai, parce que)

(34) Il est content que tu viennes.

=> Holmes : ats(est, content), mod(est, que), obj(que, viennes)

=> Stanford : cop(content, est), ccomp(content, viennes),

complm(viennes, que)

Holmes autorise une relation *ats* (attribut du sujet) entre un verbe copule et une préposition (exemple 35) ou une conjonction (exemple 36).

(33) Ce livre est à vous.

=> Holmes : ats(est, à), obj(à, vous)

=> Stanford : prep(est, à), pobj(à, vous)

(34) Il me semble que tu le sais déjà.

=> Holmes : ats(semble, que), obj(que, sais)

=> Stanford : ccomp(semble, sais), complm(sais, que)

Dans ces cas précis, l'analyse de Stanford conserve le verbe comme tête de proposition, même s'il est copule, et par conséquent ne crée pas de relation *cop*. Nous nous sommes donc alignés sur cette dernière interprétation (exemples 33 et 34 sur l'analyse de Stanford).

Une autre différence sépare les deux modèles de dépendances au niveau de la finesse d'analyse : celle des compléments d'objet pronoms. Stanford les sépare en deux catégories : les *iobj* (exemple 35) et les *dobj (*exemple 36*)*. Dans la sortie de Holmes, les pronoms « lui », « le », « la », « nous », « vous », et « leur » sont différenciés entre *a_obj* (pour l'objet indirect) et *obj (pour l'objet direct)*. La transformation s'effectue alors simplement (*a_obj* devenant *iobj* et *obj* devenant *dobj)*.

(35) Il lui parle. *=> Holmes : a_obj(parle, lui)*

=>Stanford : iobj(parle, lui)

(36) Elle le garde. *=> Holmes : obj(garde, le)*

=> Stanford : dobj(garde, le)

Quant aux les clitiques « me », « te », « se », « en » et « y » en tant que clitiques figés, ils sont tous analysés par Holmes avec la relation *aff,* sans différentiation objet direct-indirect. Pour les clitiques réfléchis, (« me », « te », « se » dans des tournures pronominales), nous avons décrit une relation créée spécialement (chapitre 7 1.1). Pour les autres tournures, nous avons effectué de nombreux tests et déduit des comportements. Nous avons ainsi conclu que lorsque le verbe est transitif direct (et uniquement direct), les clitiques « me » et « te » sont des objets directs (exemple 37).

(37) Il t'influence. *=> Holmes : aff(influence, t')*

=> Stanford : dobj(influence, t')

Pour les verbes transitifs indirects (et uniquement indirects), (exemple 38), ou transitifs directs et indirects (exemple 39), les clitiques « me » et « te » sont au contraire toujours les objets indirects.

(38) Il m'obéit. *=> Holmes : aff(obéit, m')*

=> Stanford : iobj(obéit, m')

(39) Il me la donne. *=> Holmes : aff(donne, me)*

=>Stanford : iobj(donne, me)

Les cas de « en » et « y » sont un peu plus complexes. Nous avons détaillé et traité chacun des cas. En effet, « y » a un sens locatif lorsqu'il est en relation avec un verbe copule (exemple 40), un verbe uniquement transitif (exemple 41) ou un verbe intransitif (exemple 42). Nous le transformons alors en *advmod*. Dans les autres cas, il sera *iobj*.

(40) Il y semble heureux. *=> Holmes : aff(semble, y)*

=> Stanford : advmod(heureux, y)

(41) Il y vis sa vie. *=> Holmes : aff(vis, y)*

=> Stanford : advmod(vis, y)

(42) Il y dors. *=> Holmes : aff(dors, y)*

=> Stanford : advmod(dors, y)

Le clitique « en » en relation avec un verbe copule sera adapté en *pobj*, objet de la préposition (ex : « Il en est fier. » -> « il est fier de quoi ? »). Il est locatif (donc *advmod* dans Stanford lorsqu'il dépend d'un verbe intransitif (exemple 43), mais objet direct (donc *dobj*) avec les verbes uniquement transitifs (exemple 44), et *iobj* dans les autres cas (exemple 45). Nous verrons au chapitre 8 comment nous avons transmis la caractéristique de transitivité dans les règles.

(43) J'en viens. *=> Holmes : aff(viens, en)*

=> Stanford : advmod(viens, en)

(45) Il en rêve. *=> Holmes : aff(rêve, en)*

=> Stanford : dobj(rêve, en)

(46) Il s'en souvient. *=> Holmes : aff(souvient, en)*

=> Stanford : iobj(souvient, en)

Par ailleurs, Holmes traite les cas d'articles contractés comme « du », « des », « au » ou « aux » comme constitués d'une préposition « de » ou « à » et d'un déterminant de lemme « le » ou « les » (et de forme vide : « »). Nous avons choisi de supprimer ces déterminants inexistants dans le nouveau modèle afin de ne conserver que les termes composant réellement la phrase. Ces articles seront alors considérés comme de simples prépositions (figures Figure 31 et Figure 32).

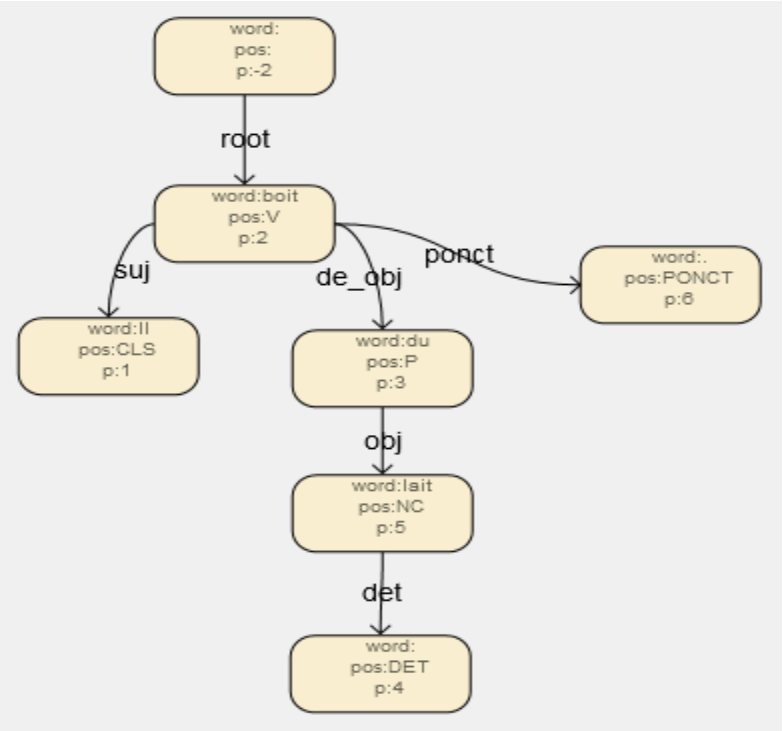


Figure 31 : *analyse Holmes de « Il boit du lait »*

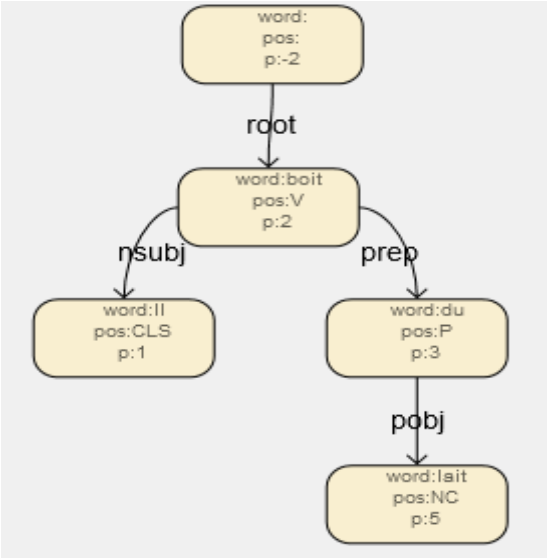


Figure 32 : *« Il boit du lait » après transformation*

2.3 Difficultés et choix liés aux particularités du *parsing*

Deux types de problèmes sont soulevés par le modèle statistique de Holmes.

D'une part, la désambiguïsation de cas particuliers au niveau de l'annotation du FTB ne se répercute pas toujours dans le *parsing* de Holmes. En effet, le FTB ayant été annoté manuellement, les décisions prises par les humains ne sont pas toujours reconnues par le modèle statistique qui ne se fonde pas sur ces critères « intellectuels ».

Par exemple, l'analyse du FTB différencie les interprétations de « dont » dans les propositions relatives en fonction du gouverneur [Candito *et al.,* 2011] (exemples 46 et 47). L'interprétation est différente.

(46) Le monsieur dont l'appartement est loué. => FTB : mod(appartement, dont)

(47) Le film dont il a vu la fin => FTB : de_obj(fin,dont).

Cependant, en pratique après le *parsing* de Holmes, toutes les interprétations de « dont » sont du même format, considérant toujours le verbe (ou l'attribut) de la relative comme gouverneur, dans une relation *de_obj*. Nous ne pouvons corriger et désambiguïser à ce stade les différentes interprétations de « dont ». Nous avons donc choisi de seulement adapter la relation de Holmes dans l'équivalent *pobj* de Stanford (exemple 48).

(48) Le film, dont il a vu la fin, … *=> Holmes : de_obj(vu, dont)*

=> Stanford : pobj(vu, dont)

De même, l'analyse de Holmes ne permet pas de différencier les compléments du nom, définis par une relation sous-spécifiée *dep*, d'autres compléments analysés dans la même relation : par exemple le complément circonstanciel de lieu (exemple 49) contre le complément du nom (exemple 50). Le complément du nom devra alors être considéré comme n'importe quelle relation de préposition *prep*, au lieu du *possessive* de Stanford.

(49) C'est un porte-clé de Paris. *=> Holmes : dep(porte-clé, de)*

=> Stanford : prep(porte-clé, de)

(50) C'est le porte-clé de Marie. *=> Holmes : dep(porte-clé, de)*

=> Stanford : prep(porte-clé, de)

Par ailleurs, ce modèle statistique implique des interprétations de relations multiples pour un même type de structure. Un simple changement de terme peut ainsi engendrer une relation différente (exemples 51 et 52). Il faudra donc tenter de couvrir tous les cas possibles à travers les règles de transformation.

(51) Il me propose de venir. *=> Holmes : obj(propose, de)*

Il me dit de venir. *=> Holmes : de_obj(dit, de)*

(52) Il me donne une chaise. *=> Holmes : aff(donne, me)*

Il me donne un abricot. *=> Holmes : a_obj(donne, me)*

3. Listes des relations du modèle final français

Voici le résultat de la transformation du modèle de dépendances de Stanford après adaptation au français (Figure 33).

root - *root*
dep - *dependent*
auxpass - *passive auxiliary*
cop – *copula*
acomp - *adjectival* complement
attr – *attributive*
pcomp – *prepositional complement*

ccomp - *clausal complement with internal subject*
xcomp - *clausal complement with external subject*
complm - *complementizer*
dobj - *direct object*
iobj - *indirect object*
pobj - *object of preposition*
refl – *reflexive pronoun object*
mark - *marker (word introducing an advcl)*
nsubj - *nominal subject*
nsubjpass - passive nominal subject
cc - *coordination*
conj - *conjunct*
mod - *modifier*
amod - *adjectival modifier*
advcl - *adverbial clause modifier*
purpcl - *purpose clause modifier*
det - *determiner*
infmod - *infinitival modifier*
partmod - *participial modifier*
advmod - *adverbial modifier*
neg - *negation modifier*
rcmod - *relative clause modifier*
quantmod - *quantifier modifier*
nn - *noun compound modifier*
tmod - *temporal modifier*
num - *numeric modifier*
number - *element of compound number*
prep - *prepositional modifier*
poss - *possession modifier*
compar – *comparative modifier*
punct - *punctuation*

Figure 33 : *Liste des Stanford dependencies pour le français*

La première étape fut l'analyse des dépendances et la prise de décision concernant les relations à conserver, celles à modifier, etc. A présent, nous allons nous intéresser à l'implémentation des règles permettant la transition de l'ancien système vers le nouveau.

Chapitre 8 - Conversion de l'analyse syntaxique : les règles

L'écriture des règles est la phase finale pour la conversion des relations. Elle nécessite elle aussi une phase de tests, et implique des choix.

1. Outils de tests

De nombreux outils nous ont aidés dans cette phase de transformation. En effet, la gestion d'un grand nombre de règles peut impliquer des interactions non souhaitées si leur portée n'a pas été testée.

1.1 Scientext[29] et ScienQuest[30]

Dans l'optique de proposer une plateforme web (nommée ScienQuest) utilisable par toute personne non spécialiste de la linguistique computationnelle, le projet Scientext permet l'interrogation d'un corpus constitué de textes scientifiques, constituant 4,8 millions de mots au total pour la partie française [Falaise *et al.,* 2011].

Plusieurs modes sont disponibles en fonction de la précision des requêtes d'interrogation. Dans le cadre de notre travail, nous avons utilisé le mode « libre », qui permet de rechercher des expressions dans le corpus scientifique mis à disposition, avec la possibilité de préciser des contraintes sur chacun des mots composant l'expression recherchée. Il est alors possible de spécifier le lemme, la forme et/ou la catégorie grammaticale (plus ou moins précise : ex : verbe -> infinitif, conjugué, etc.). Les mots peuvent être considérés comme côte à côte, ou bien comme reliés par une relation syntaxique à préciser dans une liste disponible. Le résultat s'affiche dans un concordancier[31].

Cet outil nous a ainsi permis de tester la portée de certaines règles, pour s'assurer que celles-ci ne provoquent pas de surgénération sur d'autres cas semblables (par exemple même catégories grammaticales ou même relations syntaxiques) mais dont l'interprétation en dépendance est différente.

Cependant, le corpus étant composé uniquement d'écrits scientifiques, les résultats furent parfois trop spécifiques, voire inexistants.

[29] http://scientext.msh-alpes.fr/scientext-site/spip.php?article1, dernière consultation : 08/09/13
[30] http://scientext.msh-alpes.fr/scientext14/?tab=search, dernière consultation : 08/09/13
[31] Voir Annexe 7 (p.113)

1.2 Emobase[32]

La plateforme Emobase permet d'interroger un corpus constitué pour le projet Emolex[33]. L'objectif de ce projet était d'étudier l'expression des émotions dans cinq langues (français, allemand, anglais, espagnol et russe) à travers des outils permettant d'interroger le corpus sous différents angles (sémantique, discursif, syntaxique, etc.). L'application EmoConc de la plateforme Emobase permet d'interroger le corpus dans la langue souhaitée, et fournit entre autres un concordancier regroupant les résultats de la requête.[34] Il est ainsi possible de rechercher une combinaison de termes, liés par une relation syntaxique ou non, avec plus ou moins de critères de précision (lemme, traits morphologiques, etc.)[35]. Deux corpus sont disponibles pour le français, l'un comportant plus de 124 millions de termes, et l'autre plus de 136 millions.

Cet outil nous a aidé, comme pour ScienQuest, à vérifier la portée des règles afin d'éviter les surgénérations. Les intérêts de cette plateforme ont été doubles pour nous. D'une part, le caractère non spécifique des corpus à interroger permet d'obtenir des résultats variés, couvrant un plus grand nombre de possibilités que ScienQuest. D'autre part, un corpus de plus de 128 millions de mots est également mis à disposition pour l'anglais. Nous avons alors pu rechercher l'équivalent de structures de cas particuliers du français en anglais, pour ensuite tester l'analyse de Stanford sur ces phénomènes, et donc connaître la transformation à effectuer (ex : la double préposition : Il voyage partout sauf en Asie. -> *He travels everywhere except in Asia.*).

Cependant, de nombreux problèmes ont été rencontrés. D'une part, il n'est pas possible de limiter les résultats à un nombre maximum. Par conséquent, au vue de la quantité de données, une requête entraînant un trop grand nombre de résultats crée une erreur et ne renvoie aucun résultat. Des erreurs d'analyse syntaxique ont par ailleurs posé problème, de nombreux mots se retrouvant en dehors de toute relation. Certaines requêtes n'ont alors pas abouti en raison d'un critère syntaxique qui avait été spécifié dans la requête.

[32] http://emolex.u-grenoble3.fr/emoBase/ dernière consultation : 08/09/13
[33] www.emolex.eu dernière consultation : 08/09/13
[34] Voir Annexe 9 (p.115)
[35] Voir annexe 8 (p.114)

1.3 Linguee[36]

Enfin, dans notre objectif d'adapter les relations de Stanford au français, nous avons utilisé l'outil Linguee qui nous a permis de rechercher des traductions d'expressions françaises. Linguee est une plateforme qui met en parallèle des textes traduits en plusieurs langues, textes provenant de sources web. Il indique lorsque la traduction est de sources sures (sites gouvernementaux, etc.) ou s'il est possible qu'elle ne soit pas exacte. De plus, le moteur de recherche est performant. Il est ainsi possible de rechercher une expression en spécifiant seulement des mots clés, en conjuguant le verbe ou bien en le laissant à l'infinitif, sans pour autant limiter les résultats. Ce site nous a ainsi permis de déterminer l'équivalence (ou non) de structures spécifiques entre le français et l'anglais en cas de doutes.[37]

2. La création des B-rules

2.1 Choix

Le choix de l'ordre d'application des règles a été d'une grande importance tout au long de l'implémentation. En effet, cette méthode permet de traiter les cas les plus spécifiques avant les autres. Par exemple, la différence entre la relation *acomp* (qui relie les verbes non copules à l'adjectif qui dépend d'eux) et la relation *xcomp* est la présence d'un objet (exemple *53*) ou non (exemple 54) dans la sortie de Holmes. Ainsi, la règle de *xcomp* s'applique avant celle de *acomp*.

(53) Je l'ai mangé froid. *=> Holmes : obj(mangé, l'), mod(mangé, froid)*

=> Stanford : xcomp(mangé, froid), nsubj(froid, l')

(54) Il tombe malade. *=> Holmes : mod(tombe, malade)*

=> Stanford : acomp(tombe, malade)

Sur un autre plan, nous avons dû gérer la possibilité d'enchainer les verbes modaux, particularité de la syntaxe française (ex : « Il doit pouvoir le faire. »). Cependant, la limite du langage est seulement située là où la phrase n'est plus intelligible par l'humain. Or, le modèle de Stanford ne considère pas ces verbes comme têtes de phrase,

[36] www.linguee.fr dernière consultation : 08/09/13
[37] Voir Annexe 10 (p.116)

au profit du premier verbe « plein » de la suite. Nous avons décidé de nous arrêter au traitement d'une suite de maximum deux auxiliaires modaux (Figure 34), considérant les cas de succession de trois d'entre eux ou plus beaucoup moins fréquents.

```
/*
        Stanford relation aux
        Succession of two modals and one non modal infinitive verb.
        Other examples : "Il pourrait avoir été mangé."
*/
6, aux1 ([1 pos=_v vclass=MODAL][2 pos=_vbis vclass=_mod][3
pos=_vter])(1-2[type=obj] 2-3[type=obj]) =>
([1][2][3])(3-1[type=aux] 3-2[type=aux])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS")}
{TestStaticFunctions.regxp(vbis,"VPP|VINF")}
{TestStaticFunctions.regxp(vter,"VPP|VINF")}
{TestStaticFunctions.regxp(mod,"MODAL|AUX")}
).
test:"Il devrait pouvoir manger."
```

***Figure 34** : Règle de transformation d'une succession de trois verbes*

Similairement, nous avons dû établir une limite au niveau des nombres. En effet, Stanford relie les composants d'un nombre entre eux (figures Figure 35 et Figure 36) par la dépendance *number*, alors que le FTB les interprète tous comme les déterminants du nom auquel ils se rapportent. Afin de limiter les règles régissant ces cas particuliers, nous nous sommes restreints à une succession de maximum trois composants dans le nombre, considérant que ceux comportant quatre composants ou plus sont le plus souvent traduits en chiffres.

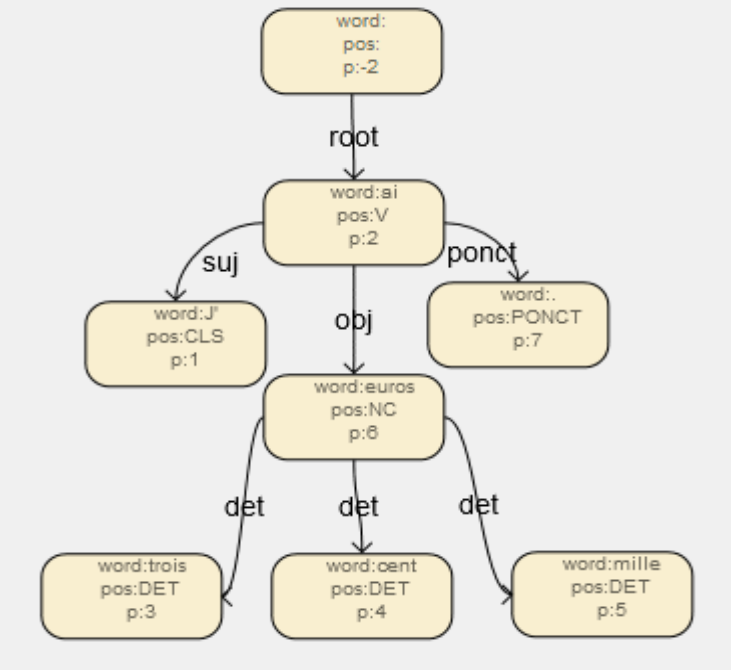


***Figure 35** : Analyse Holmes de « J'ai trois cent mille euros. »*

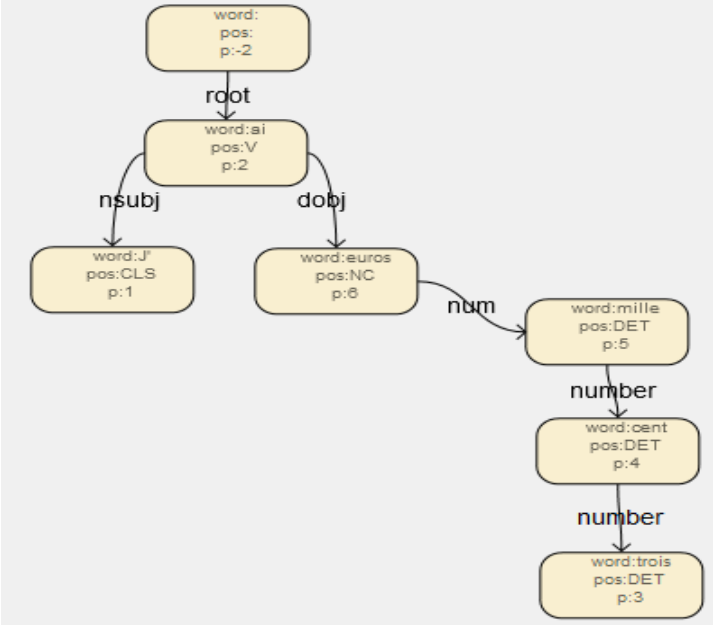


***Figure 36 :** « J'ai trois cent mille euros » après transformation*

Toujours sur le traitement des nombres, plusieurs décisions ont dues être adoptées. Stanford différencie le nombre qui qualifie le nom (relation *num*) des composants du nombre (relation *number*) (Figure 36). Il faut alors déterminer dans un nombre complexe lequel prendre comme modifieur du nom. Le plus optimal serait de considérer le plus grand des composants (ex : « millions » dans « trois millions », « cent » dans « cent cinquante-trois »). Cependant, dans le temps imparti, nous n'avons pas pu

mettre en place une méthode réalisant ce traitement. Nous avons donc fait le choix de considérer le composant du nombre qui est immédiatement avant le nom comme celui à inclure dans la relation *num*. Pour cela, nous avons spécifié la condition suivante : la position du composant du nombre dans la phrase (« position ») doit correspondre à la position précédent celle du nom (« positionbis») (autrement dit, position==positionbis-1) (Figure 37)

```
/*
        Stanford relation num
*/
168, num
([1 pos=_detAdj numType=CARD p=_position]
[2 pos=NC numType=_notCard p=_positionbis])
(2-1[type=_detmod])
=>
([1][2])(2-1[type=num])
:=
({TestStaticFunctions.regxp(detAdj,"DET|ADJ")}
{(position==positionbis-1)}
{!TestStaticFunctions.regxp(notCard,"CARD")}
{TestStaticFunctions.regxp(detmod,"det|mod")}
).
test:"J'ai trois euros dans ma poche."
```

***Figure 37** : Règle du traitement de la relation « num »*

Nous avons en outre fait le choix de supprimer automatiquement une relation résultant d'une erreur d'analyse de Holmes : la relation « root » lorsqu'elle lie la racine et une ponctuation. En effet, ce phénomène se produit lorsque l'analyse de la phrase par Holmes est erronée : le fil de la phrase est cassé, et la racine est celle qui permet encore de relier chacun des morceaux de la phrase. Cependant, les ponctuations doivent être supprimées dans la sortie de Stanford. En effet, les dépendances de Stanford contiennent la relation *punct* (équivalent de *ponct* dans Holmes), mais il est précisé dans le manuel [Candito *et al.,* 2011] que celle-ci n'est en réalité pas utilisée par défaut. Nous tirons parti de cette opportunité pour supprimer une erreur, en retirant de la sortie en dépendances de Stanford toute relation *root* contenant une ponctuation.

D'autre part, pour nous permettre de réaliser les traitements sur les verbes uniquement transitifs et ceux intransitifs, nous avons défini des « *gazeteers* » comprenant les listes des verbes concernés par ces propriétés. Ces verbes ont été tirés du Lefff[38], le Lexique des Formes Fléchies du français. Il donne, entre autre, la structure argumentale des formes fléchies des verbes, de laquelle nous pouvons déduire le caractère transitif, ou non, du verbe. Cependant, l'ambiguïté de certains verbes implique plusieurs possibilités au niveau de la transitivité. Pour les listes de verbes uniquement transitifs ou intransitifs, nous avons donc relevé uniquement les termes non-ambigus. Un « *gazeteer* » a été également constitué pour indiquer les nombres (en chiffre ou en lettres) pour la mise en place des règles *num* / *number*. Ainsi, un trait CARD a été ajouté aux termes concernés.

2.2 Difficultés/problèmes

Certaines difficultés ont été rencontrées lors de la création des règles, qu'elles soient dues aux différences entre les deux modèles de dépendances ou bien au niveau de l'implémentation des règles en elle-même.

Nous avons dû par exemple prendre en compte le fait que, dans Stanford, la tête de la phrase est l'attribut en cas de verbe copule (exemple 55), ou l'auxiliaire non modal lors de conjugaisons complexes (exemple 56) contrairement à Holmes. Nous avons donc été contraints d'effectuer des traitements spécifiques pour les relations qui associent, dans Holmes, le verbe à un autre terme de la phrase. La complexité réside surtout dans les cas de propositions, où nous devons prendre en considération le fait que chacune des deux propositions peut être amenée à changer de tête. De nombreuses règles (environ 1/3 du nombre total de règles) ont été alors ajoutées dans cet unique but (Figure 38).

(55) Il est blond comme son frère => Holmes : mod(est, comme)

=> Stanford : prep(blond, comme)

(56) Il a pu manger. => Holmes : suj(pu, il)

=> Stanford : nsubj(manger, il)

[38] Alpage.inria.fr/~sagot/lefff.html

```
/*
Stanford relation nsubj
Cases of auxiliary or copula (change of head)
Other example : "Il est efficace."
*/
28, nsubj_auxcop ([1][2][3])(3-2[type=_auxcop] 2-1[type=suj])
 =>
([1][2][3])(3-2[type={auxcop}] 3-1[type=nsubj])
:=
({TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il a pu manger."
```

Figure 38 : *Règle de changement de gouverneur pour le sujet*

Deux messages persistent à la compilation des règles. Ce ne sont que des alertes car les règles qu'elles visent s'appliquent malgré tout.

L'une concerne la comparaison de variables entre la partie gauche et la partie droite de la règle. En effet, pour regrouper au maximum les règles, nous avons traité pour chaque relation concernée le changement de tête de phrase en cas de verbe copule (ajout de relation *cop*) et de verbes modaux (ajout de relation *aux*) ensembles. Ainsi, nous avons déclaré une variable *_auxcop* pour désigner la présence soit de la relation *cop*, soit de la relation *aux* dans la partie gauche, relation qui devra être répétée dans la partie droite afin de ne pas la supprimer. Etant donné que nous ne pouvons prédire laquelle des deux relations sera mise en correspondance avec la règle lors de l'analyse, nous répétons la valeur de la variable *auxcop* en partie droite (type={auxcop}) (voir partie rouge Figure 38. C'est à ce niveau-là que la compilation marque une alerte sur la syntaxe. Cependant, le traitement des règles concernées ne semble pas affecté.

La deuxième porte sur la suppression des ponctuations, comme vu précédemment en 2.1. Cependant, pour éliminer cette relation lors de l'analyse, nous avons dû créer une règle dont la partie droite contenant les types de relation est vide. Le compilateur émet alors une alerte, mais la ponctuation est bien supprimée (Figure 39).

```
/*
        Deletion of Holmes relation ponct
*/
194,ponct([1][2 pos=PONCT])(1-2[type=ponct])=>
([1])().
test:"Voilà."
```

Figure 39 : *Règle de suppression de la relation « ponct »*

Enfin, nous avons noté un problème avec la règle traitant la relation *partmod* qui ne fonctionne pas avec les participes passés (exemple 57). Effectivement, il y a un conflit entre cette règle et la règle *ccomp3* qui relie les têtes de deux propositions apposées (exemple 58). La même relation *mod* relie la tête de proposition avec le participe passé, qui est dans un cas la tête de la deuxième proposition, et dans l'autre un simple modifieur isolé. L'ordre des règles ne changera rien au problème car si la règle *partmod* est placée en premier, la règle ccomp3 ne s'appliquera pas, même si deux propositions (dont une avec participe passé) sont apposées. Aucune solution n'a donc pour le moment été trouvée.

(57) Séparés, les enfants étaient tristes.

=> Holmes : mod(étaient, Séparés)

=> Stanford (théorique) : partmod(tristes, séparés)

=> Stanford (réel) : ccomp(tristes, séparés)

(58) Il est triste, il a perdu.

=> Holmes : mod(est, perdu)

=> Stanford (théorique et réel) : ccomp(triste, perdu)

Certains choix ont été adoptés, motivés par des difficultés rencontrées, alors que certains problèmes n'ont pas encore été résolus. Nous allons maintenant proposer des améliorations à apporter à différents niveaux.

3. Améliorations possibles

Le travail effectué durant ce stage est le premier à expérimenter le nouveau système de règles *B-rules*. Des modifications sont alors à envisager par la suite pour optimiser le traitement des règles.

En effet, l'exécution de l'ensemble des règles du module est particulièrement lent. Une amélioration majeure est alors à prévoir dans la prochaine version des *B-rules*. Nous avons constaté qu'une règle faisant intervenir une contrainte via une fonction Java est environ quatre fois plus lente au chargement qu'une règle ne contenant que des attributs renseignés directement dans la règle (rappel Figure 22 et Figure 23). Une optimisation à ce niveau est déjà envisagée dans un futur proche.

De plus, une importante diminution du nombre de règles (actuellement au nombre de 100) apporterait une nette augmentation de rapidité d'analyse. Une possibilité à envisager dans ce sens serait le traitement des changements de gouverneur (dus aux verbes copules et conjugaisons complexes) non pas au niveau de chaque cas, comme effectué présentement (rappel Figure 38), mais à la fin du module, une fois toutes les transformations effectuées, grâce à une règle passant en revue à elle seule l'ensemble des cas. Cela aurait pour effet de regrouper 32 règles, soit un tiers, en une seule (ou légèrement plus si des cas particuliers ne peuvent pas être couverts par cette règle).

En outre, plusieurs points peuvent être examinés au niveau de l'efficacité des règles en elles-mêmes.

Tout d'abord, le traitement des *dobj* et *iobj*, ainsi que l'interprétation des « en » et « y » devraient faire l'objet de recherches plus approfondies afin de fournir des résultats plus fiables. En effet, ces cas sont déterminés par le caractère transitif (direct ou non) ou intransitif du verbe auquel ils se rapportent. Or, l'ambiguïté au niveau des verbes implique une incertitude quant à la transitivité, provoquant ainsi des erreurs d'interprétation. L'idéal, mais peu réalisable, serait alors l'annotation de la transitivité directement dans les arbres du FTB, le contexte permettant ainsi de fournir ces données exactes qui seront apprises par le modèle statistique.

D'autre part, le traitement des *num* et *number* n'est pas optimal. En effet, un simple adjectif entre le nombre et le nom empêche toute la chaine de transformation, les

deux termes n'étant alors plus adjacents. Il faudrait donc pouvoir choisir le terme (au trait CARD) inclus dans une relation *det* avec le nom, qui soit le plus proche de celui-ci (mais pas obligatoirement à côté de lui). De plus, nous avons remarqué que dans la relation *num*, l'analyse devrait considérer le composant du nombre le plus grand pour devenir optimale (ex : « cent » dans « cent trente-trois »). Une méthode Java pourrait alors être développée afin de réaliser les opérations nécessaires : comparaison et classement dans un ordre croissant (ou décroissant) de nombres en chiffres et en lettres, en tenant compte des possibles mélanges de format (ex : « 3 millions » : millions > 3). Une autre possibilité, mais qui nous éloignerait alors de l'analyse proposée par Stanford, serait d'agir sur la tokenisation pour regrouper tous les composants d'un nombre, et le considérer ainsi dans son ensemble. Pour l'instant cependant, il est difficile d'effectuer des modifications à ce niveau-là de la chaîne de traitement sans engendrer des erreurs d'analyse aux étapes suivantes.

Enfin, des recherches linguistiques approfondies pourraient être menées de pair avec l'équipe des SD afin de déterminer les termes qui pourraient équivaloir à ceux des relations *predet, preconj, attr, nwe, quantmod* et *npadvmod* que nous n'avons pas conservés, pour manque de précision (voir chapitre 7 1.2)

Le développement du module de règles a permis d'exécuter la transformation du modèle actuel de Holmes vers un modèle suivant les dépendances profondes de Stanford, adaptées pour le français. Nous allons maintenant en réaliser une brève évaluation.

Chapitre 9 – Evaluation

1. Le test

Afin de tester et d'évaluer le module de règles développé dans ce projet, nous avons exécuté l'analyse sur un ensemble de textes courts, puis effectué les calculs de rappel et précision à partir de de la sortie de l'analyse de Holmes.

1.1 Les textes

Les textes sont tous tirés d'un site d'information belge (http://www.7sur7.be/)[39]. Celui-ci a été choisi comme source des tests car les textes qu'il propose sont variés et les constructions de phrases de taille raisonnable. Ce choix a été fait de telle sorte que l'analyse de Holmes soit elle-même la plus correcte possible, afin de tester les règles sur une base la plus valide possible.

Sur les trois textes analysés, nous avons comptabilisé 19 phrases de 27 tokens en moyenne (en moyenne 33 pour le texte 1, 27 pour le texte 2 et 18 pour le texte 3). Ils ont été sélectionnés pour la variété de structures de dépendances, et donc d'application de règles, qu'ils impliquent. Ils incluent en effet des propositions relatives, circonstancielles, participiales (participe passé et présent), infinitives, ainsi que des modifieurs variés (adjectifs, adverbes, nombres, expressions temporelles, etc.), des constructions passives, des verbes copules, etc. Cette variété de constructions nous permet alors de tester un bon nombre de règles, malgré la petite taille des textes. Nous avons seulement supprimé les titres afin de ne conserver que des phrases complètes, et les éventuels guillemets pour ne pas provoquer d'erreurs au niveau de l'analyse, la phrase à analyser devant elle-même être encadrée de guillemets.

1.2 La méthodologie

Nous insistons sur le fait que le module de règles n'implique que des transformations de dépendances, et ne constitue donc en aucun cas un parser à lui seul. L'objectif de ces tests est donc d'évaluer l'efficacité du module en lui-même, et non pas de la justesse générale du parsing. Nous souhaitons ici mettre en évidence les erreurs engendrées directement par le module, et non pas celles qui ne dépendent pas de lui, mais de l'analyse de Holmes. Nous avons donc établit la méthodologie comme suit.

Premièrement, nous avons réalisé l'analyse de chacun de ces textes par Holmes, phrase par phrase. Après avoir stocké les résultats obtenus en dépendances de surface, nous avons exécuté le module de règles de transformation pour chacune des phrases. Nous avons ensuite comparé à la main chacune des relations de Holmes avec les relations transformées, et déterminé lorsque la transformation était erronée par rapport aux

[39] Détail des textes annexes 3,4 et 5 (p.109, 110, 111)

relations de surface. Nous n'avons donc pas comptabilisé les erreurs par rapport à une analyse correcte, mais en se basant sur la sortie de Holmes et ses possibles erreurs.

2. *Calculs et résultats*

Au niveau des calculs, nous avons effectué le rappel (le nombre de relations transformées sur le nombre de relations total) et la précision (le nombre de relations correctement transformées sur le nombre de relations total) de l'analyse obtenue après l'application des règles, à partir de la sortie d'analyse de Holmes. Des suppressions de relation (pour la ponctuation) ayant été réalisées après l'analyse en dépendances profondes, le nombre de relations totales a été calculé sur le nombre de dépendances à la sortie de Holmes, et ne sera donc pas le même après transformation.

Les calculs[40] (Tableau 1) ont montrés sans surprise un rappel de 100%. En effet, toutes les relations sont transformées au minimum en dépendance *dep*, comme vu précédemment, *dep* étant la relation la plus générale. Dans ce but, une règle transformant toute relation Holmes non transformée est appliquée. Nous notons par ailleurs la présence de ces relations sous-spécifiées, quatre d'entre elles ayant été recensées sur l'ensemble des textes (annexe 6).

D'autre part, la précision sur l'ensemble des trois textes est de 98%. Ce résultat très positif reste à relativiser au vu de la petite taille du corpus de test. Cependant, nous avons noté que les phrases contiennent une bonne variété de cas. De ce fait, nous pouvons d'ores et déjà affirmer que le module de règles est opérationnel, et efficace au moins sur des phrases de taille moyenne.

N° texte	Rappel	Précision	F mesure
1	100%	98.4%	99.2%
2	100%	99.1%	99.5%
3	100%	97.1%	98.5%
Total	100%	98.2%	99.1%

***Tableau 1** : Résultats de précisions et rappels pour chaque texte*

Les résultats ayant montré peu d'erreurs, nous allons alors nous intéresser en détails à celles-ci.

[40] Voir détail Annexe 6 (p.112)

3. *Analyse des résultats*

Certaines erreurs d'analyse sont dues à la présence d'erreurs déjà dans l'analyse de Holmes. Par exemple, dans le texte 1 dans la deuxième phrase, une erreur a été relevée au niveau du sujet. En effet, le verbe « inquiétée » étant un passif, nous aurions dû obtenir une relation *nsubjpass* avec le sujet « sélection ». Cependant, *nsubj* a été appliqué ici. Nous pouvons expliquer cette erreur par le fait que l'analyse de Holmes a créé une relation de sujet avec le syntagme nominal « 39ème » (« 139ème » a été segmenté en deux parties : « 1 » et « 39ème »). C'est cette relation qui a alors été transformée au mode passif (figures Figure 40 et Figure 41).

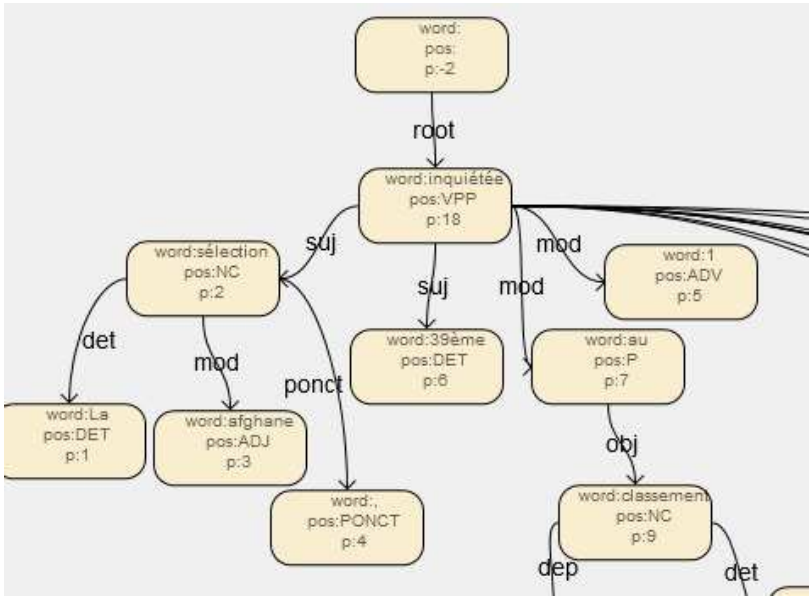


Figure 40 : *Analyse de Holmes pour : « La sélection afghane, 139ème au classement […], n'a jamais été inquiétée, […] »*

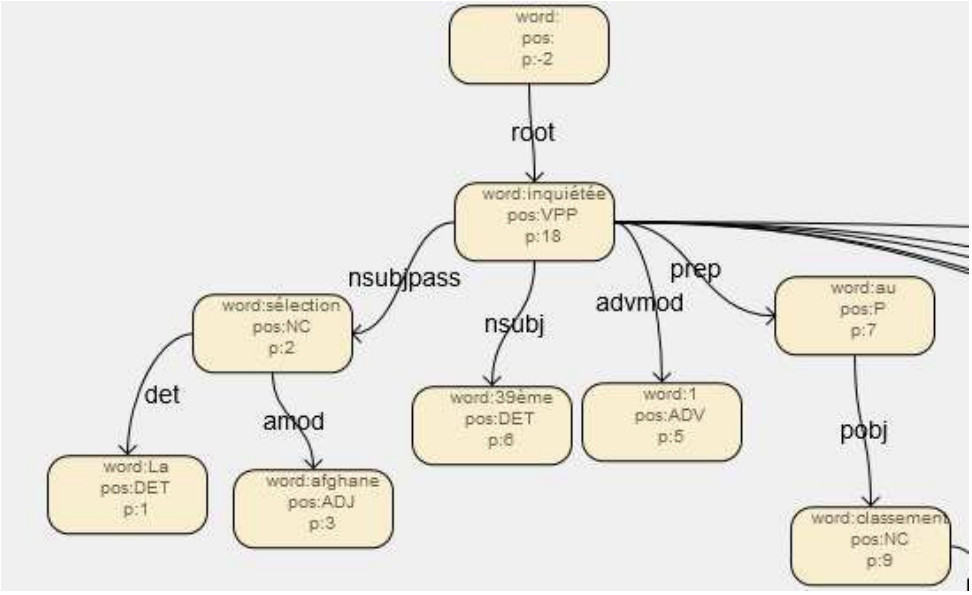


Figure 41 : *Analyse après transformation pour : « La sélection afghane, 139ème au classement […], n'a jamais été inquiétée, […] »*

D'autre part, une relation *amod* lie « stade » à « situé » en phrase 3 du même texte, au lieu de *partmod.* Cette relation a été obtenue car « situé » a été étiqueté en tant qu'adjectif par Holmes, et non pas en tant que participe passé, d'où l'erreur engendrée (figures Figure 42 et Figure 43).

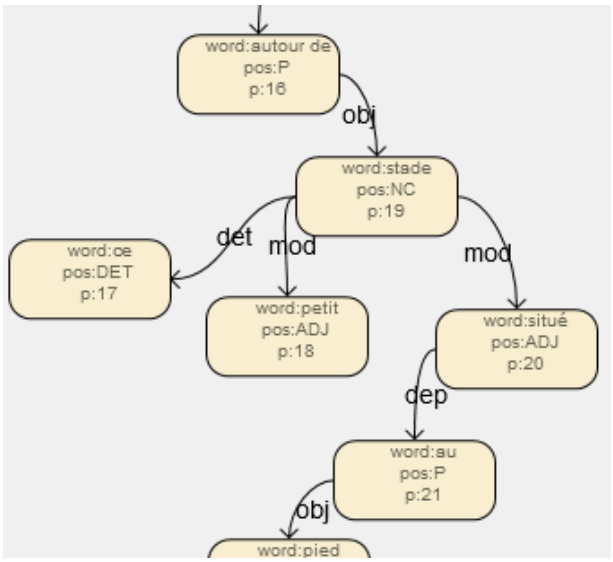


Figure 42 : *Analyse de Holmes pour : « […] autour de ce petit stade situé au pied des montagnes […] »*

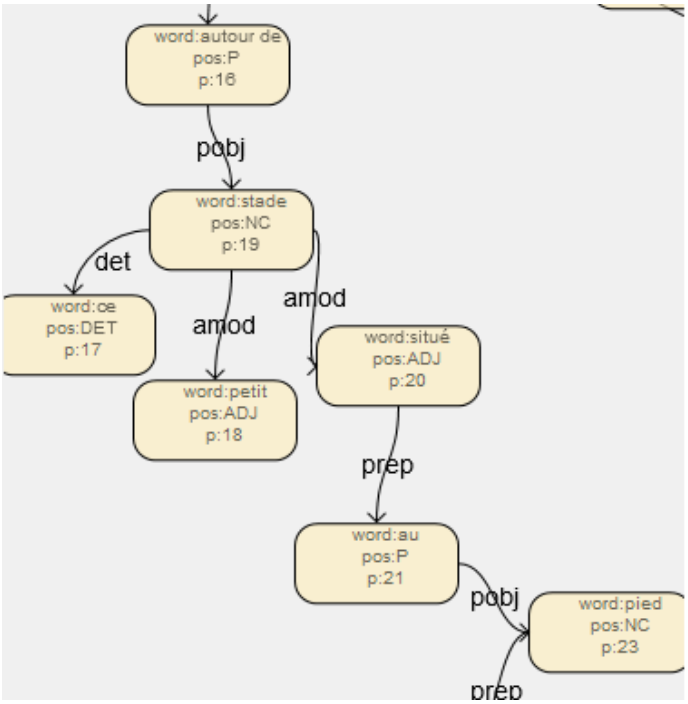


Figure 43 : *Analyse après transformation pour : « […] autour de ce petit stade situé au pied des montagnes […] »*

L'inverse est constaté par ailleurs au texte 3 phrase 3 : *partmod* joint « raison » à « indéterminée » (au lieu de *amod*), car ce dernier a été étiqueté comme participe passé au lieu d'adjectif. En outre, deux erreurs ont été relevées au niveau d'expressions temporelles. Dans le texte 2, phrase 1, une relation *nn* a été attribuée entre « contrôle » et « mercredi » et entre « chute » et « samedi » dans le texte 3 phrase1 (figures Figure 44 et Figure 45).

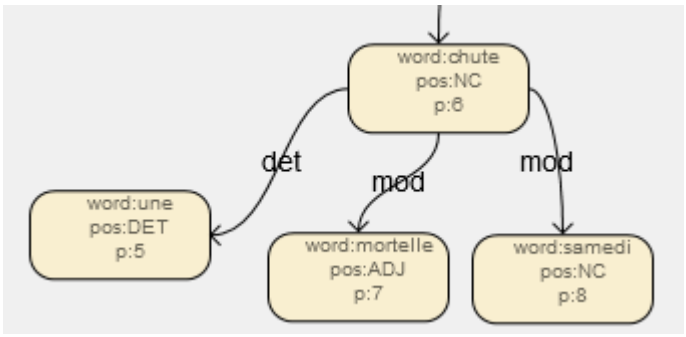


Figure 44 : *Analyse de Holmes pour : « Deux alpinistes ont fait une chute mortelle samedi[…] »*

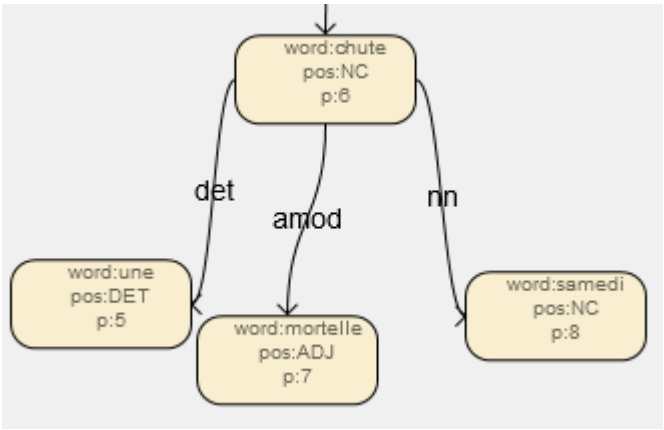


Figure 45 : *Analyse transformée pour : « Deux alpinistes ont fait une chute mortelle samedi[…] »*

L'absence de la relation *tmod* qui marque les expressions temporelles est due au fait que l'analyse de Holmes a établi la relation avec l'objet (qui est un nom commun : « contrôle » et « chute ») au lieu du verbe (« fait »).

Au niveau des erreurs non liées à l'analyse erronée de Holmes, nous remarquons, dans le texte 1 phrase 2, que la relation entre « buts » et « deux » est restée *det,* au lieu de se transformer en *num*. Cette erreur est due à la présence d'un mot (« autre ») entre le nombre et le nom, qui, comme nous l'avions mentionné en chapitre 8 2.2, empêche l'application de la règle qui est trop restreinte (figures Figure 46 et Figure 47).

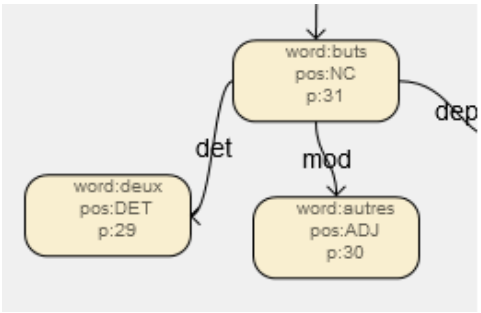


Figure 46 : *Analyse de Holmes pour :*
« […] deux autres buts […] »

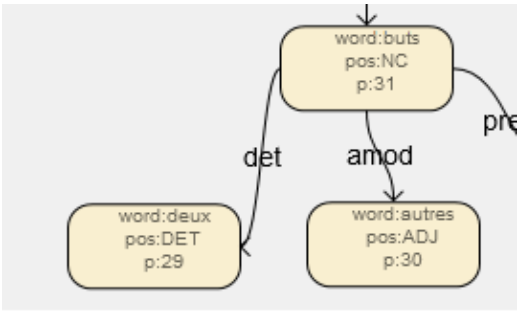


Figure 47 : *Analyse après transformation pour :*
« […] deux autres buts […] »

D’autre part, dans ce même texte, en phrase 3, la coordination des termes composant le sujet s’est effectuée au niveau des déterminants, lorsque l’analyse profonde a pour but de relier les mots pleins (figures Figure 48 et Figure 49. La coordination de l’analyse de surface s’effectuant ici au niveau des déterminants, aucune règle n’a été prévue pour modifier cette liaison.

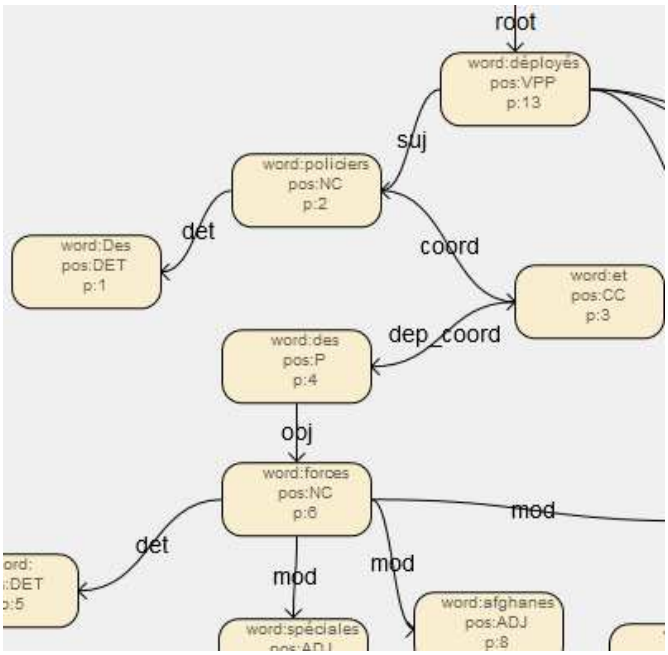


Figure 48 : *Analyse de Holmes pour : « Des policiers et des forces spéciales afghanes munis de boucliers […] »*

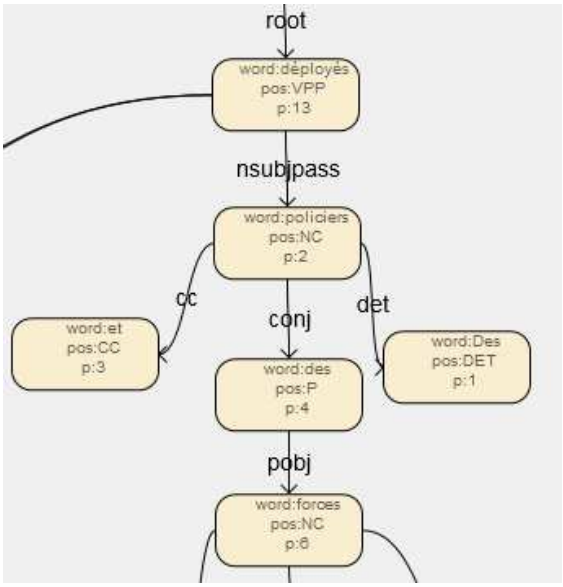


Figure 49 : *Analyse transformée pour :« Des policiers et des forces spéciales afghanes munis de boucliers […] »*

Les quatre relations sous-spécifiées sont également intéressantes à analyser, car sans être erronées, elles mettent tout de même en évidence l'absence de règles détectant une relation adaptée. Ainsi, la phrase 3 du texte 1 dégage un cas non pris en charge par les règles : la coordination de prépositions. C'est pour cette raison qu'une relation *mod* a été désignée entre « déployés » et « dans » (exemple 59).

(59) « […] étaient déployés dans et autour de ce petit stade. »

=> Stanford (observé) : **mod**(déployés, dans), cc(dans, et), conj(dans, autour de)

=> Stanford (théorique) : **prep**(déployés, dans), cc(dans, et), conj(dans, autour de)

De la même manière, le « que » de la négation dans le texte 3 phrase 5 est inclus dans une dépendance sous-spécifiée *advmod* avec le verbe, au lieu de la relation plus précise *neg* (exemple 60). Cette structure de phrase n'est donc pas prise en charge par les règles pour l'application de la relation *neg*.

(60) « les secours […] n'ont pu que constater […]. »

*=> Stanford (observé) : **advmod**(constater, que)*

*=> Stanford (théorique) : **neg**(constater, que)*

Dans le texte 2 phrase 6, nous remarquons également une relation *mod* entre « été et « qu' » au lieu de la relation plus précise *compar* qui aurait dû relier « nombreux » et « qu' » (exemple 61).

(61) « ceux-ci ont été beaucoup moins nombreux cette année qu'en 2012. »

*=> Stanford (observé) : cop(nombreux, été), **mod**(**été**, qu')*

*=> Stanford (théorique) : cop(nombreux, été), **compar**(nombreux, qu')*

De l'analyse de ces résultats, nous comprenons que l'étude des erreurs est tout aussi importante que l'étude des relations sous-spécifiées qui ont été générées. En effet, bien qu'elles ne soient pas erronées, elles mettent en évidence une structure de phrase non prévue par les règles, puisqu'elles ne s'appliquent pas sur des cas pourtant traités.

Chapitre 10 - Conclusion et perspectives

1. Conclusion

Ce travail s'inscrit dans une démarche relativement récente consistant à utiliser la méthode de transformation de graphes pour une conversion d'analyse de surface vers une analyse profonde. Nous n'avons répertorié à ce jour que deux outils uniquement consacrés à ce type de traitement du langage et utilisant ce procédé (Ogre et Grew Chapitre 6, 1.2 et 1.3). Ces derniers sont récents mais montrent cependant des résultats encourageants. L'outil que nous avons adopté n'étant, lui, spécifique à aucun domaine, *Holmes Semantic Solutions* a donc créé un module qui ne regroupe que les informations et les opérations dont nous avons besoin pour notre domaine d'application. Nous pouvons donc présumer dans un avenir proche un développement important des recherches sur ce type de modèle de transformation en traitement automatique de la langue.

Le travail fournit durant ce stage a permis, hormis la transformation de dépendances, de tester et trouver des points à améliorer au niveau de l'outil, pour pouvoir l'adapter plus précisément à nos besoins.

Le module de règles développé ici renvoie de bon résultats, même s'il dépend majoritairement de la qualité de l'analyse en dépendances de surface effectuée en premier lieu. Cependant, sa plus grande limite est son temps d'exécution qui ne permet pas encore d'envisager, sous sa forme actuelle, d'application sur un grand nombre de textes.

2. Perspectives

Plusieurs perspectives pour l'amélioration de cette méthode de transformation peuvent être envisagées.

Pour optimiser le module et accroître la vitesse d'analyse, il serait intéressant d'étudier en profondeur le procédé développé pour Ogre [Ribeyre, 2012] au niveau des contraintes d'arcs, dans le but de l'intégrer au module et de traiter les cas de passages de structure de surface en structure profonde (par exemple les propositions, etc.).

D'autre part, un changement de fond pourrait être envisagé pour augmenter la vitesse d'analyse et réduire la chaine d'erreurs (erreurs du *parsing* en dépendances de surface + erreurs du module de transformation, dont certaines engendrées par les celles de l'étape précédente). Nous pourrions reconsidérer la méthode de conversion. En effet, le module de règles pourrait être appliqué directement sur le corpus du FTB, afin de constituer un nouveau corpus, analysé en dépendances de Stanford. Les erreurs générées par le module seraient diminuées car le *treebank* contient lui-même peu d'erreurs. De plus, les relations sous-spécifiées pourraient être traquées et transformées manuellement dans leur relation spécifique correspondante. Une fois le corpus constitué et corrigé, nous pourrions ensuite entraîner directement le *parser* dessus, permettant ainsi de réaliser l'analyse en dépendances profondes, sans conversion supplémentaire à effectuer (et donc sans erreurs supplémentaires engendrées).

Les avantages de cette solution en seraient alors un *parsing* simplifié, car la transformation se fait au niveau du corpus source une fois pour toutes, et une diminution d'erreurs car cette méthode permet une révision humaine des résultats.

Conclusion

1. Bilan

L'organisation du stage s'est répartie en deux missions principales dans deux domaines du traitement automatique de la langue : la parole et l'écrit.

En ce qui concerne la première mission, nous avons effectué la correction de transcriptions phonétiques, proposé des corrections automatiques lorsque cela était possible et établi une classification des erreurs récurrentes, en fonction du type de correction possible (automatique, manuel). Les frontières phonémiques ont également été ajustées au niveau de l'audio, par rapport aux anomalie-types détectées. Les données phonétiques et acoustiques ont donc été révisées et inclues dans le système de la synthèse vocale livrée au client en juillet 2013.

Pour notre seconde et principale mission, nous avons également atteint les objectifs fixés initialement. Nous avons établi une correspondance entre les spécificités du français et de l'anglais, et défini un modèle de dépendances pour le français sur la base de la représentation des *Stanford Dependencies*. Nous avons de plus développé le module de conversion des dépendances du système actuel vers la nouvelle version en dépendances profondes. Nous avons alors rempli l'objectif de la conversion de dépendances de surface vers une représentation plus profonde initialement créée pour l'anglais et adaptée au français.

La dualité du stage m'a permis d'entrer en contact avec deux tailles d'entreprises complètement différentes.

La grande structure de Nuance m'a permis de découvrir le fonctionnement d'une multinationale, à travers la gestion d'un projet mettant en connexion des équipes localisées sur plusieurs sites, et dans plusieurs pays. J'ai ainsi découvert l'importance de l'organisation hiérarchique entre et au sein des différents groupes.

Holmes Semantic Solutions (cinq employés, dont trois en télétravail) m'a permis de découvrir les avantages d'une petite structure, avec la possibilité d'échanger facilement et rapidement sur les projets, sur les erreurs, sur les décisions à prendre, etc.

Par ailleurs, j'ai pu découvrir durant ce stage le fonctionnement d'un projet ANR : la séparation des tâches entre les différents acteurs en fonction de leur spécialité, les réunions régulières et *workshops*, les comptes rendus, etc.

Chacun des deux projets réalisés pendant ce stage (avec Nuance et avec Ho2s), m'a permis d'évoluer au milieu d'outils/produits nouveaux (plus précisément un modèle de règles et un synthétiseur vocal). J'ai pu, dans les deux cas, participer à l'amélioration de ceux-ci, les tester et les évaluer. J'ai ainsi travaillé à la fois sur un produit pour un client et sur un projet de recherche.

J'ai également pu mettre à profit mes connaissances linguistiques, que ce soit sur la phonétique ou la syntaxe française, et les approfondir. D'autre part, j'ai beaucoup appris sur le domaine du traitement sémantique : les méthodes utilisées, les applications possibles, etc.

En outre, j'ai découvert le fonctionnement d'un environnement de développement (Eclipse[41]) et ainsi appris à travailler à plusieurs sur un projet commun.

2. Perspectives

Les 6 mois passés dans cette entreprise ont confirmé mon souhait de m'orienter vers le traitement automatique de l'écrit plus que de la parole, et plus particulièrement dans le domaine de la sémantique.

L'étude du sens des mots, phrases, etc. est une discipline qui permet une large variété de domaines d'action et de traitements (domaine médical, analyse d'opinions, etc.). Malgré le plein essor de la sémantique actuellement, beaucoup d'applications sont encore à envisager, et la recherche à ce niveau est loin d'être épuisée. Ce qui m'intéresse particulièrement dans ce domaine sont les types de traitements à mettre en œuvre, qui rassemblent dans des proportions plutôt équivalentes les deux disciplines principales du TAL : la linguistique et l'informatique.

En effet, les compétences principales mobilisées pour les deux projets durant ces 6 mois ont surtout été de l'ordre linguistique : analyse phonétique et étude approfondie de la syntaxe française. Mes connaissances dans ce domaine m'ont été très utiles.

[41] http://www.eclipse.org/

Pourtant, à l'avenir, je souhaiterais également mettre à profit et développer mes compétences informatiques, pour exercer réellement le métier de linguiste-informaticien tel que je le conçois : la réalisation de tâches nécessitant une combinaison (si possible équilibré) de connaissances en linguistique et en informatique.

Bibliographie

[Abeillé *et al.,* 2003] Abeillé A., Clément L., Toussenel F., 2003. Building a treebank for French. In Abeillé A. (Ed). *Treebanks :Building and Using Parsed Corpora* . Kluwer : Dordrecht . pp. 165-188

[Böhmová *et al.,* 2001] Böhmová A., Hajič B., Hajičová E., and Hladká B., 2001. The Prague Dependency Treebank: Three-Level Annotation Scenario. In Abeillé A. (Ed). *Treebanks: Building and Using Syntactically Annotated Corpora*, Kluwer Academic Publishers

[Buch-Kromann *et al*., 2009] Buch-Kromann M., Korzen I., Müller H. H., 2010. Uncovering the 'lost' structure of translations with parallel treebanks. In special issue of Copenhagen Studies of Language, vol. 38: Alves F., Göpferich S., and Mees I. (eds.). *Methodology, Technology and Innovation in Translation Process Research*. pp. 199-224

[Bonfante *et al.,* 2010] Bonfante G., Guillaume B., Morey M., Perrier G., 2010. Réécriture de graphes de dépendances pour l'interface syntaxe-sémantique. In *Proceedings of TALN'10*, Montréal, Canada

[Calia, 2002] Calia A., 2002. *La synthèse de la voix*. <http://www.iict.ch/Tcom/Presentations/Parole/Synthese.pdf> (dernière consultation : 08/08/13)

[Candito *et al.,* 2008] Candito M., Crabbé B., 2008. Expériences d'analyse syntaxique statistique du français, In *Proceedings of TALN'08*, Avignon, France

[Candito *et al.,* 2010] Candito M., Crabbé B., Denis P., 2010. Statistical French dependency parsing: treebank conversion and first results. In *Proceedings of LREC'2010*, La Valletta, Malta

[Candito *et al.,* 2011] Candito M., Crabbé B., Falco M., 2011. *Dépendances syntaxiques de surface pour le français - Schéma d'annotation pour un corpus en dépendances obtenu par conversion du FrenchTreebank* (dernière version)

[Cer *et al.,* 2010] Cer D., Jurafsky D., Manning C. D., Marneffe (de) M.-C., 2010. Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC'2010*, La Valletta, Malta

[Charniak, 1997] Charniak E., 1997. Statistical techniques for natural language parsing. AI Magazine, vol. 18 (n°4), pp.33-44

[Constant, 2012] Constant M., 2012. *Mettre les expressions multi-mots au cœur de l'analyse automatique de textes : sur l'exploitation de ressources symboliques externes.* Université Paris-Est, 104

[Copestake *et al.,* 2006] Copestake A, Flickinger D., Pollard C., Sag V., 2006, Minimal Recursion Semantics: An Introduction. In Springer Netherlands (Ed). *Research on Language and Computation,* vol. 3, pp.281–332

[Daille *et al.,* 2000] Daille B., Fourour N., Morin E., 2000, Catégorisation des noms propres : une étude en corpus. In Erss (Ed). Cahiers de Grammaire, vol. 25 , *Sémantique et Corpus* , pp. 115-129

[Dobrišek *et al.,* 1999] Dobrišek S., Mihelic F., Pavesik N., 1999, Acoustical modeling of phone transitions : biphones and diphones – What are the differences? In *Proceedings EUROSPEECH*, pp. 1307-1310.

[Falaise *et al.*, 2011] Falaise A., Kraif O., Tutin A., 2011, Une interface pour l'exploitation de corpus arborés par des non informaticiens : la plate-forme ScienQuest du projet Scientext. In TAL, vol 52 (n° 3). pp.241–246.

[Guillaume B. *et al.*, 2012] Guillaume B., Perrier G., 2012, Annotation sémantique du French Treebank à l'aide de la réécriture modulaire de graphes. In *Proceedings of TALN'12*, Grenoble, France

[Halls *et al.,* 2006] Halls J., Nilson J., Nivre J., 2006, MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of LREC 2006*, Genoa, Italy, pp. 2216-2219

[Johansson *et al.,* 2007] Johansson R., Nugues P., 2007, Extended Constituent-to-Dependency Conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, pp. 105–112

[Kahane, 2001] Kahane S., 2001, Grammaires de dépendance formelle et théorie Sens-Texte. In *Tutoriel*, *Actes TALN'2001*, vol. 2, Tours, 63p.

[Kalmbach, 2011] Kalmbach J-M, 2011, Phonétique et prononciation du français pour apprenants finnophone. Kielten laitos, Jyväskylän yliopisto (Eds)

[Karlov *et al.* 2012] Karlov B., Lacroix O., 2012, Prémices d'une analyse syntaxique par transition pour des structures de dépendance non-projectives. In *Proceedings of RECITAL 2012*, Grenoble, France

[Korte *et al*., 2010] Korte H., Pass G., Reichartz F., 2010, Semantic Relation Extraction With Kernels Over Typed Dependency Trees. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington, DC, USA, page 773-782.

[MacCartney *et al.,* 2006] MacCartney B., Manning C. D., Marneffe (de) M.-C., Generating Typed Dependency Parses from Phrase Structure Parses, In *Proceedings of LREC 2006*, pp.449-454

[Manning *et al.,* 2008] Manning C. D., Marneffe (de) M.-C., 2008, The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp.1-8

[Manning *et al*., 2012] Manning C. D., Marneffe (de) M.-C., 2012 (revised), Stanford typed dependencies manual, <http://nlp. stanford. edu/software/dependencies_manual. pdf>

[Mariani, 2009] Mariani J., 2009, *Language and speech processing(2nd version)*

[Mel'čuk, 1988] Mel' čuk I. A., 1988, *Dependency Syntax: Theory and Practice*. The SUNY Press (Ed), Albany, N.Y.. 428 pp.

[Nivre, 2005] Nivre J., 2005, *Dependency grammar and dependency parsing*. MSI report 05133. Växjö University: School of Mathematics and Systems Engineering.

[Pols *et al.,* 1996] Pols L.C.W., van Son R.J.J.H., 1996, An acoustic profile of consonant reduction. In P*roceedings of ICSLP 96*, vol.3, Philadelphia, PA, pp. 1529 - 1532

[Ribeyre, 2012] Ribeyre C., 2012, *Mise en place d'un système de réécriture de graphes appliqué à l'interface syntaxe-sémantique*. Université Paris Diderot 7, 73pp.

[Ribeyre, 2013] Ribeyre C., 2013, Vers un système générique de réécriture de graphes pour l'enrichissement de structures syntaxiques. In *Proceedings of TALN 2013*, Sables d'Olonne, France, pp.178-191

[Robinson, 1970] Robinson J.J., 1970 , dependency structure and transformational rules. In Linguistic Society of America (Eds), *Language,* vol. 46, n° 2, pp. 259-285

[Soli, 1981] Soli S.D., 1981, Second formants in fricatives: Acoustic consequences of fricative-vowel coarticulation. In Journal of the Acoustical Society of America, vol.70, n*4, pp. 976-984

[Taentzer, 1999] Taentzer G., 1999, AGG : A tool environment for algebric graph transformation. Applications of Graph Transformations with Industrial Relevance, vol. 1779, *Lecture Notes in Computer Science,* pp 481-488

[Tesnière, 1934] Tesnière Lucien, 1934, « Comment construire une syntaxe »

[Tesnière, 1959] Tesnière L., 1959, Introduction à la syntaxe structurale. In: Langue française. vol. 1, n°1. *La syntaxe*. pp. 36-40.

[Vicente, 2005] Vicente M., 2005, La glose comme outil de désambiguïsation référentielle des noms propres purs. In *CORELA - Le traitement lexicographique des noms propres | Numéros thématiques*. [En ligne] Publié en ligne le 02 décembre 2005. <http://corela.edel.univ-poitiers.fr/index.php?id=1212>. Consulté le 9/09/2013.

Annexe 1

Symboles de transcription utilisés

Alphabet Phonétique International (API) :[42]

VOYELLES		CONSONNES	
[i]	épi, il, lyre	[p]	père, soupe
[e]	aller, blé, chez (e fermé)	[t]	terre, vite
[ɛ]	merci, lait, fête (e ouvert)	[k]	cou, qui, sac, képi
[a]	ami, patte (a antérieur)	[b]	bon, robe
[ɑ]	pâte, pas (a postérieur)	[d]	dans, aide
[ɔ]	fort, donner, sol (o ouvert)	[g]	gare, bague, gui
[o]	mot, dôme (o fermé)	[f]	feu, neuf, photo
[u]	genou, roue	[s]	sale, celui, ça, dessous, tasse
[y]	rue, vêtu	[ʃ]	chat, tache, schéma
[ø]	peu, deux	[v]	vous, rêve
[œ]	peur, meuble	[z]	zéro, maison, rose

[42] Alphabet phonétique et valeur des signes, Le Petit Robert, http://pr.bvdep.com/aide/Pages/TableAPI.HTML

[ə]	premier (e caduc)	[ʒ]	gilet, je, geôle
[ɛ̃]	bain, brin, plein	[l]	lent, sol
[ɑ̃]	sans, vent	[ʁ][43]	rue, venir
[ɔ̃]	bonté, ton, ombre	[m]	mot, flamme
[œ̃]	lundi, brun, parfum	[n]	nous, tonne, animal
SEMI-CONSONNES		[ɲ]	agneau, vigne
[j]	paille, yeux, pied, panier	[ŋ]	camping (mots empruntés à l'anglais)
[w]	oui, fouet, joua (et joie)		
[ɥ]	huile, lui		

Autre symbole :

: Désigne la séparation entre deux tokens.

[43] Le Petit Robert note ce phonème /R/. En accord avec l'alphabet phonétique international, le symbole /ʁ/ sera utilisé dans ce papier pour le « r » français.

Annexe 2

Tagset des relations grammaticales testées par Candito et al. [2008] sur le *French Treebank*

TAG	CAT	SOUS-CAT	MODE
V	V	-	indicatif
VIMP	V	-	impératif
VINF	V	-	infinitif
VS	V	-	subjonctif
VPP	V	-	participe passé
VPR	V	-	participe présent
NPP	N	P	-
NC	N	C	-
CS	C	S	-
CC	C	C	-

TAG	CAT	SOUS-CAT	MODE
CLS	CL	suj	-
CLO	CL	obj	-
CLR	CL	refl	-
P	P	-	-
P+D		*voir texte*	
P+PRO		*voir texte*	
I	I	-	-
PONCT	PONCT	-	-
ET	ET	-	-

TAG	CAT	SOUS-CAT	MODE
ADJWH	A	int	-
ADJ	A	¬int	-
ADVWH	ADV	int	-
ADV	ADV	¬int	-
PROWH	PRO	int	-
PROREL	PRO	rel	-
PRO	PRO	¬(int | rel)	-
DETWH	D	int	-
DET	D	¬int	-

Annexe 3

Texte d'évaluation 1[44]

Les quelques 6.000 supporteurs afghans réunis au stade de la fédération étaient gonflés à bloc pour ce premier match international à domicile de leur équipe nationale depuis celui de 2003 contre le Turkménistan.

La sélection afghane, 139ème au classement de la FIFA, n'a jamais été inquiétée, prenant les devants dès la 21ème minute et ajoutant deux autres buts en seconde mi-temps.

Des policiers et des forces spéciales afghanes munis de boucliers étaient déployés dans et autour de ce petit stade situé au pied de montagnes escarpées.

Les aficionados afghans du ballon rond, dont des femmes rassemblées dans une section spéciale des tribunes, se sont époumonés dès l'hymne national et pendant une grande partie du match.

La FIFA souhaitait un duel amical entre ces deux pays voisins aux relations tendues qui ne s'étaient pas affrontés sur une pelouse de football à Kaboul depuis 1977, deux ans avant l'invasion soviétique.

L'Afghanistan accuse régulièrement son voisin du Pakistan d'aider les talibans à reprendre le pouvoir à Kaboul, des accusations rejetées par Islamabad qui accuse de son côté les autorités afghanes de laisser opérer sur leur territoire les talibans pakistanais.

La tenue pacifique de ce match est un accomplissement en soi pour l'Afghanistan, pays miné par plus de trois décennies consécutives de guerre et qui, malgré l'insécurité, a lancé l'an dernier son premier championnat national de football.

[44] source : http://www.7sur7.be/7s7/fr/1505/Monde/article/detail/1689882/2013/08/20/Afghanistan-Pakistan-duel-amical-sous-haute-securite.dhtml

Annexe 4

Texte d'évaluation 2[45]

Environ 200 personnes ont été évacuées en raison d'un incendie qui a brûlé 450 hectares de forêt et qui restait hors de contrôle mercredi matin sur l'île touristique de Majorque aux Baléares, ont indiqué les autorités de cette région espagnole.

Environ 200 personnes de 50 logements ont été évacuées. 450 hectares ont été brûlés à Arta et Capdepera.

C'est une zone d'agrotourisme, dans le nord-est de Majorque, a déclaré un porte-parole du gouvernement des Baléares.

Pour l'heure, le feu reste hors de contrôle, a-t-il ajouté, et environ 130 personnes ont été mobilisées pour combattre le feu attisé par une forte chaleur et du vent, selon les services de secours.

Les incendies de forêt et de broussailles sont très fréquents en été en Espagne, souvent favorisés par le vent sur des sols et une végétation très secs.

Jusqu'à présent, après un hiver pluvieux, ceux-ci ont été beaucoup moins nombreux cette année qu'en 2012, une année particulièrement dévastatrice.

13.335 hectares ont brûlé entre le 1er janvier et le 21 juillet, selon le ministère de l'Agriculture, contre 147.854 hectares durant la même période de 2012.

[45] source : http://www.7sur7.be/7s7/fr/1505/Monde/article/detail/1690183/2013/08/21/Incendie-sur-l-ile-de-Majorque-200-personnes-evacuees.dhtml

Annexe 5

Texte d'évaluation 3[46]

Deux alpinistes ont fait une chute mortelle samedi en milieu de journée au Weisshorn, l'un des plus hauts sommets des Alpes suisses.

Il s'agit d'un Valaisan de 23 ans et d'un Français de 37 ans, a indiqué dimanche la police valaisanne.

Le drame s'est produit lors de la descente alors qu'ils étaient encordés à une altitude de 4.200 mètres.

Ils ont chuté pour une raison indéterminée sur environ 300 mètres.

Les secours dépêchés sur place par un hélicoptère n'ont pu que constater le décès des deux alpinistes.

[46] source : http://www.7sur7.be/7s7/fr/1505/Monde/article/detail/1688543/2013/08/18/Chute-mortelle-de-deux-alpinistes-en-Suisse.dhtml

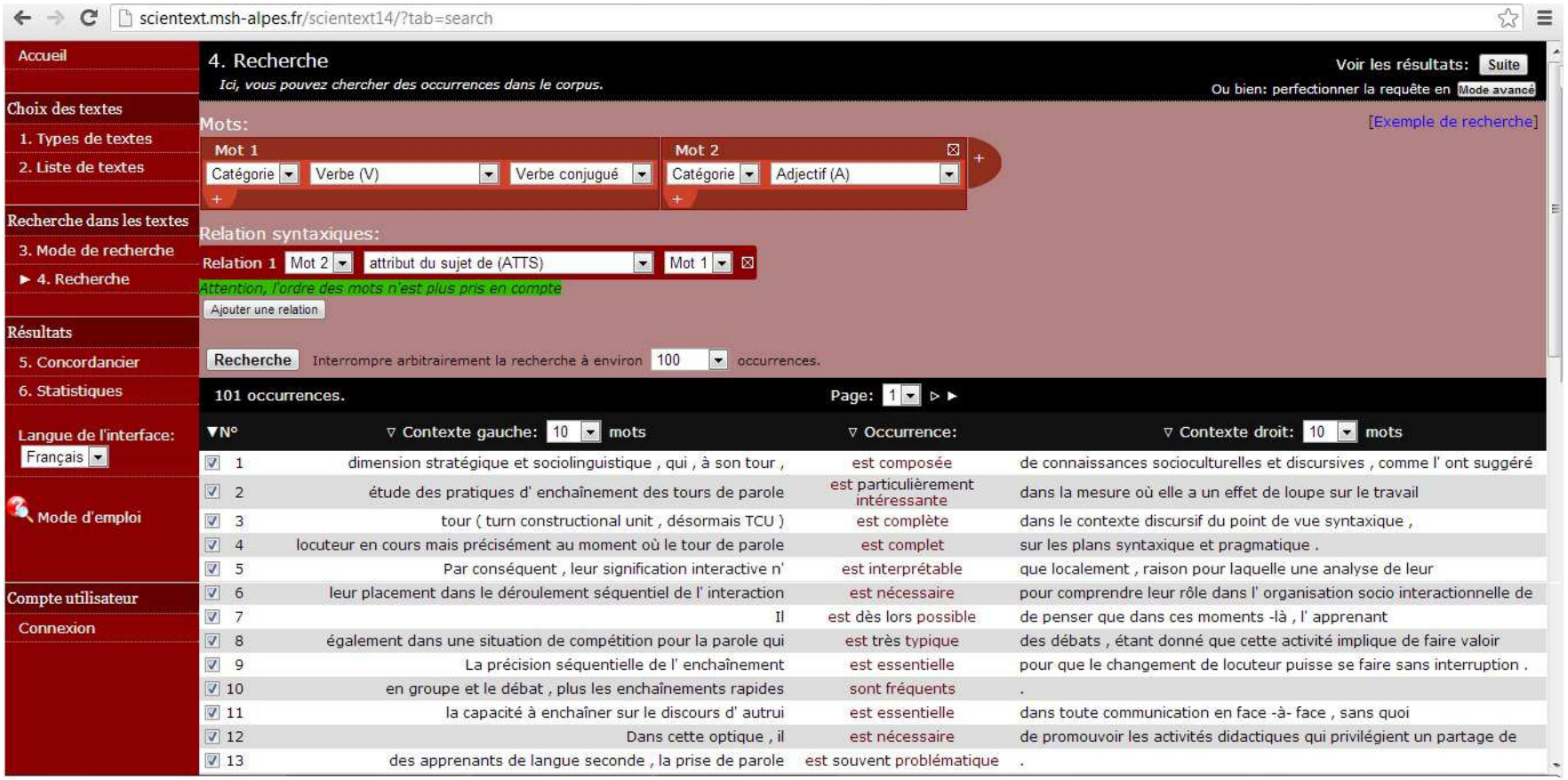
Annexe 6

Tableau de résultats

N° texte	Nb total de relations Holmes	Nb de relations transformées	Nb d'erreurs	Nb de relations sous-spécifiées	Précision	Rappel
1	257	257	4	2	98.4%	100%
2	212	212	2	2	99.1%	100%
3	103	103	3	0	97.1%	100%
Total	572	572	9	4	98.2%	100%

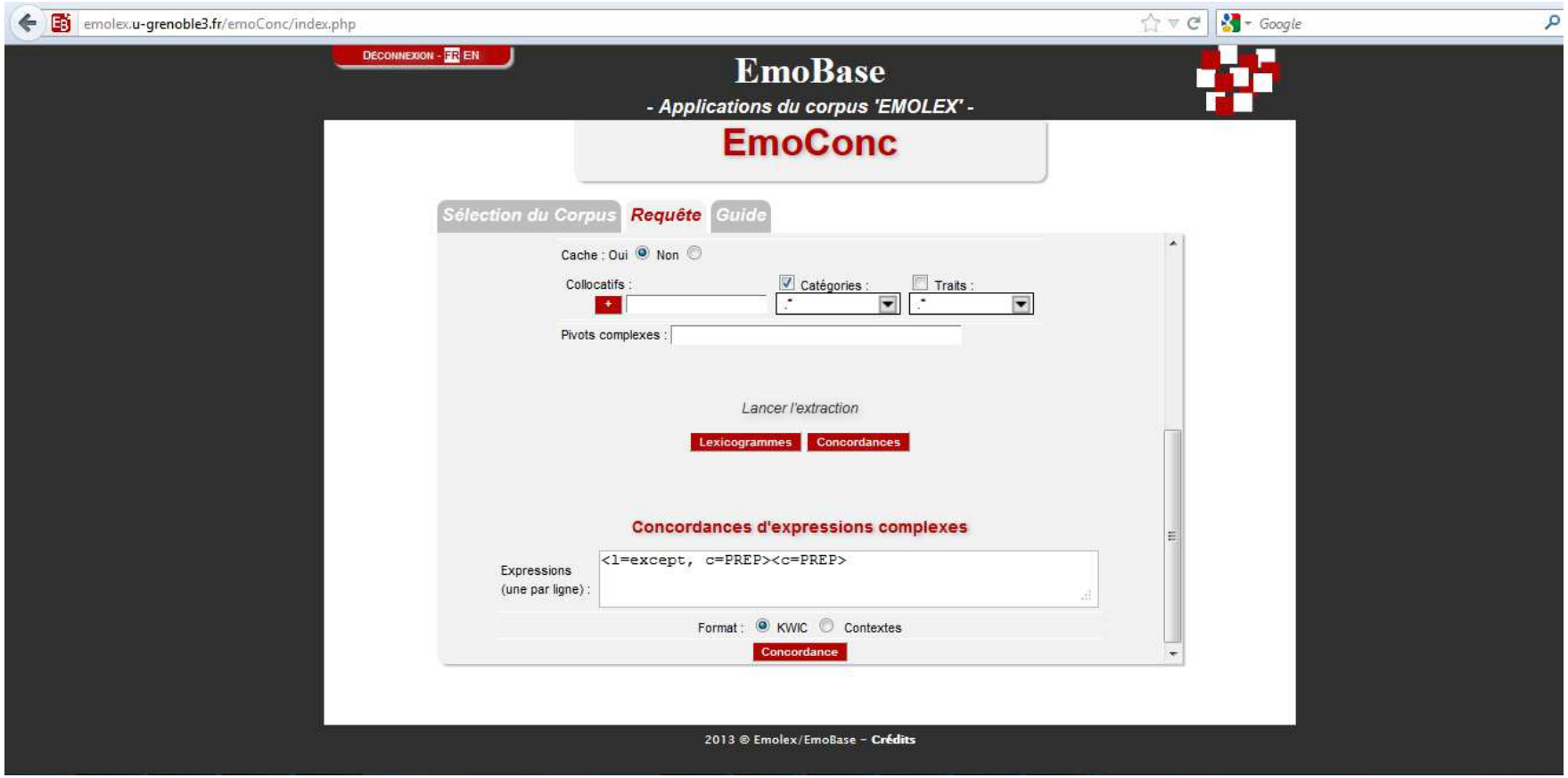
Annexe 7

Outil ScienQuest : résultat de la requête « *Adjectif* attribut du sujet de *Verbe* »



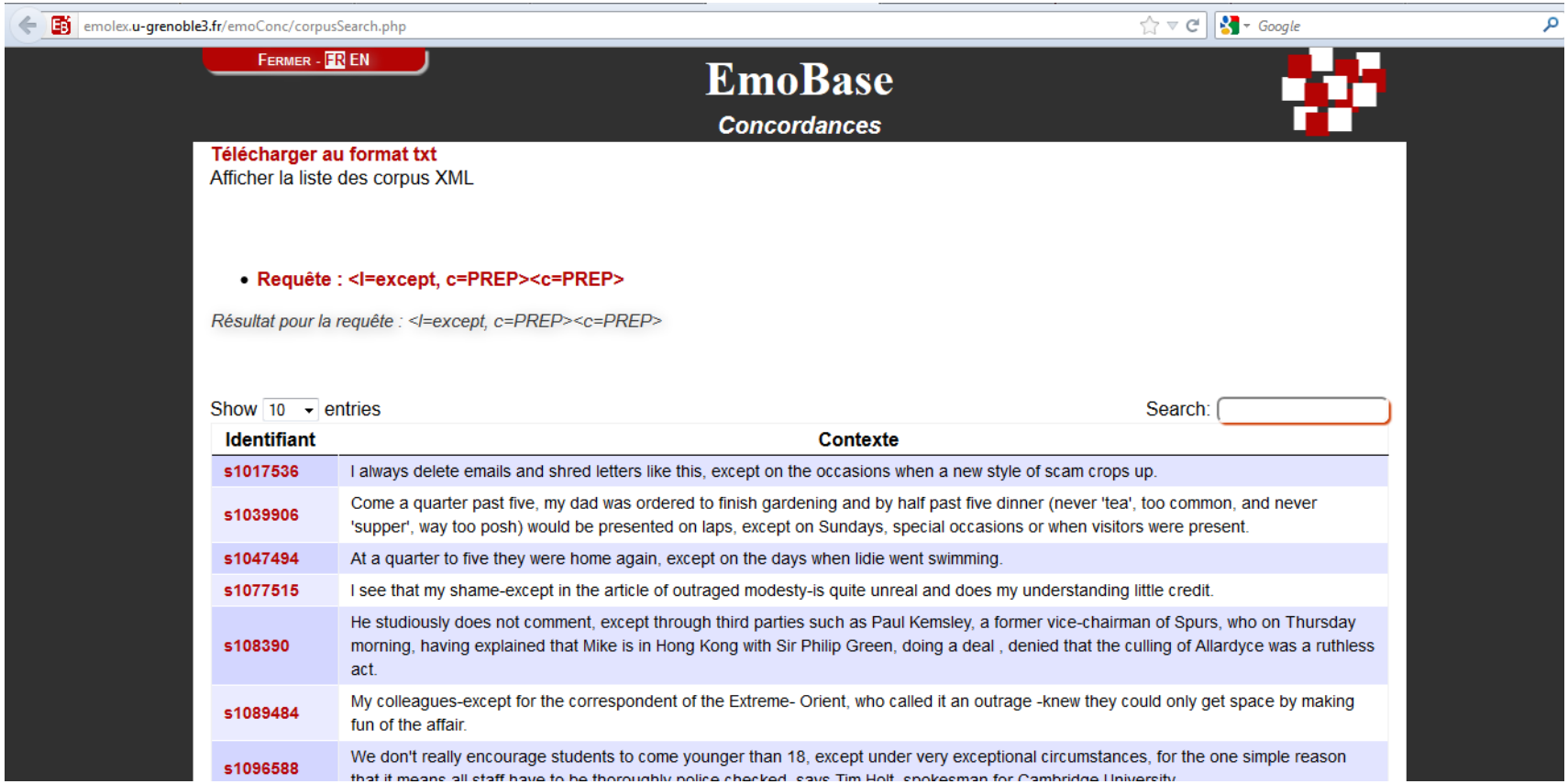
Annexe 8

Outil EmoConc : Formulaire de requête



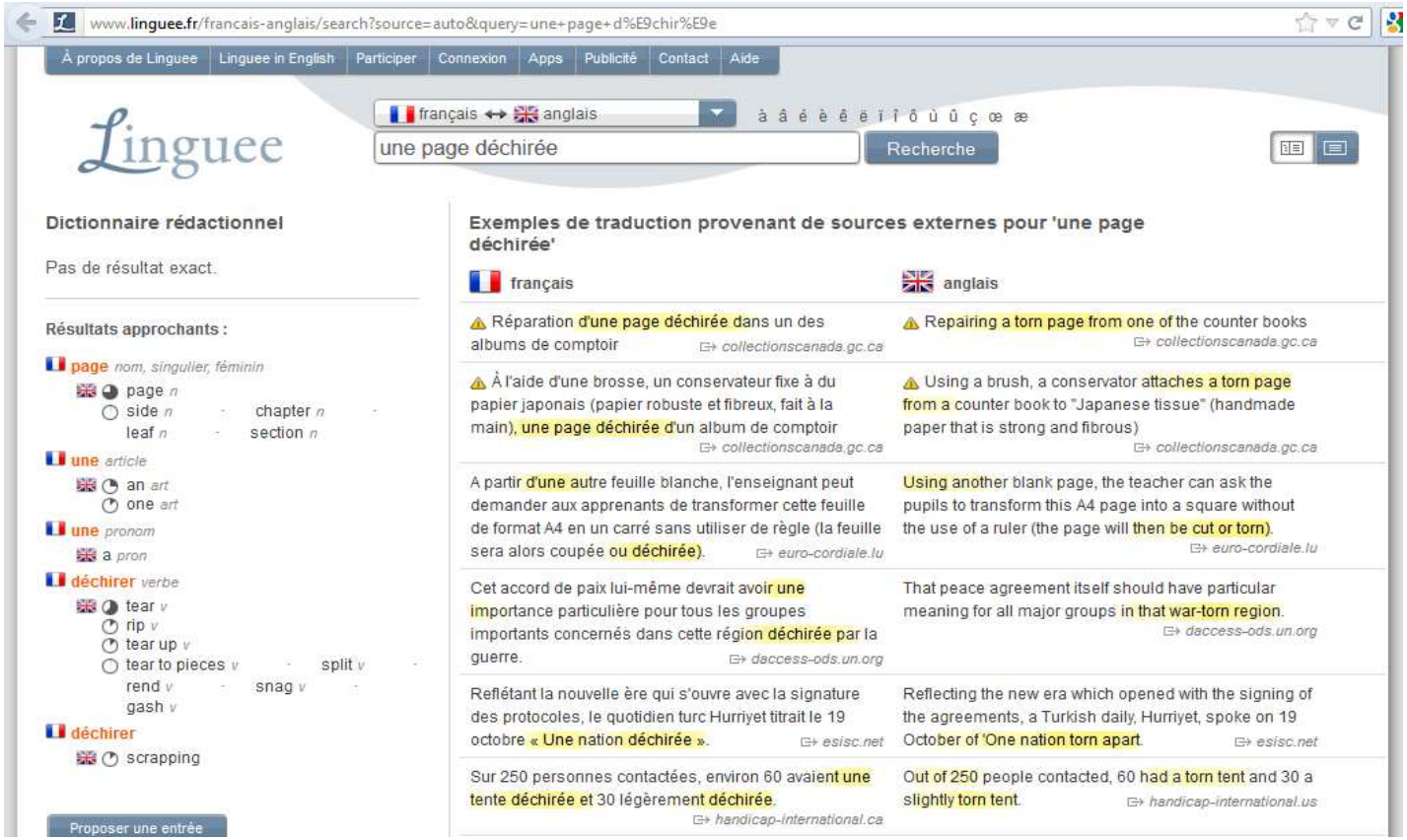
Annexe 9

Résultat de la recherche d'EmoConc pour la suite « except » + préposition



Annexe 10

Plateforme Linguee : Résultat de la recherche « une page déchirée »



Annexe 11

Extrait du module de règles

```
pos=PartOfSpeechAnnotation.string
vclass=VerbSenseAnnotation.string
lemma=LemmaAnnotation.string
vtrans=VTransOnlyAnnotation.string
vintrans=VIntransOnlyAnnotation.string
numType=NumberTypeAnnotation.string
name=name.int
p=p.int

%
f=fr.ho2s.brules.helpers.TestStaticFunctions.
%



/*
        Stanford relation ccomp
        Examples by type :
        [obj] "Il la laisse venir."
        [mod] "Il l'observe coudre."
        [ato] "Il laisse le chien venir."
*/
96, ccomp([1 pos=_v][2 pos=VINF])(1-2[type=_multi]) =>
([1][2])(1-2[type=ccomp])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(multi,"obj|mod|ato")}
).
test:"Il la laisse venir."


/*
        Stanford relation nsubj (in context of ccomp)
        Examples by type :
        [a_obj] "Il pourrait me laisser venir."
        [obj] "Il pourrait le laisser venir.", "Il laisse le chien venir.", "Il
pourrait laisser Marie venir."
        [aff] "Il s'imagine partir."
*/
98, nsubj_ccomp([1_me pos=_multipos][2_laisser pos=_v][3_venir pos=VINF])(2-
1[type=_multi] 2-3[type=ccomp]) =>
([1][2][3])(3-1[type=nsubj] 2-3[type=ccomp])
:=
({TestStaticFunctions.regxp(multipos,"CLO|NC|NPP")}
{TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(multi,"a_obj|obj|aff")}
).
test:"Il pourrait me laisser venir."
```

```
/*
        Stanford relation complm
        Cases of auxiliary or copula (change of head)
        Other example : "J'aimerais qu'il soit heureux."
*/
100, complm_auxcop([1_que lemma=_que pos=CS][2_pourrait pos=_v][3_prendre])(1-
2[type=obj] 3-2[type=_auxcop]) =>
([1][2][3])(3-1[type=complm] 3-2[type={auxcop}])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(que,"que|qu'")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il dit que tu pourrais prendre la voiture."


/*
        Stanford relation complm
        Other cases

*/
102, complm([1_que lemma=_que pos=CS][2_aimes pos=_v])(1-2[type=obj]) =>
([1][2])(2-1[type=complm])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(que,"que|qu'")}
).
test:"Il dit que tu aimes les cerises."


/*
        Stanford relation ccomp (structure V + "que" + V)
        Added dependency (compared to Holmes) + deletion of relation between (V,
"que")
        Cases of auxiliary or copula (change of head)
        Examples by type :
        [mod] "Il est content que tu manges des légumes."
        [obj] "Il est important que vous veniez."
*/
104, ccomp_que_auxcop([1_pourrait pos=_v][2_content][3_que lemma=_que
pos=CS][4_manges])(2-1[type=_auxcop] 1-3[type=_multi] 4-3[type=complm]) =>
([1][2][3][4])(2-1[type={auxcop}] 2-4[type=ccomp] 4-3[type=complm])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(que,"que|qu'")}
{TestStaticFunctions.regxp(multi,"mod|obj")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il pourrait être content que tu manges des légumes."


/*
        Stanford relation ccomp
        Added dependency (compared to Holmes) + deletion of relation between (V,
"que")
        Other cases
        Examples by type :
        [mod] "Il persuade Marie qu'il faut partir."
```

```
        [dep] "Paul reste convaincu que vous allez partir."
        [obj] "Il croit que tu manges des légumes."
        [ats] "Il semble que tu ne te sois pas réveillé."
*/
106, ccomp_que([1_persuade pos=_v][2_qu lemma=_que pos=CS][3_partir])(1-
2[type=_multi] 3-2[type=complm]) =>
([1][2][3])(1-3[type=ccomp] 3-2[type=complm])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(que,"que|qu'")}
{TestStaticFunctions.regxp(multi,"ats|mod|dep|obj")}
).
test:"Il persuade Marie qu'il faut partir."


/*
        Intermediary relation intermod
        Cases of "subordonnée complétive interrogative" or  clauses coordinated
with punctuation:
        Cases of auxiliary or copula (change of head) in the first clause
        Other examples : "Il est maigre, il doit avoir faim." "Je ne pourrais
pas savoir lequel a faim."
*/
108, intermod_auxcop([1_pourrait pos=_v][2_manger][3_pourrait pos=_vbis])(1-
3[type=mod] 2-1[type=_auxcop]) =>
([1][2][3])(2-3[type=intermod] 2-1[type={auxcop}])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(vbis,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il devrait manger, il a faim."


/*
        Intermediary relation intermod
        Cases of "subordonnée complétive interrogative" or  clauses coordinated
with punctuation:
        Other cases
        Other example : "Je ne sais pas lequel a faim."
*/
110, intermod([1_dort pos=_v][2_pourrait pos=_vbis])(1-2[type=mod]) =>
([1][2])(1-2[type=intermod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(vbis,"V|VPP|VS|VINF")}
).
test:"Il dort, il doit avoir faim."


/*
        Stanford relation ccomp (in context of intermod)
        Cases of "subordonnée complétive interrogative" or clauses coordinated
with punctuation
        Cases of auxiliary or copula (change of head) in the second clause
        Other example : "Je ne sais pas qui est le meilleur."
*/
112, ccomp_intermod_auxcop([1_sais][2_pourrait pos=_vbis][3_meilleur])(1-
2[type=intermod] 3-2[type=_auxcop]) =>
```

```
([1][2][3])(1-3[type=ccomp] 3-2[type={auxcop}])
:=
({TestStaticFunctions.regxp(vbis,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Je ne sais pas lequel pourrait être le meilleur."


/*
        Stanford relation ccomp (in context of intermod)
        Cases of "subordonnée complétive interrogative" or clauses coordinated
with punctuation:
        Other cases
*/
114, ccomp_intermod([1_sais][2_joue pos=_vbis])(1-2[type=intermod]) =>
([1][2])(1-2[type=ccomp])
:=
({TestStaticFunctions.regxp(vbis,"V|VPP|VS|VINF")}
).
test:"Je ne sais pas qui il imite."


/*
        Stanford relation ccomp
        Other cases of "subordonnée complétive interrogative" (Holmes relation
obj)
        No need for special copula and auxiliary cases
*/
116, ccomp2([1_sais pos=_v][2_est pos=_vbis])(1-2[type=obj]) =>
([1][2])(1-2[type=ccomp])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(vbis,"V|VPP|VS|VINF")}
).
test:"Je ne sais pas qui il est."


/*
        Stanford relation acomp
        Hanldes V + adj cases when V isn't a copula, and that there is no clitic
(already handled in the xcomp_ADJ_VPP rules)
        Other examples :
        [mod] "Il tombe malade.", "Il vole bas.", "Il mange chaud."
        [ato] "Il se croit malin."
*/
118, acomp([1 pos=_v][2 pos=ADJ])(1-2[type=_modAto]) =>
([1][2])(1-2[type=acomp])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(modAto,"mod|ato")}
).
test:"Il tombe malade."


/*
        Stanford relation prep (in context of "pour que" + V)
        Cases of auxiliary or copula (change of head)
        Other example : "Il vient pour qu'elle puisse l'aider."
*/
```

```
120, prep_pour_auxcop([1_pourque lemma=_PourQue][2_soit
pos=_v][3_tranquille])(1-2[type=obj] 3-2[type=_auxcop]) =>
([1][2][3])(3-1[type=prep] 3-2[type={auxcop}])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(PourQue,"pour qu[e']")}
).
test:"Il vient pour qu'elle soit tranquille."


/*
        Stanford relation prep (in context of "pour que" + V or "pour" + Vinf)
        Other cases
        Other example : "Il vient pour qu'elle l'aide."
*/
122, prep_pour([1_pourque lemma=_PourQue][2_dormir pos=_v])(1-2[type=obj]) =>
([1][2])(2-1[type=prep])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(PourQue,"pour( qu[e'])?")}
).
test:"Il vient pour dormir."


/*
        Stanford relation purpcl (in context of : V + "pour"( + Vinf) et V +
"pour que"( + V))
        Cases of auxiliary or copula (change of head)
        Other example : "Il pourrait être gentil au moins pour aider son
frère.", "Il aurait pu être gentil pour qu'elle l'aide.", "Il pourrait venir
pour qu'elle soit heureuse."
*/
124, purpcl_auxcop([1_est pos=_v][2_gentil][3 lemma=_PourQue][4])(2-
1[type=_auxcop] 1-3[type=mod] 4-3[type=prep]) =>
([1][2][3][4])(2-1[type={auxcop}] 2-4[type=purpcl] 4-3[type=prep])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(PourQue,"pour( qu[e'])?")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il est gentil uniquement pour aider son frère."


/*
        Stanford relation purpcl (in context of : V + "pour"( + Vinf) et V +
"pour que"( + V))
        Other cases
        Other examples : "Il fait ça pour qu'elle soit tranquille.", "Il est
venu pour qu'elle puisse l'aider."
*/
126, purpcl([1 pos=_v][2 lemma=_PourQue][3])(1-2[type=mod] 3-2[type=prep]) =>
([1][2][3])(1-3[type=purpcl] 3-2[type=prep])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(PourQue,"pour( qu[e'])?")}
).
test:"Il fait ça pour aider son frère."
```

```
/*
        Stanford relation amod
        Handles NC + ADJ cases
        Other examples : "Il mange une pomme verte.", "Je vois Marie anxieuse."
*/
128, amod([1 pos=_NcNpp][2 pos=ADJ numType=_notCard])(1-2[type=mod]) =>
([1][2])(1-2[type=amod])
:=
({TestStaticFunctions.regxp(NcNpp,"NC|NPP")}
{!TestStaticFunctions.regxp(notCard,"CARD")}
).
test:"Le livre a une page déchirée."


/*
        Stanford relation infmod
        Handles NC + Vinf cases
        Cases of auxiliary (change of head)
*/
130, infmod_aux([1 pos=NC][2 pos=P lemma=_DeA][3 pos=VINF][4 pos=VINF])(1-
2[type=dep] 2-3[type=obj] 4-3[type=aux]) =>
([1][2][3][4])(1-4[type=infmod] 4-2[type=prep] 4-3[type=aux])
:=
({TestStaticFunctions.regxp(DeA,"de|à")}
).
test:"Il a l'impression de devoir partir."


/*
        Stanford relation infmod
        Handles NC + Vinf cases
        Other cases
        Other examples : "C'est l'occasion de mettre le plan en route."
*/
132, infmod([1 pos=NC][2 pos=P lemma=_DeA][3 pos=VINF])(1-2[type=dep] 2-
3[type=obj]) =>
([1][2][3])(1-3[type=infmod] 3-2[type=prep])
:=
({TestStaticFunctions.regxp(DeA,"de|à")}
).
test:"Il a sa voiture à réparer."


/*
        Stanford relation partmod (VPR or VPP)
        Cases of auxiliary or copula (change of head)
        Warning : doesn't work for VPP at the moment : clash with ccomp3 (rule
116) : "Séparés, les enfants étaient tristes" vs "Il est triste, il a perdu."
        Other examples : "S'attendant à de meilleurs résultats, les médecins
étaient tristes."
        */
134, partmod_auxcop([1 pos=_v][2][3 pos=VPR])(2-1[type=_auxcop] 1-3[type=mod])
=>
([1][2][3])(2-1[type={auxcop}] 2-3[type=partmod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
```

```
test:"S'attendant à de meilleurs résultats, les médecins pourraient arrêter les
tests."


/*
        Stanford relation partmod (VPR or VPP)
        Other cases
        Other example : "S'attendant à de meilleurs résultats, les médecins
étaient déçus."
*/
136, partmod([1][2 pos=_VppVpr])(1-2[type=mod]) =>
([1][2])(1-2[type=partmod])
:=
({TestStaticFunctions.regxp(VppVpr,"VPP|VPR")}
).
test:"Les cerises cueillies au mois de juin sont les meilleures."


/*
        Stanford relation nn
        Handles noun + noun (NC and NPP) cases
        Other examples : "Le service client est ouvert.", "Elle s'appelle Marie
Bernard.", "Elle y a pensé vers la fin mai."
*/
138, nn([1 pos=_NcNpp][2 pos=_NcNppbis])(1-2[type=mod]) =>
([1][2])(1-2[type=nn])
:=
({TestStaticFunctions.regxp(NcNpp,"NC|NPP")}
{TestStaticFunctions.regxp(NcNppbis,"NC|NPP")}
).
test:"Dans la ville dortoir, rien ne se passe."

/*
        Stanford relation poss
        Handles possessive determiners cases
        Other example : "Je lui ai donné mon PC."
*/
140, poss([1 pos=DET lemma=_poss][2 pos=_NcNpp])(2-1[type=det]) =>
([1][2])(2-1[type=poss])
:=
({TestStaticFunctions.regxp(poss,"mon|ton|son|ma|ta|sa|mes|tes|ses|notre|votre|
leur|nos|vos|leurs")}
{TestStaticFunctions.regxp(NcNpp,"NC|NPP")}
).
test:"Il a mon CD."


/*
        Stanford relation neg
        Handles "ne" next to "pas" or "plus"
        Other example : "Il pourrait ne pas dormir." "Il pourrait ne plus être
malade."
*/
142, neg_double_aux([1 pos=_v][2 pos=ADV lemma=_ne][3 pos=ADV
lemma=_PasPlus][4])(1-3[type=mod] 3-2[type=mod] 4-1[type=aux]) =>
([1][2][3][4])(4-2[type=neg] 4-3[type=neg] 4-1[type=aux])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(ne,"ne|n'")}
```

```
{TestStaticFunctions.regxp(PasPlus,"plus|pas")}
).
test:"Il pourrait ne plus être malade."


/*
	Stanford relation neg
	Handles negative "ne", "non" or "pas" (different analysis for "pas" in
Holmes compared to "plus")
	Cases of auxiliary or copula (change of head)
	Other example : "Il n'est pas rapide.", "Il est belge et non pas
français."
*/
144, neg_auxcop([1 pos=_v][2][3 pos=ADV lemma=_NeNonPas])(1-3[type=mod] 2-
1[type=_auxcop]) =>
([1][2][3])(2-3[type=neg] 2-1[type={auxcop}])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(NeNonPas,"ne|n'|non|pas")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"Il ne pourrait pas manger."


/*
	Stanford relation neg
	Handles negative "ne", "non" or "pas" ("pas" not handled like "rien",
"personne" etc)
	Other cases
	Other example : "Il part non pas de Paris mais de Lyon."
*/
146, neg([1 pos=_v][2 pos=ADV lemma=_NeNonPas])(1-2[type=mod]) =>
([1][2])(1-2[type=neg])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(NeNonPas,"ne|n'|non|pas")}
).
test:"Il ne mange pas."


/*
	Stanford relation neg
	Hanldes other case of negative adverb : "que"
	No handling of "plus" (ambiguity) Ex : "Je n'en ai pas plus que toi.",
"Je n'en veux pas non plus.", "Je n'en veux plus."
*/
148, neg_otheradv([1 pos=ADV][2][3 pos=ADV lemma=_qu])(2-1[type=neg] 2-
3[type=mod]) =>
([1][2][3])(2-1[type=neg] 2-3[type=neg])
:=
({TestStaticFunctions.regxp(qu,"qu'|que")}
).
test:"Cela ne fait qu'un an."


/*
	Stanford relation neg
	Handles cases of negative pronouns : "personne", "rien", "aucun" (except
cases like "je ne connais aucune boulangerie ici" ->det)
```

```
	Other examples : "Il n'y a personne.", "Il peut n'y avoir personne." "Il
pourrait ne rien y avoir." "Je n'en connais aucun."
*/
150, neg_otherpro([1 pos=ADV][2][3 pos=PRO lemma=_multi])(2-1[type=neg] 2-
3[type=obj]) =>
([1][2][3])(2-1[type=neg] 2-3[type=neg])
:=
({TestStaticFunctions.regxp(multi,"personne|rien|aucun")}
).
test:"Il n'y a rien."


/*
	Stanford relation advmod
	Cases of auxiliary or copula (change of head)
	Other examples : "C'est seulement une histoire.", "L'endroit, où je suis
heureux, est ici."
*/
152, advmod_auxcop([1 pos=_v][2][3 pos=_AdvProrel])(1-3[type=mod] 2-
1[type=_auxcop]) =>
([1][2][3])(2-3[type=advmod] 2-1[type={auxcop}])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(AdvProrel,"ADV|ADVWH|PROREL")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"L'endroit, où je pourrais aller, est loin."


/*
	Stanford relation advmod
	Other cases
	Other examples : "L'endroit, où je mange, est cher.", "Il roule vite.",
"Cela ne concerne vous pas tous, mais seulement lui.", "Je ne sais pas quand
faire ça."
*/
154, advmod([1][2 pos=_AdvProrel])(1-2[type=mod]) =>
([1][2])(1-2[type=advmod])
:=
({TestStaticFunctions.regxp(AdvProrel,"ADV|ADVWH|PROREL")}
).
test:"Il mange salement."


/*
	Stanford relation tmod
	Cases of auxiliary or copula (change of head)
	"soirée" not included in the list (ambiguity of meaning between "end of
the day" and "party")
	Other example : "La semaine dernière, le temps aurait pu être beau."
*/
156, tmod_auxcop([1_semaine lemma=_temp][2_pu pos=_v][3_faire])(3-
2[type=_auxcop] 2-1[type=mod]) =>
([1][2][3])(3-2[type={auxcop}] 3-1[type=tmod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(temp,"lundi|mardi|mercredi|jeudi|vendredi|samedi|dim
anche|an|année|mois|semaine|jour|journée|matin|matinée|soir|janvier|février|mar
```

```
s|avril|mai|juin|juillet|août|septembre|octobre|novembre|décembre|aujourd'hui|h
ier|avant-hier|demain|après-demain|été|printemps|hiver|automne")}
{TestStaticFunctions.regxp(auxcop,"aux|cop")}
).
test:"La semaine dernière, il aurait pu faire beau."


/*
        Stanford relation tmod
        Other cases
        Examples by type :
        [mod] "La semaine dernière, il faisait des crêpes."
        [obj] "Il a passé/aurait pu passer la journée à la plage."
*/
158, tmod([1_journée lemma=_temp][2_passé pos=_v])(2-1[type=_objMod]) =>
([1][2])(2-1[type=tmod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF")}
{TestStaticFunctions.regxp(objMod,"obj|mod")}
{TestStaticFunctions.regxp(temp,"lundi|mardi|mercredi|jeudi|vendredi|samedi|dim
anche|an|année|mois|semaine|jour|journée|matin|matinée|soir|janvier|février|mar
s|avril|mai|juin|juillet|août|septembre|octobre|novembre|décembre|aujourd'hui|h
ier|avant-hier|demain|après-demain|été|printemps|hiver|automne")}

).
test:"Il a passé la journée à la plage."


/*
        Handling of Holmes arg relation : de...à...
        Case 1
        Examples by type:
        [de_obj] "Le taux est passé de 5 à 7%"
        [mod] "De 300 à 400 personnes étaient invitées."
*/
160, arg_Holmes1 ([1_invite][2_de pos=P][3_nb][4_a
pos=P][5_nb][6_personnes])(1-2[type=_multi] 2-4[type=arg] 4-6[type=pobj] 2-
3[type=det] 6-5[type=det]) =>
([1][2][3][4][5][6])(1-2[type=prep] 2-3[type=pobj] 1-4[type=prep] 4-
6[type=pobj] 6-5[type=det])
:=
({TestStaticFunctions.regxp(multi,"de_obj|mod")}
).
test:"Il a invité de 300 à 400 personnes."


/*
        Handling of Holmes arg relation : de...à...
        Case 2
        Other examples: "Il y a vécu de 2003 à 2007." "Il va de Paris à Cannes."
*/
162, arg_Holmes2 ([1][2][3][4][5])(1-2[type=prep] 2-3[type=pobj] 2-4[type=arg]
4-5[type=pobj]) =>
([1][2][3][4][5])(1-2[type=prep] 2-3[type=pobj] 1-4[type=prep] 4-5[type=pobj]).
test:"Il y va de juin à juillet."


/*
        Relation compar
```

```
        Comparision handling : new relation
        Other examples : "Sa situation est meilleure que la mienne.", "Elle a
les pieds plus grands que les tiens.", "Elle va aussi vite que moi."
*/
164, compar([1 pos=_AdvAdj][2 lemma=que pos=CS])(1-2[type=dep]) =>
([1][2])(1-2[type=compar])
:=
({TestStaticFunctions.regxp(AdvAdj,"ADV|ADJ")}
).
test:"Il court moins vite que toi."


/*
        Relation refl
        Reflexive pronouns handling : new relation
        Other examples : "Il aurait pu se rappeler de toi.", "Je m'adresse à
toi.", "Les chemises se lavent à 40°."
*/
166, refl([1 pos=CLR][2 pos=_v])(2-1[type=aff]) =>
([1][2])(2-1[type=refl])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
).
test:"Il se base sur des faits."


/*
        Stanford relation num
*/
168, num([1 pos=_detAdj numType=CARD p=_position][2 pos=NC numType=_notCard
p=_positionbis])(2-1[type=_detmod]) =>
([1][2])(2-1[type=num])
:=
({TestStaticFunctions.regxp(detAdj,"DET|ADJ")}
{(position==positionbis-1)}
{!TestStaticFunctions.regxp(notCard,"CARD")}
{TestStaticFunctions.regxp(detmod,"det|mod")}
).
test:"J'ai trois euros dans ma poche."


/*
        Stanford relation number
*/
170, number([1 pos=DET numType=CARD p=_position][2 pos=DET numType=CARD
p=_positionbis][3 pos=NC])(3-1[type=det] 3-2[type=num]) =>
([1][2][3])(2-1[type=number] 3-2[type=num])
:=
({(position==positionbis-1)}
).
test:"J'ai trois cent euros dans ma poche."


/*
        Stanford relation number
*/
172, number2([1 numType=CARD p=_position][2 numType=CARD p=_positionbis][3
numType=CARD][4 pos=NC])(4-1[type=det] 3-2[type=number] 4-3[type=num]) =>
([1][2][3][4])(2-1[type=number] 3-2[type=number] 4-3[type=num])
```

```
:=
({(position==positionbis-1)}
).
test:"J'ai trois cent cinquante euros dans ma poche."


/*
	Stanford relation number
*/
174, number3([1 numType=CARD p=_position][2 numType=CARD p=_positionbis][3
numType=CARD][4 numType=CARD][5 pos=NC])(5-1[type=det] 3-2[type=number] 4-
3[type=number] 5-4[type=num]) =>
([1][2][3][4][5])(2-1[type=number] 3-2[type=number] 4-3[type=number] 5-
4[type=num])
:=
({(position==positionbis-1)}
).
test:"J'ai trois cent cinquante euros dans ma poche."


/*
	Stanford relation number
*/
176, number4([1 numType=CARD p=_position][2 numType=CARD p=_positionbis])(2-
1[type=det]) =>
([1][2])(2-1[type=number])
:=
({(position==positionbis-1)}
).
test:"Ca leur a coûté 300 millions d'euros."


/*
	Stanford relation advmod (for "y")
	Cases of copula (change of head)
	Handling of Holmes aff relation
	Other example : "Il doit y être heureux."
*/
178, advmod_cop_y([1 pos=CLO lemma=y][2 pos=_v][3])(2-1[type=p_obj] 3-
2[type=cop]) =>
([1][2][3])(3-1[type=advmod] 3-2[type=cop])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
).
test:"Il y est heureux."


/*
	Stanford relation pobj (for "en")
	Cases of copula (change of head)
	Handling of Holmes aff relation
	Other example : "Il pourrait en être content."
*/
180, pobj_cop_en([1 pos=CLO lemma=en][2 pos=_v][3])(2-1[type=de_obj] 3-
2[type=cop]) =>
([1][2][3])(3-1[type=pobj] 3-2[type=cop])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
).
```

```
test:"Il en est fier."


/*
        Stanford relation advmod (for "y")
        Only cases of transitive verbs
        Handling of Holmes aff relation
*/
182, advmod_y([1 pos=CLO lemma=y][2 pos=_v vtrans=TRUE])(2-1[type=aff]) =>
([1][2])(2-1[type=advmod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
).
test:"Il y vis sa vie."


/*
        Stanford relation advmod (for "y" and "en")
        Only cases of intransitive verbs
        Handling of Holmes aff relation
        Other example : "J'en viens."
*/
184, advmod_y_en([1 pos=CLO lemma=_YEn][2 pos=_v vintrans=TRUE])(2-1[type=aff])
=>
([1][2])(2-1[type=advmod])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
{TestStaticFunctions.regxp(YEn,"y|en")}
).
test:"Il y dors."


/*
        Stanford relation dobj
        Cases of "direct" transitive verbs only
        Handling of Holmes aff relation (rule including "en")
*/
186, dobj_trans([1 pos=CLO][2 pos=_v vtrans=TRUE])(2-1[type=aff]) =>
([1][2])(2-1[type=dobj])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
).
test:"Il m'aide."


/*
        Stanford relation iobj
        Handles all non only-"direct"-transitive verbs
        Handles cases of direct and indirect transitive verbes (at the same time
and in the same context) ex:Je te donne une chemise
        However over-generation of the rule in case of a direct transitive verb
in a context and indirect in another context (ex : Je te/lui rappelle de
prendre un ballon. vs Je te/le rappelle. (téléphone))
        (includes also "y" and "en cases)
        Examples by type :
        [aff] "Il me donne sa chaise."
        [a_obj] "Il lui donne une feuille."
        [de_obj] "Il s'en souvient.", "Il en est déçu."
*/
```

```
188, iobj ([1 pos=CLO][2 pos=_v])(2-1[type=_multi]) =>
([1][2])(2-1[type=iobj])
:=
({TestStaticFunctions.regxp(v,"V|VPP|VS|VINF|VPR")}
{TestStaticFunctions.regxp(multi,"aff|a_obj|de_obj")}
).
test:"Il me donne sa chaise."


/*
        Stanford relation dobj
        Cases of auxiliary in relative clause (change of head)

*/
190, dobj_aux_rel([1_qu][2_faut][3_acheter])(2-1[type=obj] 3-2[type=aux]) =>
([1][2][3])(3-1[type=dobj] 3-2[type=aux]).
test:"C'est la voiture qu'il faut acheter."


/*
        Stanford relation dobj
        General (specific cases handled in previous layers)
*/
192, dobj([1][2])(2-1[type=obj]) =>
([1][2])(2-1[type=dobj]).
test:"Il mange une poire."


/*
        Deletion of Holmes relation ponct
*/
194,ponct([1][2 pos=PONCT])(1-2[type=ponct])=>
([1])().
test:"Voilà."


/*
        Deletion of any relation containing a punctuation
        Handles Holmes errors
*/
196,ponct2([1][2 pos=PONCT])(1-2[type=_rel])=>
([1])()
:=
({TestStaticFunctions.regxp(rel,".*")}
).
test:"Bob (qui le sait) commanda un poulet."


/*
        Stanford relation dep
        Change of any Holmes relation into general relation "dep"
        Exceptions : "mod" -> also Stanford upper-level category, "det"->also
Stanford category, "dep" : no need to transform to the same thing
*/
198,dep([1][2])(1-2[type=_Hrel])=>
([1][2])(1-2[type=dep])
:=
({TestStaticFunctions.regxp(Hrel,"suj|obj|de_obj|a_obj|p_obj|ats|ato|aux_tps|au
x_caus|aux_pass|aff|mod_rel|coord|arg|dep_coord|ponct")}
```

```
).
test:"Tu choisis qui tu veux."

/*
        Stanford relation root
        Moves the root to the head of the tree
        Other examples : "Il est content.", "Il devrait pouvoir manger."
        Otherwise, no changes for the root relation
*/
200, root([1_root p=_val][2_mais][3_content])(3-2[type=_rel] 1-2[type=root])=>
([1][2][3])(3-2[type={rel}] 1-3[type=root])
:=
({val==-2}
{TestStaticFunctions.regxp(rel,".*")}
).
       test:"Mais il est content."
```

Table des illustrations

Sigles et abréviations utilisés

DG : *Dependency Grammar*

DT : *Dependency Tree*

FTB : French TreeBank

Ho2s *: Holmes Semantic Solutions*

Holmes : *Hybrid Operable platform for Language Management and Extensible Semantic*

PTB : *Penn TreeBank*

PST : *Phrase Structure Tree*

PSG : *Phrase Structure Grammar*

SD : Stanford Dependencies

Glossaire

Assimilation : « Action par laquelle deux phonèmes, du fait qu'ils sont contigus à brève distance, tendent à devenir identiques ou à acquérir des caractères communs » (Dictionnaire de l'Académie Française : http://atilf.atilf.fr/academie9.htm)

Diphone : Paires de phonèmes contigus, c'est-à-dire des unités de parole qui ne sont pas considérées comme des entités distinctes, mais comme deux unités reliées par une transition [Dobrišek *et al*., 1999].

Entité nommée : « Entité Nommée est la notion utilisée en TAL pour désigner les éléments discursifs monoréférentiels qui coïncident en partie avec les noms propres (note : […]concerne les noms propres mais aussi les dates et les mesures) et qui suivent des patrons syntaxiques déterminés » [Vicente, 2005]

Homographe : « (Mot) dont la graphie est identique à celle d'un autre mot.» (TLFi)

Homophone : « [En parlant d'unités ou de groupements graphiques (signe, syllabe, mot, phrase)] De prononciation identique. » (TLFi)

Liaison : « [En français] Procédé consistant à prononcer devant la voyelle ou le h muet initial d'un mot, la consonne finale, ordinairement muette, du mot précédent.» (TLFi)

Projectif : « Une structure est projective si pour chaque mot w tous les mots qui se trouvent entre w et ses subordonnés sont aussi dominés par w » [Karlov et al., 2012]

Token : Une unité lexicale composée d'un ou plusieurs mots (exemple : « parce que »)

Typologie syntaxique :

La typologie syntaxique d'une langue est liée à l'ordre de certains mots dans sa syntaxe : le sujet, le verbe et l'objet. Certaines langues sont donc dites de typologie SVO, SOV, etc. tandis que certaines n'ont pas d'ordre fixe.

Référent : « Ce à quoi le signe linguistique renvoie soit dans la réalité extra-linguistique ou univers réel, soit dans un univers imaginaire » (TLFi)