Université de Strasbourg

Faculté des Langues

Mémoire de Master 2
Linguistique Informatique Traduction, option Informatique

ODACR : un Outil de Détection Automatique des Chaînes de Référence à base de règles linguistiques

rédigé par Bruno Oberlé
sous la direction de Mme Todirascu

soutenu le 12 juin 2017

devant un jury composé de :

Mme Catherine Schnedecker	LiLPa, Université de Strasbourg
Mme Amalia Todirascu	LiLPa, Université de Strasbourg

Sommaire

Remerciements

Je tiens à remercier Madame Schnedecker, qui m'a suggéré de m'inscrire dans le master Linguistique Informatique Traduction, et sans qui ce mémoire n'aurait donc pas vu le jour. Je la remercie également pour ses précieux conseils, sa disponibilité constante et son aide irremplaçable.

Je tiens à remercier Madame Todirascu, directrice de recherche de ce mémoire, qui m'a guidé dans mon travail et m'a aidé à trouver des solutions pour avancer. Je la remercie également d'avoir tout fait pour que je puisse soutenir avant l'audition pour l'attribution des contrats doctoraux de recherche de l'ED 520.

Enfin, j'adresse mes plus sincères remerciements à ma mère, pour son soutien financier et affectif sans faille.

Introduction

0.1 Les chaînes de références

Les chaînes de référence sont définies comme « la suite des expressions d'un texte entre lesquelles l'interprétation construit une relation d'identité référentielle » (Corblin 1985a). Les *expressions référentielles* qui les composent sont appelées, par métaphore, des *maillons*. Une chaîne est constituée d'au moins trois maillons (Schnedecker 1997), sinon la notion n'est guère pertinente (celle d'*anaphore* suffit).

Illustrons cette définition par le texte suivant, extrait des fables d'Ésope, où l'on voit deux chaînes : celle dont le référent est « le laboureur », et celle dont le référent est « l'aigle »[1] :

(1) [Un laboureur]$_i$, ayant trouvé [un aigle pris au filet]$_j$, fut si frappé de [sa]$_j$ beauté qu'[il]$_i$ [le]$_j$ délivra et [lui]$_j$ donna la liberté. [L'aigle]$_j$ ne [se]$_j$ montra pas ingrat envers [[son]$_j$ bienfaiteur]$_i$...

Introduites dans une perspective philosophique par Chastain (1975) sous le nom de « chaînes anaphoriques », et étudiées en linguistique française par Corblin (1985a, 1987, 1995), Charolles (1988) et Schnedecker (1997), les chaînes de référence sont un domaine de recherche actif, tant en linguistique (un numéro de la revue *Langages* leurs a été consacré en 2014) qu'en traitement automatique des langues (dorénavant TAL) (un projet ANR, Democrat[2], leur est consacré).

Les chaînes de référence jouent en effet un rôle essentiel dans ce qui fait qu'un ensemble de phrases est perçu comme *un texte*. Tout texte s'organise autour de quelques entités

1. Les indices i et j après les crochets désignent l'indice de l'expression référentielle ; quand deux expressions ont le même indice, elles renvoient au même référent.

2. « DEscription et MOdélisation des Chaînes de Référence : outils pour l'Annotation de corpus (en diachronie et en langues comparées) et le Traitement automatique ».

qui réapparaissent à plusieurs reprises sous formes d'expressions linguistiques diverses, qui « font système » Schnedecker (1997) et, en formant des *chaînes*, créent *l'unité* du texte. De plus, la coréférence participe de la *cohésion textuelle* (Halliday et Hasan 1976) et, par un jeu subtil d'interactions, de la *structuration des textes* : Charolles (1988) insiste sur le rôle *organisateur* des chaînes de référence, qu'il considère comme un des plans de l'organisation textuelle.

La détection automatique des chaînes est un enjeu majeur de la recherche, notamment en raison de leurs applications concrètes, par exemple pour l'indexation de documents (les moteurs de recherche), la traduction automatique ou le résumé automatique.

0.2 Les systèmes de détection automatique

0.2.1 La résolution automatique de la coréférence

La détection automatique est toutefois une tâche complexe. Il faut d'abord trouver les expressions référentielles et les délimiter, puis relier les expressions qui sont *coréférentielles*, c'est-à-dire qui ont le même référent. Cela peut sembler une tâche triviale pour un humain, car nous avons une compétence linguistique et une connaissance du monde qui nous permet de faire ces liens de façon naturelle. Mais une machine est dépourvue de cette compétence et de cette connaissance, qu'il faut alors les suppléer par des algorithmes et des ressources auxiliaires.

Plusieurs algorithmes ont été proposés : certains s'attachent à appliquer des règles plus ou moins inspirées de théories linguistiques quand d'autres font usage de larges corpus pour en déduire des modèles statistiques (voir le chapitre 2). Mais ces algorithmes sont limités : les règles linguistiques ne peuvent gérer qu'un ensemble limité de cas, même quand elles sont très nombreuses, et l'apprentissage automatique dépend beaucoup du corpus d'entraînement, qui doit être à la fois disponible, de taille suffisante, et annoté correctement.

La détection automatique est d'abord difficile du « point de vue formel » : « le problème est non seulement que l'identité ne s'établit pas en vertu de la répétition, mais en outre que l'identité peut reposer sur une série très étendue de formes différentes [...], de la position vide aux groupes nominaux pleins en passant par une gamme diversifiée de formes dites pronominales » (Corblin 1995). Mais c'est surtout au niveau sémantique que la tâche devient la plus complexe, puisqu'aucune règle ni aucun apprentissage ne peut permettre de détecter les relations sémantiques (par exemple *chat* et *animal* ou

entre *Rhin* et *fleuve*) sans l'ajout de ressources auxiliaires (ontologies, dictionnaires, etc.).

0.2.2 Les systèmes français

En français, seuls deux systèmes existent. L'un est à base d'apprentissage : il s'agit de CROC[3] (Muzerelle, Lefeuvre, Antoine, Schang, Maurel, Villaneau et Eshkol 2011) ; mais il a été entraîné sur le corpus ANCOR (Désoyer, Landragin, Tellier, Lefeuvre et Antoine 2015), qui est un corpus de français oral (c'est cependant le seul corpus français de taille importante annoté en coréférence). Cela signifie qu'il peut difficilement résoudre la coréférence pour les textes écrits, or la plupart des applications travaillent sur l'écrit, et non sur l'oral. De plus, CROC ne fait que résoudre la coréférence : il ne détecte pas les expressions référentielles. Cela signifie qu'il ne peut pas trouver les chaînes de référence dans un texte tout-venant : il lui faut non seulement un texte dans lequel les expressions référentielles ont déjà été *repérées*, mais aussi *annotées* avec des propriétés syntaxiques et sémantiques ; certaines sont même propres à l'oral, comme l'indication du tour de parole.

L'autre est à base de règles : il s'agit de RefGen, élaboré par Longo (2013) au cours de sa thèse. C'est un système à base de règles linguistiques qui prend appui, notamment, sur la théorie de l'accessibilité d'Ariel (1990). Nous décrirons le fonctionnement de ce système plus en détail au chapitre 2, nous insisterons ici surtout sur ses limites, qui nous incitent à en proposer une alternative.

RefGen est composé de trois modules :

TTL Un étiqueteur morpho-syntaxique (qui assigne à chaque token une catégorie grammaticale) et un *chunker* qui regroupe les tokens en groupes simples[4] (nominaux, adjectivaux, verbaux et prépositionnels),

RefAnnot Un module qui regroupe les syntagmes nominaux en groupes nominaux complexes (des syntagmes nominaux), et qui marquent les entités nommées et les pronoms *il* impersonnels.

CalcRef Un module qui crée les chaînes de référence.

Plusieurs problèmes limitent cependant l'usage de RefGen. D'abord, le module TTL est un programme auxiliaire, créé par Ion (2007). Il a été entraîné par Longo pour

3. CROC pour *Coreference Resolution for Oral Corpus.*
4. Ce ne sont pas des syntagmes.

le français, mais le code du programme lui-même n'est pas disponible : il faut l'utiliser *via* un service web (c'est-à-dire uniquement en étant connecté à Internet), avec une contrainte de taille (on ne peut lui envoyer que de petits morceaux de texte).

Le module CalcRef pose aussi un certain nombre de problèmes dans la résolution des chaînes de référence : par exemple, il ne détecte pas les anaphores infidèles (*un chat... cet animal*). De plus, ses règles sont trop simples ; par exemple, les expansions ne sont pas prises en compte dans le calcul de la référence, de sorte que *le chat de ma tante* et *le chat de ma voisine* peuvent être dans la même chaîne simplement parce que le lemme *chat* est le même dans les deux syntagmes.

Selon les calculs de Longo (2013), RefGen a une performance[5] d'environ 55 %. Non seulement il fait des erreurs, mais en plus il détecte peu de maillons. Dans le texte suivant, par exemple, sont en police normale les maillons détectés par RefGen, en gras ceux qu'il aurait dû détecter mais ne l'a pas fait.

(2) [Abdelmalek Benbara]$_i$, retrouvé mort mercredi dans le coffre de [son]$_i$ véhicule, a reçu plusieurs coups de pied ou de poing au visage et trois coups de couteau qui **[lui]$_i$** ont été portés alors [il]$_i$ agonisait. L'un de ces coups **[lui]$_i$** a brisé le larynx, provoquant [son]$_i$ asphyxie. Les deux autres coups ont touché le cœur et le thorax. Toujours selon les constatations du légiste, la mort de **[M. Benbara]$_i$** est intervenue rapidement. La date de **[sa]$_i$** mort serait contemporaine à celle de **[sa]$_i$** disparition. En effet, **[sa]$_i$** barbe n'a pas eu le temps d'apparaître.

Par ailleurs, Longo (2013) a élaboré son système dans le cadre d'une convention CIFRE, en partenariat avec une entreprise[6], aujourd'hui disparue. Le cœur du programme a été développé par un ingénieur de l'entreprise, et ne peut donc pas être librement distribué : cela le rend indisponible, et donc inexploitable.

5. Voir le chapitre 7 pour les différents calculs de la performance.
6. *Ready Business System*.

0.3 Problématique et objectif

Notre objectif est de proposer un outil de détection automatique des chaînes de référence, à base de règles linguistiques[7], *end-to-end*[8], qui soit une alternative à RefGen, à la fois meilleure (en termes de performance) et plus disponible (sans problème de droits d'auteur).

Nous essayerons de pallier les défauts de RefGen sur plusieurs aspects. Sur l'usabilité du programme, d'abord. Nous reprogrammerons l'ensemble du système, afin de pouvoir le redistribuer dans la communauté scientifique. Ensuite, nous chercherons un étiqueteur morpho-syntaxique facilement intégrable et surtout *libre de droits* (contrairement à TTL, qui est ni intégrable, ni libre de droits).

Sur la syntaxe, ensuite. Nous nous servirons d'une analyse syntaxique complète, car la détection de la coréférence peut bénéficier des propriétés syntaxiques des expressions référentielles.

Sur la sémantique, également. Aucun des deux systèmes français ne fait usage de ressources sémantiques pour résoudre les anaphores infidèles, or de telles ressources sont indispensables pour détecter correctement les chaînes de référence. Mais aucune ressource adéquate n'existe pour le français ; nous suivrons deux voies pour en élaborer : l'une concernera les entités nommées, à la fois pour la détection et pour leur reprise anaphorique ou coréférentielle, l'autre les hyperonymes.

Sur l'algorithme, enfin. Nous rechercherons un algorithme plus performant que celui proposé actuellement par RefGen, trop simple et n'exploitant que des indices de surface.

0.4 Enjeux

La recherche sur les systèmes à base de règle est importante pour l'introduction de ressources lexicales, notamment des dictionnaires d'entités nommées ou d'hyperonymes (voir chapitre 4). L'apprentissage automatique seul ne peut pas, en effet, apprendre des listes d'entités nommées avec leur caractéristiques (comme le sexe de la personne) ou les liens d'hyperonymie, car, même pour des corpus très grands, il n'y a pas assez données

7. Un système par apprentissage requiert un corpus de grande taille, qui n'existe pas pour le français écrit.

8. C'est-à-dire que le système accepte du texte brut tout-venant en entrée : aucune annotation préalable n'est requise.

pour avoir plusieurs fois toutes les entités nommées possibles ou toutes les relations lexicales possibles. Il faut donc utiliser des ressources lexicales spécialisées, et le moyen les plus rapide de les intégrer est d'utiliser un ensemble de règles[9].

Par ailleurs, plusieurs recherches montrent que les systèmes *hybrides*, qui associent règles linguistiques, ressources lexicales et apprentissage automatique, permettent d'atteindre de meilleures performances que les systèmes qui n'utilisent que l'un ou l'autre de ces méthodes (Longo 2013). C'est pourquoi l'étude des systèmes à base de règles est importante et permet de contribuer à l'élaboration de ces systèmes hybrides.

Enfin, on peut supposer que les systèmes à base de règles et les systèmes à base d'apprentissage sont complémentaires, dans le sens où les uns sont bons là où les autres sont mauvais, et inversement. Par exemple, on peut supposer que les systèmes à base de règles repèrent facilement les pronoms impersonnels (qui ne peuvent pas entrer dans une chaîne de référence car dépourvus de référence) grâce à des patrons combinés à des listes de verbes (par exemple, des verbes de météo) et d'adjectifs (par exemple, *il* est impersonnel s'il est suivi d'un verbe de météo comme *pleuvoir*, avec éventuellement un adverbe de négation entre les deux : *il pleut, il ne pleut pas*). Par contre, l'algorithme d'apprentissage aura sans doute plus de mal à reconnaître les pronoms impersonnels de ce type (notamment parce qu'il risque de manquer d'exemples dans le corpus d'entraînement). Les systèmes à base de règles peuvent donc venir complémenter les systèmes par apprentissage, ou même les remplacer selon le résultat que l'on souhaite : les systèmes à base de règles, notamment, pourraient être meilleurs en précision[10] et les systèmes à base d'apprentissage meilleurs en rappel[11].

0.5 Organisation du travail

Ce travail présentera d'abord les questions que pose la modélisation linguistique des chaînes de référence, et les choix d'annotation que nous avons faits (chapitre 1). Puis

9. L'intégration de ressources lexicales dans un système par apprentissage est possible, mais complexe. Voir Constant, Tellier, Duchier, Y. Dupont, Sigogne et Billot (2011) et Uryupina (2007) pour quelques exemples.

10. Précision et rappel sont deux façons de mesurer la performance d'un outil. Imaginons que nous voulions seulement des documents qui parlent de *chat* au sens d'animal, et non de *chat* au sens de conversation par messages électroniques (un *chat* sur Internet). La précision mesure la pertinence des résultats : on ne veut *que* des documents qui parlent des félins, quitte à en oublier quelques-uns : l'important est de ne pas avoir l'anglicisme. Le rappel, c'est l'inverse : on veut *tous* les documents qui parlent des félins, quitte à en avoir quelques-uns qui parlent aussi de conversation électronique.

11. C'est ce qui a été constaté pour la détection automatique des entités nommées.

nous examinerons les différents types de systèmes de détection automatique (chapitre 2). Nous présenterons ensuite l'architecture générale de notre programme (chapitre 3), avant de décrire plus précisément ses besoins en ressources lexicales (chapitre 4) et en analyse syntaxique (chapitre 5). Nous exposerons ensuite l'algorithme dans le détail (chapitre 6). Enfin, nous évaluerons le programme et le situerons par rapport à d'autres programmes de détection automatique des chaînes de référence (chapitre 7).

0.6 Le projet Democrat

Nous terminerons cette introduction en évoquant le projet Democrat (*DEscription et MOdélisation des Chaînes de Référence : outils pour l'Annotation de corpus (en diachronie et en langues comparées) et le Traitement automatique*), un projet financé par l'Agence Nationale de la Recherche (no. ANR-15-CE38-0008) pour quatre ans depuis janvier 2016. Il est coordonné par Frédéric Landragin et associe quatre laboratoires : le LaTTICe à Paris (dont le responsable pour le projet est Frédéric Landragin), LiLPa à Strasbourg (Catherine Schnedecker), et les laboratoires ICAR et IHRIM à Lyon (Céline Guillot-Barbance (ENS Lyon)).

Le projet « vise à développer les recherches sur la langue et la structuration textuelle du français via l'analyse détaillée et contrastive des chaînes de référence [...] dans un corpus diachronique de textes écrits entre le 9ème et le 21ème siècle, avec des genres textuels variés »[12].

Concernant le TAL, le projet vise à « optimiser CROC » et à « ouvrir la voie à la réalisation de systèmes hybrides, conciliant plusieurs techniques d'apprentissage ainsi que des systèmes de règles comme celui utilisé dans l'outil RefGen développé au LiLPa ».

Notre étude s'inscrit donc dans la perspective de Democrat d'explorer des techniques de systèmes à base de règles afin d'ouvrir la voie à des systèmes hybrides. Nous espérons que ce travail pourra contribuer à cette réflexion au sein de Democrat.

12. Extrait du résumé du projet sur le site de l'ANR : http ://www.agence-nationale-recherche.fr/projet-anr/ ?solr=run&tx_lwmsuivibilan_pi2%5BCODE%5D=ANR-15-CE38-0008

Chapitre 1

Chaînes de référence et choix d'annotation

1.1 Modélisation des chaînes de référence

L'étude linguistique des chaînes de référence a surtout été initiée, pour le français, par Schnedecker (1997). Beaucoup de questions, synthétisées par Schnedecker et Landragin (2014), restent cependant encore en discussion.

Qu'est-ce que la « référence » ? Les chaînes de référence ont surtout été étudiées lorsque leur référent est humain. Certains auteurs décident ainsi de n'annoter que les référents humains (Glikman, Guillot-Barbance et Obry 2014 ; Landragin 2011), ou, lorsqu'ils incluent les chaînes à référent non humain, il s'agit généralement de référents concrets ou bien définis (par exemple, l'objet du litige opposant deux parties dans les 'year books' étudiés par Capin (2014)). Seules Longo et Todirascu (2014) s'aventurent à annoter des référents abstraits ou prédicatifs, comme la réduction des gaz à effet de serre.

Qu'est-ce qu'un maillon ? Pour Charolles (1988, p. 8), « seules peuvent appartenir (donner lieu à) une chaîne des expressions employées référentiellement, c'est-à-dire toutes et rien que les expressions nominales (ou pronominales) permettant d'identifier un individu (un objet de discours) quelle que soit sa forme d'existence (personne humaine, événement, entité abstraite) ». On peut pourtant se demander s'il ne faudrait

pas aussi annoter des prédicats, qu'ils soient des noms ou des verbes, comme le proposent Longo et Todirascu (2014).

Il faut aussi examiner le cas des « éléments non-référentiels qui participent... à la co-référence » (Landragin 2011, p. 63), comme les sujets zéro ou les marques d'accord. Landragin (2011), par exemple, propose d'annoter certains de ces éléments (qu'il nomme « maillons faibles »), mais pas tous.

Quelles sont les limites des chaînes ? On peut s'interroger sur la persistance d'une chaîne sur tout un texte : un personnage d'un roman initie-t-il une chaîne qui s'étend sur *tout* le roman ? On peut trouver des critères textuels pour couper les chaînes, par exemple lorsqu'on trouve un inter-titre, ou bien lorsqu'on commence un nouveau chapitre, voire un nouveau paragraphe (Schnedecker et Landragin 2014, p. 5-8). On peut également envisager des critères linguistiques, propres aux chaînes, pour effectuer le découpage, ce que Schnedecker (1997, p. 24) appelle des « bornes internes aux chaînes ». Par exemple, quand une chaîne se compose essentiellement de pronoms personnels, et de quelques expressions nominales, comme dans :

(3) SN1... Pro... Pro... Pro... (SN2)... Pro... Pro... Pro... SN1'... Pro... Pro...

on peut dire que cette

> redénomination est décisive car, en présentant le référent comme il l'était en première mention, elle indique qu'il y a *re-saisie*, *re-démarrage référentiels*. Partant, on dispose de moyen de fractionner de manière très cohérente, les chaînages jusque-là considérés comme des suites non délimitées de maillons (Schnedecker 1997, p. 32).

Quelle relation établir entre les maillons ? Toutes les relations entre les maillons d'une même chaîne ne peuvent pas être mises sur le même plan. La différence la plus évidente est celle entre anaphore et co-référence ; Corblin (1995, p. 167-169) en fait deux types de chaînes : les chaînes-A (pour anaphoriques) et les chaînes-R (pour référentielles). L'anaphore concerne « des termes dont l'interprétation n'est pas fixée pour tout emploi » (p. 32), comme le pronom de troisième personne *il* dans :

(4) a. Pierre est venu. *Il* repartira demain. (nous soulignons)

L'anaphore entraîne des « liens anaphoriques », pour lesquels « la connexion à une expression antérieure est déclenchée et régie par le contenu linguistique de la forme »

(p. 168). À l'inverse, dans la coréférence, l'interprétation « est indépendante de son contexte immédiat d'usage » (p. 32), comme le nom propre dans :

(5) b. Pierre est venu. *Pierre* repartira demain. (nous soulignons)

La coréférence entraîne des « liens référentiels », « obtenus sur la base d'un savoir concernant la communication : connaissance directe de l'univers de référence, mémoire d'usage antérieur, hypothèses concernant les intentions du locuteur, etc. » (p. 168).

On peut aussi opposer les liens entre les maillons en regardant le type d'anaphore (Schnedecker et Landragin 2014) : anaphores fidèles (*un homme... cet homme*), anaphores infidèles hyperonymiques (*un boeuf... l'animal*), recatégorisantes (*Paul... cette andouille*), pronominales (*Paul... il*).

Comment caractériser les chaînes ? Schnedecker (1997, 2005) propose un certain nombre de paramètres des chaînes, résumés par Schnedecker et Landragin (2014), par exemple :

- leur longueur (*i.e.* le nombre de leurs maillons),
- leur portée (est-elle locales (les chaînes dépassent-elles par exemple le pagaraphe) ou est-elle bien globales (les chaînes couvrent-elles tout le texte, ou du moins une bonne partie ?)) (voir Schnedecker (2014)),
- la distance intermaillonnaire moyenne (il y a différentes façons de calculer cette distance : caractères/mots entre les maillons/têtes de maillon),
- la catégorie et la fonction des maillons, notamment du premier maillon de la chaîne,
- le patron, c'est-à-dire la ou les séquences les plus courantes des catégories grammaticales des maillons (voir notamment Schnedecker et Longo (2012)),
- le mode de cohabitation : succession, entrecroisement, dérivation, partition, fusion, déroulement parallèle.

Le genre a-t-il une influence sur le comportement des chaînes ? De nombreuses études montrent que le genre « le genre textuel conditionne les modalités de l'expression référentielle et, plus précisément, la composition des chaînes de référence » (Schnedecker (2014) ; voir aussi Longo et Todirascu (2009) et Schnedecker (2015) pour un aperçu général sur le genre et les chaînes de référence, Schnedecker (2005) pour le portrait journalistique, Schnedecker et Longo (2012) pour les faits divers, Schnedecker (2014) pour les recettes de cuisine et les introductions, etc.).

Il est important de pouvoir répondre à ces questions avant d'établir un système de détection des chaînes, car elles révèlent les aspects sur lesquels le système doit faire des choix, par exemple : quels sont les segments linguistiques que le système doit considérer comme référentiels ? La chaîne doit-elle recommencer à chaque nouveau nom propre ? Concernant cette dernière question, notamment, les deux principaux systèmes pour le français s'opposent : alors que RefGen (Longo 2013) recommence une chaîne à chaque nom propre, CROC (Désoyer, Landragin, Tellier, Lefeuvre et Antoine 2015) continue la chaîne quelle que soit la catégorie grammaticale des expressions référentielles (y compris, donc, les noms propres).

1.2 Comparaison entre différents choix d'annotation

Notre projet s'inscrit à la fois dans la continuité de notre master de Sciences du Langage (Oberlé 2016) et dans celle du projet Democrat[1]. Nous avons consacré un chapitre (chapitre 4) de notre mémoire du premier aux choix d'annotations (pour annoter les textes du corpus de notre recherche), en les justifiant d'un point de vue linguistique ; nous y renvoyons le lecteur. Le second s'est doté d'un *manuel d'annotation*, qui répond à la plupart de ces questions (pour annoter les textes du corpus du projet). Les deux documents, cependant, ne sont pas toujours compatibles. Dans le cadre de ce projet-ci, nous avons préféré utiliser les choix que nous avons fait en Sciences du Langage, linguistiquement justifiés, aux choix de Democrat. C'est le cas, par exemple, de la délimitation des expressions référentielles : nous annotons le syntagme complet, là où Democrat demande de n'inclure ni les propositions relatives, ni les participes, ni les appositions nominales. Ce choix nous semble manquer de pertinence scientifique (en fait, la raison de ce choix est *technique* et non linguistique), puisque qu'un adjectif sera inclus dans l'expression référentielle, mais pas une relative à valeur adjectivale. De plus, on perd beaucoup d'informations sur les propriétés linguistiques de l'expression référentielle (longueur, nombre et type d'expansions) en coupant ainsi le syntagme. Enfin, d'un point de vue technique, la plupart des analyseurs syntaxiques automatiques indiquent les dépendances syntaxiques en reliant toutes les dépendances sous une même tête syntaxique, et donc, *de facto*, prennent le syntagme comme unité.

Notre projet s'inscrit également dans la continuité de RefGen, dont les choix d'annotations sont basés sur des principes linguistiques, plutôt que techniques. Longo choisit

1. Que nous avons présenté p. 13.

par exemple de redémarrer une nouvelle chaîne à chaque nouveau nom propre, en suivant Schnedecker (1997). Néanmoins, nous ne sommes pas toujours d'accord avec les choix de RefGen[2] : de façon générale, nous avons préféré nos propres choix d'annotation, justifiés au cours de notre master de Sciences du Langage[3]. Pour les noms propres, par exemple, il nous paraît peu opportun de recommencer une nouvelle chaîne pour d'Artagnan à chaque fois qu'il est nommé par son nom, comme dans l'exemple suivant[4] :

(6) Tout le long de la route, le duc se fit mettre au courant par *d'Artagnan* non pas de tout ce qui s'était passé, mais de ce que *d'Artagnan* savait.

De même, dans les textes juridiques, par exemple les textes de loi ou les conventions européens, il y a très peu de pronoms : *la Commission européenne*, notamment, est très souvent citée, mais presque toujours en reprenant son nom (*Commission*). Si l'on recommence une chaîne à chaque nouveau nom propre, ce genre de textes n'auraient pas de chaîne *Commission*, alors même qu'il s'agit de l'un des acteurs les plus important du texte, très souvent cité. Pour ces raisons, nous ne redémarrons pas de chaînes à chaque nouveau nom propre.

Nous cherchons aussi à résoudre des phénomènes que Schnedecker (1997), Longo (2013) ou Democrat ne cherchent pas à résoudre, c'est le cas des cataphores (voir l'exemple 9, p. 20), des pronoms personnels de première et deuxième personnes dans le discours direct (voir l'exemple 18, p. 81), ou encore du sujet zéro des verbes coordonnés[5] (voir l'exemple de la figure 5.9, p. 93).

1.3 Nos principaux choix d'annotation

Nous listons ici les principaux choix d'annotation que nous avons faits.

Qu'est-ce qu'une expression référentielle? Notre programme annote les syntagmes nominaux, que leur tête soit un nom ou un pronom. Il annote ainsi tous les pronoms, qu'ils soit relatifs ou réfléchis. Il n'annote cependant pas d'adverbe

2. C'est un point dont il faut avoir conscience lors de l'évaluation et de la comparaison de RefGen et de notre programme.

3. Cela ne signifie pas que n'avons pas adapté nos choix précédents : nous avons ainsi décidé de ne pas annoter les « sujets zéros » des verbes à l'impératif, par exemple. Nous réservons le « sujet zéros » pour les coordinations.

4. Extrait des *Trois Mousquetaires* de Dumas.

5. Democrat les annote.

(notamment *ici, là*), sauf quand ils sont pronoms relatifs (*où*). Nous annotons aussi les appositions nominales, comme dans

(7) [Le président de la République, [Emmanuel Macron]$_i$]$_i$...

Nous annotons aussi les déterminants possessifs (*[[son]$_i$ chat]$_j$*). Nous annotons également les groupes (*[[Pierre]$_i$ et [Marie]$_j$]$_k$*).

Délimitation Notre programme annote l'ensemble du syntagme, y compris les appositions, qu'elles soient nominales (comme dans l'exemple 7), adjectivales (ce qui inclut les relatives) ou participiales, sauf lorsqu'elles sont l'équivalent d'un circonstant placé en début de phrase comme dans *Consul, César...*, équivalent de *Alors qu'il était consul, César...*, ou encore dans *Retraité de fraîche date, François Hollande a disparu des écrans de télévisions*, etc. Nous suivons en cela le manuel d'annotation du *French Treebank* (Abeillé, Toussenel et Chéradame 2015, p. 18) ; c'est ce corpus qui sert en effet de corpus d'entraînemenet à la plupart des analyseurs syntaxiques.

Sujet zéro Notre programme annote les sujets zéros des verbes coordonnés, comme dans :

(8) a. [Pierre]$_i$ mange et [ø]$_i$ boit.

Nous annotons le verbe plutôt que d'introduire un symbole « ø » :

b. [Pierre]$_i$ mange et [boit]$_i$.

Ce choix a une justification technique : nous ne voulons pas modifier le texte d'origine, mais seulement lui ajouter des annotations. Nous n'annotons pas les sujets zéros des verbes non conjugués.

Pronoms des deux premières personnes Les pronoms de première et deuxième personne sont annotés non seulement dans le texte principal, mais aussi dans les citations au discours direct. Dans ce dernier cas, le programme cherche à établir la coréférence avec une personne hors de la citation, comme expliqué page 81.

Cataphore Notre programme cherche les cataphores, comme dans :

(9) Quand [il]$_i$ était petit, [Paul]$_i$ avait peur de l'eau.

Nom propre Notre programme ne recommence pas une chaîne à chaque nom propre, comme le fait RefGen. Au contraire, il réunit même dans une même chaîne les désignations différentes d'une même entité, comme

(10) [Paris]$_i$... La [Ville lumière]$_i$

Anaphores fidèles et infidèles Notre programme cherche à continuer la chaîne s'il y a reprise nominale, que ce soit par une anaphore fidèle (reprise du même lemme) ou une anaphore infidèle (reprise par un synonyme, un hyperonyme ou un hyponyme (*Le chat... Cet animal*). Cependant, il est parfois délicat de savoir s'il y a coréférence dans le cas des entités abstraites. Nous renvoyons le lecteur à notre mémoire de Sciences du Langage (Oberlé 2016), où nous consacrons deux chapitres à ces problèmes (chapitre 1 et 5).

Voici les principales différences avec Democrat (ce qui n'est pas listé ici est commun avec nos choix) :

- Democrat n'annote pas les pronoms réfléchis,
- Democrat annote certains attributs, même quand il s'agit clairement d'une « étiquette » (*Marie est vendeuse*), ce que nous ne faisons pas. Par contre, nous cherchons à annoter l'attribut lorsqu'il établit l'égalité de deux entités (*Londres est la capitale de l'Angleterre*), ce que ne fait pas toujours Democrat. En ce sens, nous nous plaçons plutôt du côté de Frege que de Democrat,
- Democrat n'annote pas le syntagme entier, mais seulement une partie de celui-ci (voir notre discussion ci-dessus). Pour notre part, nous annotons le syntagme entier. De plus, nous considérons que les appositions, nominales ou adjectivales, font partie du syntagme et donc nous les incluons dans les expressions référentielles.

Chapitre 2

La résolution automatique de la coréférence

2.1 Présentation des différentes approches

La perspective du TAL est quelque peu différente de celle de la linguistique : le TAL ne vise pas l'étude d'un phénomène linguistique mais l'analyse automatique de documents en vue d'en extraire des informations (comme des entités nommées[1]) ou de les modifier (comme les traduire, les résumer, les annoter).

Annoter un document peut consister, par exemple, à trouver tous les syntagmes nominaux (dont la tête syntaxique sont des noms ou des pronoms) ; ils sont entre crochets dans la phrase suivante :

(11) [Il] semble que [mon chat] mange [le gâteau [qu'] [il] avait chipé à [ton chien]]. [Félix] est [un vrai goinfre].

Mais tous les pronoms ne réfèrent pas : dans l'exemple précédent, le premier *il* (*il semble*) est impersonnel et ne peut entrer dans une chaîne de référence. L'annotation peut donc identifier ces expressions linguistiques, et les noter comme non référentielles.

1. Des expressions désignant des personnes, des organisations, des lieux, des dates, des quantités, etc. Voir par exemple Nouvel, Ehrmann et Rosset (2015).

Une étape supplémentaire consiste à expliciter les *relations anaphoriques*, c'est-à-dire à associer chaque anaphore à un antécédent[2]. Dans l'exemple, il faudrait associer *il* avec *chat* et *qu'* avec *gâteau*.

Il reste enfin à trouver les *relations de coréférence*, comme *chat* et *Félix*, dernière étape que les systèmes TAL ont à franchir pour détecter avec succès les chaînes de référence. Nous obtenons donc une chaîne (trois maillons au moins, donc *gâteau* et *qu'* ne forment pas une chaîne) :

(12) Il semble que [mon chat]$_i$ mange le gâteau qu'[il]$_i$ avait chipé à ton chien. [Félix]$_i$ est un vrai goinfre.

Les étapes que nous venons de présenter mènent à *la détection automatique des chaînes de référence* ; elles sont regroupées en deux tâches :

- la détection des expressions référentielles (la première étape),
- la détection des relations entre ces différentes expressions (les autres étapes).

Certains systèmes ne font que la deuxième tâche ; la première doit alors être réalisée par un programme auxiliaire[3]. Notre recherche concernera toutes les étapes de la détection des chaînes.

Ces tâches ne sont pas triviales. D'abord, une même forme peut appartenir à différentes catégories grammaticales (*le* peut être aussi bien déterminant que pronom personnel) : il faut donc pouvoir les désambiguïser. Ensuite, il faut déterminer si le pronom est ou non référentiel (cas de « *il* semble »), trouver le bon antécédent (pour le deuxième *il* de l'exemple, l'antécédent est-il *gâteau* ou *chat* ?), et les bons liens de coréférence (*Félix* est-il mon chat ou ton chien ?). Mais, ce que l'être humain parvient à faire « naturellement » et en s'aidant du contexte extra-linguistique, l'ordinateur ne le peut pas. Il faut donc lui fournir des moyens de résoudre autrement ces liens : soit par des *règles* définies manuellement, soit par des *régularités* trouvées par un algorithme d'apprentissage automatique à partir de données similaires, deux approches que nous allons maintenant présenter.

2. Cela ne vaut que pour les pronoms dits *anaphoriques*. D'autres pronoms sont dits *déictiques* : ils renvoient non pas à un autre segment du texte, mais directement à une entité extérieure. C'est le cas, par exemple, du pronom de première personne *je* dans un discours : il renvoie au locuteur.

3. Parmi les sytèmes français que nous présenterons dans la suite, RefGen réalise les deux tâches, alors que CROC ne fait que la deuxième.

2.1.1 Systèmes à base de règles

Les premières approches (initiées dès les années 1970[4]) fonctionnent avec des règles qui utilisent les propriétés morphologiques (par exemple l'accord en genre et en nombre entre une anaphore et son antécédent) et syntaxiques (par exemple la théorie du liage de Chomsky (1981)) des expressions linguistiques. Des heuristiques[5] viennent compléter ces propriétés linguistiques (la proximité ou *récence*, la relation sémantique (synonymie, hyperonymie), etc.).

D'autres approches utilisent des théories cognitivistes et font appel à la notion de *saillance*, « qui indique dans quelle mesure [une] entité est présente et disponible dans la mémoire du lecteur » (Victorri 2005). La saillance est déterminée par le nombre de mentions d'un référent, sa récence, sa mise en relief, sa fonction grammaticale (le sujet est considéré comme plus saillant que l'objet, par exemple), etc. (voir aussi Landragin (2005) et Schnedecker (2011)).

D'autres approches encore appliquent des règles en plusieurs passes, de plus en plus précises (Lee, Peirsman, Chang, Chambers, Surdeanu et Jurafsky 2011). La première passe détecte les mentions ; la seconde cherche à identifier les locuteurs d'un texte et à les associés à des pronoms de première et deuxième personne ; une autre passe cherche les relations hyperonymiques (grâce à *WordNet*) ; une autre encore cherche, pour chaque nom propre des « alias » à partir des infoboxes de Wikipédia (par exemple « American Online Company » et « AOL ») ; etc. (il y a 13 passes en tout).

Certaines de ces approches ont besoin de ressources linguistiques importantes (voir par exemple les travaux de Haghighi et Klein (2009), et leur discussion par Raghunathan (2010)), comme des analyseurs syntaxiques (pour trouver les dépendances entre les mots, par exemple trouver que tel déterminant dépend de tel nom), des dictionnaires de synonymes ou des *ontologies*[6] afin de trouver l'antécédent d'une anaphore infidèle (« mon chat... cet animal... »). Ces ressources sont néanmoins difficiles à construire, peu fiables et coûteuses. C'est pourquoi la plupart des systèmes préfèrent utiliser peu de connaissances externes (c'est pourquoi ils sont dit *knowledge-poor*) et n'exploiter que des *indices de surface*, facilement calculables automatiquement, comme

4. On se rapportera à Mitkov (2002) pour une liste des premiers systèmes et de leur fonctionnement.

5. Une heuristique permet de fournir rapidement une solution (qui n'est pas optimale) à un problème donné (Longo 2013).

6. Une ontologie (par exemple *WordNet*, développé à l'université de Princeton, pour la langue anglaise), en TAL, est une ressource sémantique qui organise le lexique d'une langue de façon hiérarchique et permet par exemple de trouver les hyperonymes d'un terme.

la catégorie grammaticale des expressions linguistiques ou leurs propriétés morphologiques.

De nombreux algorithmes ont été mis en place (Victorri (2005) et Longo (2013) en présentent une liste), mais il y en a peu pour le français (*cf.* Trouilleux (2001) et M. Dupont (2003), qui a implémenté son système dans « un logiciel d'étude » : *Calcoref*, qui est cependant indisponible), et moins encore ont donné lieu à un système opérationnel (Longo 2013). Seule Longo (2013) propose *RefGen*, un outil de détection de la coréférence qui peut être utilisé en l'état.

Cependant, les règles utilisées par ces approches (notamment les approches qui utilisent peu de ressources linguistiques externes) sont généralement fondées sur l'intuition du concepteur du système et sont construites en observant des corpus d'étude. Par exemple, Longo (2013) montre que l'heuristique de Mitkov (1998) consistant à privilégier les syntagmes définis comme antécédents de pronoms n'est pas valable pour tous les genres discursifs.

En outre, la plupart des ces approches attribuent des scores et des poids à différentes propriétés des paires antécédent–pronom testées. Par exemple, le système de Mitkov (1998) attribue des scores de -1, 0, 1, 2, etc. selon que l'antécédent est un groupe nominal défini (plutôt qu'indéfini), qu'il est le thème de la phrase (plutôt que rhème), etc. Or ces scores sont arbitraires, établis selon l'interprétation du taliste à partir d'un corpus d'étude, et loin de toute théorie linguistique (Landragin 2004). Cela limite la réutilisation de tel système sur des données nouvelles et des corpus différents.

D'autres approches, qui comblent en partie ces limites, ont vu le jour avec la mise à disposition de grands corpus annotés en coréférence : les méthodes statistiques.

2.1.2 Systèmes par apprentissage statistique

L'apprentissage statistique est une méthode qui permet de découvrir automatiquement des régularités dans de larges quantités de données, sans avoir besoin de définir des règles et leurs exceptions à la main (Lassalle 2015). Mais, pour être opérante, cette technique demande souvent de grands corpus d'entraînement, annotés manuellement. C'est pourquoi elle n'est apparue, pour la détection de la coréférence, que depuis les

années 2010, lorsque de grandes campagnes d'évaluation[7] ont mis à disposition de tels corpus.

Les systèmes d'apprentissage statistique utilisent des propriétés des expressions linguistiques. Ces propriétés sont appelées des *traits*. Certains de ces traits sont valables indépendamment de la langue (Recasens 2010), comme la distance entre deux expressions ou bien le niveau de dépendance dans un arbre syntaxique[8]. D'autres sont spécifiques à la langue, comme le genre et le nombre grammatical, voire à certains corpus, comme la distance entre deux expressions en nombre de tours de parole, qui est spécifique aux corpus oraux (Désoyer, Landragin, Tellier, Lefeuvre et Antoine 2015). Les systèmes pour l'anglais et le français utilisent généralement un jeu de traits standards[9], défini par Soon, H. T. Ng et Lim (2001) et V. Ng et Cardie (2002), que les chercheurs adaptent et implémentent en fonction de leurs besoins.

Pour résoudre la coréférence, la plupart de ces systèmes déterminent la *classe* de chaque paire d'expressions référentielles du texte : si les expressions d'une paire sont coréférentielles, alors la paire appartient à la classe 1 ; sinon la paire appartient à la classe 2[10]. Les chaînes sont ensuite aisément construites en regroupant les expressions des paires coréférentielles par transitivité : si A et B sont coréférentiels et que B et C le sont aussi, alors le système peut créer la chaîne {A, B, C} (Longo 2013).

7. Une campagne d'évaluation permet de comparer différents systèmes sur une tâche donnée à partir de ressources (notamment un corpus) communes. Par exemple, la campagne de 2011 adossée à la conférence CoNLL (Conference on Natural Language Learning), visait à l'évaluation de la coréférence (http ://conll.cemantix.org/2011/). On peut également citer MUC *(Message Understanding Conference)*, ACE *(Automatic Content Extraction)* ou encore SemEval *(Semantic Evaluation)*.

8. L'expression est-elle par exemple le sujet principal, ou est-elle le sujet d'un verbe d'une proposition subordonnée ou relative ?

9. Par exemple : Quelle est la distance entre les mentions (en nombre de phrases) ? La première mention est-elle un pronom ? La deuxième mention est-elle un pronom ? La deuxième mention est-elle un syntagme nominal défini ? ...démonstratif ?

10. Nous nous situons dans la perspective d'approches *supervisées*, c'est-à-dire qui apprennent un modèle à partir de données d'entraînement. D'autres approches sont dites *non supervisées* : elles n'utilisent pas de corpus d'entraînement mais tentent de *regrouper* (*cluster* en anglais, d'où le nom de la tâche associée : le *clustering*) à partir des traits selon un calcul de « distance » (*i.e.* de similarité). Ces approches sont cependant rares parce que, même si elles ne demandent pas de corpus d'entraînement, leur complexité ne les rends pas rentables

Des *classifieurs*[11] sont utilisés pour déterminer si deux expressions sont coréférentielles ou non : il s'agit d'algorithmes statistiques qu'il n'est pas possible de modifier[12] et qu'il faut donc considérer comme des outils dont il faut faire le meilleur usage possible. Or, utiliser les algorithmes de la meilleure façon possible, c'est leur donner le meilleur corpus d'entraînement possible avec le meilleur jeu de traits possible. Ils pourront alors trouver des régularités (plutôt que des règles) dans les traits de ce corpus et fournir un *modèle* utilisable pour traiter des données nouvelles, c'est-à-dire découvrir des chaînes dans des textes qui ne sont pas annotés en coréférence. Le choix de l'algorithme et des traits à utiliser font l'objet d'*expérimentations* (voir par exemple le travail effectué par Désoyer, Landragin, Tellier, Lefeuvre et Antoine (2015)), afin de sélectionner la meilleure combinaison, c'est-à-dire celle qui permet d'obtenir la meilleure performance.

Cependant, les grands corpus annotés en coréférence, notamment ceux fournis par les campagnes d'évaluation, ne sont pas en français, ou sont de trop petite taille pour servir de corpus d'entraînement aux algorithmes (Désoyer, Landragin, Tellier, Lefeuvre, Antoine et Dinarelli 2016). C'est pourquoi la plupart des outils existants (on en trouvera une liste, non exhaustive, dans Longo 2013. Dans un article nommé « "Supervised noun phrase coreference research : The first fifteen years" », V. Ng (2010) propose de plus une revue des systèmes par apprentissage depuis 1994 jusqu'en 2010) ne sont disponibles que pour l'anglais ou des langues bien dotées en corpus.

11. Un classifieur est un algorithme qui permet d'assigner une classe à un élément. Il est possible de créer des classifieurs manuellement ; par exemple, on peut chercher à attribuer l'une des trois classes *pronom personnel singulier* (PPS), *pronom personnel pluriel* (PPP) ou *autre catégorie* (AC) à un mot m. On dispose d'une liste l qui contient les pronoms personnels du français (*je, tu, il, elle, nous, vous, ils, elles, on*). Si m est dans l, on regarde s'il se termine par un « s » : si c'est le cas, alors il appartient à la classe PPP, sinon à la classe PPS. Si m n'est pas dans l, alors il appartient à la classe AC. Ce type d'algorithme s'appelle un « arbre de décision » ; il existe beaucoup d'autres types d'algorithme. L'apprentissage statistique va tenter de construire un tel raisonnement automatiquement (celui-là ou un autre qui arrive au même résultat), à partir d'un grand nombre de données qu'on lui présente. Ces données se présentent sous forme d'ensembles de *traits* (dans notre exemple : m appartient-il à l? m finit-il par « s » ?) associés à une classe. Lorsque le système « verra » que tous les mots qui se terminent par « s » et appartiennent à l sont de la classe PPP, il construira *un modèle* avec cette information. Lorsque le modèle sera appliqué à une donnée nouvelle (un mot) qui appartiendra à l et finira par un « s », il lui donnera la classe PPP. Les algorithmes utilisés dans la résolution de la coréférence sont bien plus complexes, et de ce fait ininterprétables par un être humain : ce sont des « boîtes noires ».

12. Écrire un nouvel algorithme requiert des connaissances avancées en mathématiques et statistiques. Lassalle (2015), dans sa thèse d'informatique, propose plusieurs pistes possibles.

Pour le français, il n'existe pour l'heure que le corpus ANCOR[13] (Muzerelle, Lefeuvre, Antoine, Schang, Maurel, Villaneau et Eshkol 2011), qui a servi à entraîner CROC[14] (Désoyer, Landragin, Tellier, Lefeuvre et Antoine 2015 ; Désoyer, Landragin, Tellier, Lefeuvre, Antoine et Dinarelli 2016), créé au Lattice[15] et utilisé dans Democrat. À notre connaissance, il s'agit du seul outil de détection automatique des chaînes de référence[16] par apprentissage statistique disponible pour le français.

Les méthodes par apprentissage statistique réussissent mieux à traiter des données nouvelles ou non conformes aux attendus des règles (Lassalle 2015). Leur point fort est cependant aussi leur point faible : ces méthodes ont besoin d'un corpus d'entraînement qui soit à la fois de grande taille, représentatif de l'ensemble des productions que l'on souhaite traiter[17], et bien annoté. Des systèmes hybrides, qui allient les règles linguistiques aux algorithmes statistiques, ont alors vu le jour pour combiner le meilleur des approches linguistiques et des approches statistiques.

2.1.3 Systèmes hybrides

Hartrumpf (2001) propose un algorithme hybride pour l'allemand (Mitkov (1999) évoque un algorithme analogue pour l'anglais). Son algorithme utilise un ensemble de règles pour sélectionner, pour chaque anaphore, des antécédents possibles, puis fait usage de statistiques (obtenues à partir d'un corpus annoté à la main) pour déterminer le meilleur.

Les approches statistiques « traditionnelles » utilisent en général entre dix et vingt traits pour entraîner les algorithmes. Uryupina (2006, 2007), elle, en utilise 351 (cependant, afin de réduire le temps de calcul nécessaire, l'implémentation finale qu'elle présente, le système Corry (Uryupina 2010), n'en conserve que 64). La plupart des traits supplémentaires encodent des phénomènes linguistiques utilisés par les systèmes à base de règles, comme la similarité sémantique (calculé à partir de *WordNet*, voir la note 6), la saillance ou encore la structure syntaxique interne de l'expression référen-

13. ANCOR est un corpus de français oral uniquement (des conversations, par exemple les échanges d'accueil téléphonique de l'Université de Bretagne-Sud) contenant 488 000 mots 115 000 expressions référentielles annotées. Les relations anaphoriques et les coréférences y sont annotées.

14. *Coreference Resolution for Oral Corpus*.

15. Laboratoire de Frédéric Landragin, co-directeur de la thèse.

16. Contrairement à RefGen, CROC n'a pas de module pour identifier les expressions référentielles. Cette tâche préalable doit être effectuée par un autre programme.

17. Pour un système généraliste, il faut donc prendre en compte toutes les productions de la société.

tielle, c'est-à-dire le type de déterminant, la position des modifieurs (avant ou après le nom) et leur type (restrictifs ou non). Son système, cependant, fait appel à des ressources sémantiques externes anglaises (comme *WordNet*), et est donc difficilement adaptable tel quel pour d'autres langues.

Weissenbacher (2008)[18] propose une approche qui repose sur des réseaux bayésiens. Constatant que les données d'entrée (annotation, ressources linguistiques) sont en général imparfaites (incomplètes, erronées ou ambiguës), l'auteur propose un algorithme probabiliste qui permet d'unifier connaissances linguistiques et indices de surface en corrigeant et complétant les premières avec les seconds. Le système, cependant, ne cherche à résoudre que les phénomènes d'anaphores (notamment les pronoms), et seulement pour l'anglais.

2.2 Positionnement

Il y a donc trois types d'approches : les sytèmes à base de règles linguistiques, ceux à base d'apprentissage, et enfin ceux qui tentent d'intégrer linguistique et statistique. Dans chacune de ces approches, on trouve une multitude de perspectives, d'implémentations, mais aussi de tâches qui varient plus ou moins les unes des autres : certains systèmes se limitent à certains phénomènes, comme certains types d'anaphores (pronominales ou nominales), ou certains types de référents (concrets (notamment en liaison avec la détection des entités nommées) ou événementiels). Il devient difficile de s'y retrouver (Mitkov (1999, 2002) et V. Ng (2010) tentent des récapitulatifs), mais il convient de nous positionner parmi tous ces travaux.

Les approches par apprentissage statistique comme les approches hybrides demandent des corpus d'entraînement préalablement annotés[19]. Or, pour le français, il n'existe qu'un seul corpus annoté en coréférence (ANCOR), mais il ne concerne que le français oral. Il ne nous est donc pas possible d'utiliser ces techniques pour l'heure. Parmi les approches à base de règles, nous avons vu qu'il y a des approches dites *knowledge poor* qui tentent de diminuer l'usage de ressources linguistiques. Néanmoins, notre système devra être capable de détecter *toutes* les relations de coréférence (pour former des chaînes de référence), y compris les anaphores infidèles, par exemple. Or ce genre de relations ne peut être détecté qu'à l'aide de ressources sémantiques (dictionnaires de synonymes, d'hyperonymes). De même, pour la résolution des anaphores pronomi-

18. Voir également Weissenbacher et Nazarenko (2007).
19. Bien qu'il existe des approches *non supervisées* pour lesquels les besoins en corpus sont moindres.

nales des entités nommées, une ressource décrivant avec quelque détail ces entités doit pouvoir être accessibles. C'est pourquoi nous proposerons un système à base de règles linguistiques, qui aura besoin d'importantes ressources linguistiques.

Chapitre 3

Présentation d'ODACR et de son architecture générale

Nous présenterons dans ce chapitre l'architecture générale de notre programme. Il s'agit d'un programme *end-to-end*, c'est-à-dire qu'il accepte en entrée du texte tout venant, sans qu'il y est besoin d'annotations préalables. En sortie, on trouve le même texte, avec des annotations, qui peuvent être *intégrées* dans le texte (on a alors un seul fichier) ou bien *déportées* dans un document à part (on a alors deux fichiers, l'un avec le texte original, l'autre avec les annotations), au format Glozz.

3.1 Les modules

Notre programme, écrit en Python, a une architecture modulaire, c'est-à-dire que plusieurs modules se succèdent, chacun s'occupant d'une tâche particulière. Le premier module s'occupe de la tokénisation et des tâches associées. Il découpe le texte en *tokens*, c'est-à-dire en unités élémentaires qui n'ont pas être décomposées plus avant pour les besoins du programme. Ces tokens peuvent être des mots simples (par exemple *table*), des mots composés (*arc-en-ciel, hôtel de ville*) ou des signes de ponctuation. Cette étape nécessite l'utilisation de ressources lexicales et de règles pour la détection d'entités nommées (voir le chapitre 4). Certaines opérations modifient la représentation interne du texte, notamment pour tenir compte du discours direct et des incises (voir le chapitre 5). Enfin, le module de tokenisation segmente le texte en phrases.

Le second module est un *wrapper*, c'est-à-dire une « coquille » qui appelle un autre programme. Cet autre programme est un analyseur syntaxique. Nous avons choisi d'utiliser Talismane (Urieli 2013), mais d'autres analyseurs peuvent être utilisés, sous réserve de fournir des règles de conversion pour les jeux d'étiquettes (si cet autre analyseur utilise un autre jeu d'étiquettes que celui de Talismane). L'analyseur syntaxique (voir le chapitre 5 calcule les dépendances entre les différents tokens, en indiquant par exemple que tel déterminant est rattaché à tel nom, que tel nom est le sujet de tel verbe, etc. La sortie de ce second module intègre les informations de l'analyseur avec celle du tokeniseur, notamment en ce qui concerne les informations des entités nommées et du discours direct.

Le troisième module s'occupe de la correction de la sortie de l'analyseur. Ce module est optionnel : il n'a pas lieu d'être si l'analyseur ne fait pas d'erreurs (mais nous n'avons pas trouvé un tel analyseur). Ce module est décrit en détail dans le chapitre 5.

Le quatrième module s'occupe de la détection de la coréférence. Il va d'abord identifier les expressions référentielles, puis faire trois « passes » sur le texte (voir le chapitre 6) : d'abord pour trouver les antécédents des anaphores liées, puis ceux des anaphores libres, et enfin résoudre la coréférence. Pour cette dernière opération, des ressources lexicales décrivants des relations sémantiques (hyperonymes) sont utilisées (voir le chapitre 4). Il existe un mode *chunker* qui permet de ne pas appeler ce module : dans ce cas, les expressions référentielles sont détectées, mais elles ne sont pas reliées en fonction de leur coréférence (cela est utile si l'on veut annoter la coréférence manuellement).

Enfin, le dernier module s'occupe de l'exportation des annotations. Il calcule un nom pour les chaînes, à partir du contenu du maillon le plus long (mais en privilégiant les entités nommées : *Macron* est plus pertinent que *le président de la république*, même s'il est plus court), puis il exporte les annotations. Pour l'instant, il n'exporte qu'au format SACR, que nous avons décrit dans Oberlé (2016), et un script Perl permet de passer au format Glozz, qui lui-même est directement lisible par Analec (Victorri 2011) et TXM (Heiden, Magué et Pincemin 2010).

La figure 3.1 montre l'architecture d'ODACR.

3.2 Exemple

Dans les chapitres suivants, nous décrirons successivement les ressources lexicales utilisées par le tokeniseur et le module de résolution de la coréférence (chapitre 4), la

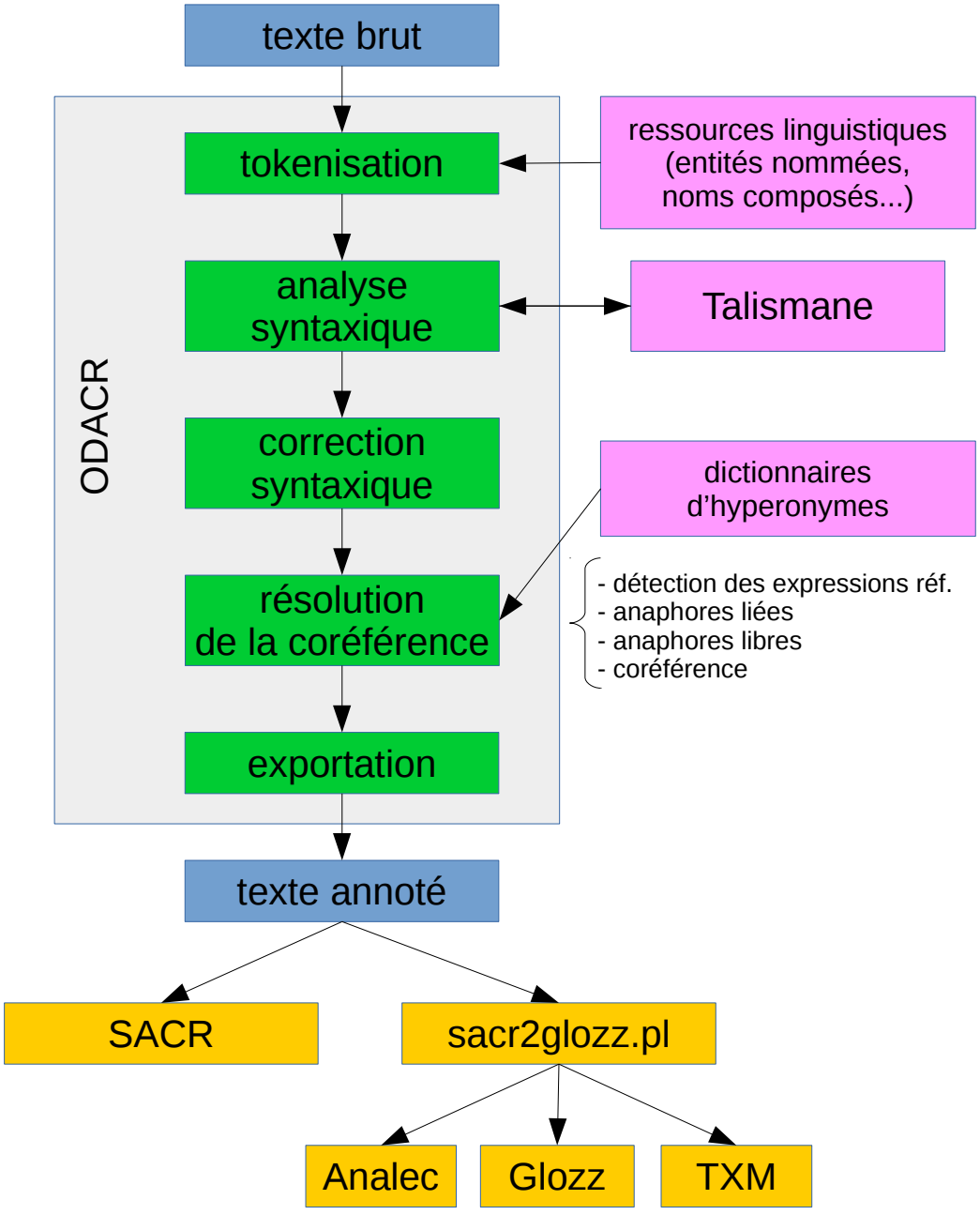

FIG. 3.1 : Architecture d'ODACR.

tokénisation et l'analyse syntaxique (chapitre 5) et le détail de l'algorithme utilisé pour la détection des chaînes de référence (chapitre 6), mais nous présentons d'abord un exemple (sans entrer dans les détails, puisque chaque module est développé dans un chapitre à part).

Soit le paragraphe :

(13) Mon chat regarde le discours du secrétaire de l'Organisation des Nations-Unies à la télévision. Il veut rester sur le canapé et refuse de boire son lait. Cet animal est un vrai paresseux.

Le premier module (le tokeniseur) va segmenter le texte en tokens. Il faut faire attention à l'entité nommée « Organisation des nations-Unies », qui doit être considérée comme un seul token. Nous avons donc :

```
Mon     WORD    WORD    0       3       False   False   DET (mon: m,s): possessive
chat    WORD    WORD    4       8       False   False   NC (chat: m,s)
regarde WORD    WORD    9       16      False   False   V (regarder: ,) VS (regarder: ,)  VIMP (regarder: ,)
le      WORD    WORD    17      19      False   False   DET (le: m,s): definite CLO (le: m,s): personal
discours        WORD    WORD    20      28      False   False   NC (discours: m,s)  NC (discours: m,p)  VIMP (
    ↪ discourir: ,) V (discourir: ,)
du      WORD    WORD    29      31      False   False   P+D (le: m,s): definite
secrétaire      WORD    WORD    32      42      False   False   NC (secrétaire: m,s) NC (secrétaire: f,s)
de      WORD    WORD    43      45      False   False   NC (de: m,s)  NC (de: m,p)  P (de: ,)     P (de: ,)
Organisation des Nations - Unies        WORD    WORD    48      78      False   False   NE (<ONU>: f,s): organization
à       WORD    WORD    79      80      False   False   P (à: ,)      P (à: ,)
la      WORD    WORD    81      83      False   False   DET (le: f,s): definite CLO (le: m,s): personal
télévision      WORD    WORD    84      94      False   False   NC (télévision: f,s)
.       PUNCT   DOT     94      95      True    False
Il      WORD    WORD    97      99      False   False   CLS (il: m,s): personal
veut    WORD    WORD    100     104     False   False   V (vouloir: ,)
rester WORD     WORD    105     111     False   False   VINF (rester: ,)
sur     WORD    WORD    112     115     False   False   ADJ (sur: m,s) P (sur: ,)  P (sur: ,)
le      WORD    WORD    116     118     False   False   DET (le: m,s): definite CLO (le: m,s): personal
canapé WORD     WORD    119     125     False   False   NC (canapé: m,s)     VPP (canaper: m,s)
et      WORD    WORD    126     128     False   False   CC (et: ,)    CC (et: ,)
refuse WORD     WORD    129     135     False   False   VIMP (refuser: ,)    VS (refuser: ,) V (refuser: ,)
de      WORD    WORD    136     138     False   False   NC (de: m,s)  NC (de: m,p)  P (de: ,)     P (de: ,)
boire   WORD    WORD    139     144     False   False   NC (boire: m,s) NC (boire: f,s) VINF (boire: ,)
son     WORD    WORD    145     148     False   False   DET (son: m,s): possessive
lait    WORD    WORD    149     153     False   False   NC (lait: m,s)
.       PUNCT   DOT     153     154     True    False
Cet     WORD    WORD    156     159     False   False   DET (ce: m,s): demonstrative
animal WORD     WORD    160     166     False   False   ADJ (animal: m,s)    NC (animal: m,s)
est     WORD    WORD    167     170     False   False   ADJ (est: m,s) ADJ (est: m,p) NC (est: m,s) NC (est: m,p
    ↪ ) V (être: ,)
un      WORD    WORD    171     173     False   False   DET (un: m,s): indefinite
vrai    WORD    WORD    174     178     False   False   ADJ (vrai: m,s) NC (vrai: m,s) ADV (vrai: ,) ADV (vrai:
    ↪ ,)
paresseux       WORD    WORD    179     188     False   False   ADJ (paresseux: m,s) ADJ (paresseux: m,p) NC (
    ↪ paresseux: m,s) NC (paresseux: m,p)
.       PUNCT   DOT     188     189     True    False
```

Chaque token est représenté sur une ligne. Les colonnes représentent, dans l'ordre : la forme, le type de token, le sous-type de token, la position initiale dans le texte, la position finale, `True` si c'est une fin de phrase `False` sinon, `True` si c'est du discours direct `False` sinon, la catégorie grammaticale avec le lemme et les informations morpho-syntaxique (il peut y avoir plusieurs catégories grammaticales, il n'y a pas de désambiguïsation à cette étape). Parfois, il y a des indications supplémentaires, par exemple *possessive, demonstrative* pour les déterminants et les pronoms.

On voit donc ici les fin de phrases après les points (il n'y a rien de compliqué dans cet exemple).

Le second module va appeler l'analyseur syntaxique ; ici, c'est Talismane. Les figures 3.2 à 3.8 représente les sorties avant et après correction (c'est le troisième module) pour chacune des phrases. Ici, on voit que Talismane n'a fait aucune erreur. Les seules « corrections » à faire sont donc la segmentation de la phrase du milieu (qui contient deux propositions radicales) en deux et la séparation de l'amalgame « du » en « de le ».

Le quatrième module va d'abord chercher les expressions référentielles :

(14) [[Mon] chat] regarde [le discours [du secrétaire de [l'Organisation des Nations-Unies]]] à [la télévision]. [Il] veut rester sur [le canapé]. [Il] [refuse] de boire [[son] lait]. [Cet animal] est un vrai paresseux.

Nous avons modifié le texte original pour refléter la segmentation en deux de la phrase du milieu (c'est la représentation interne au programme).

Puis vient la détection de la coréférence : il s'agit de relier les expressions qui sont coréférentielles selon un algorithme complexe expliqué au chapitre 6. Nous obtenons :

(15) $[[\text{Mon}]_{\text{i}}\ \text{chat}]_{\text{j}}$ regarde [le discours [du secrétaire de [l'Organisation des $\text{Nations-Unies}]_{\text{k}}]_{\text{l}}]_{\text{m}}$ à $[\text{la télévision}]_{\text{n}}$. $[\text{Il}]_{\text{j}}$ veut rester sur $[\text{le canapé}]_{\text{o}}$ et $[\text{refuse}]_{\text{j}}$ de boire $[[\text{son}]_{\text{j}}\ \text{lait}]_{\text{p}}$. $[\text{Cet animal}]_{\text{j}}$ est un vrai paresseux.

Le dernier module va exporter les annotations. Pour l'heure, il utilise le format SACR (Script d'Annotation des Chaînes de Référence (Oberlé 2016)) :

```
{MonChat: {L7: Mon} chat} regarde {L1: le discours {L2: du secrétaire
↪ de {L3: l'Organisation des Nations-Unies}}} à {L4: la télévision
↪ }. {MonChat: Il} veut rester sur {L5: le canapé} et {MonChat:
↪ refuse} de boire {L6: {MonChat: son} lait}. {MonChat: Cet animal}
↪  est un vrai paresseux.
```

La figure 3.9 montre la représentation graphique dans SACR.

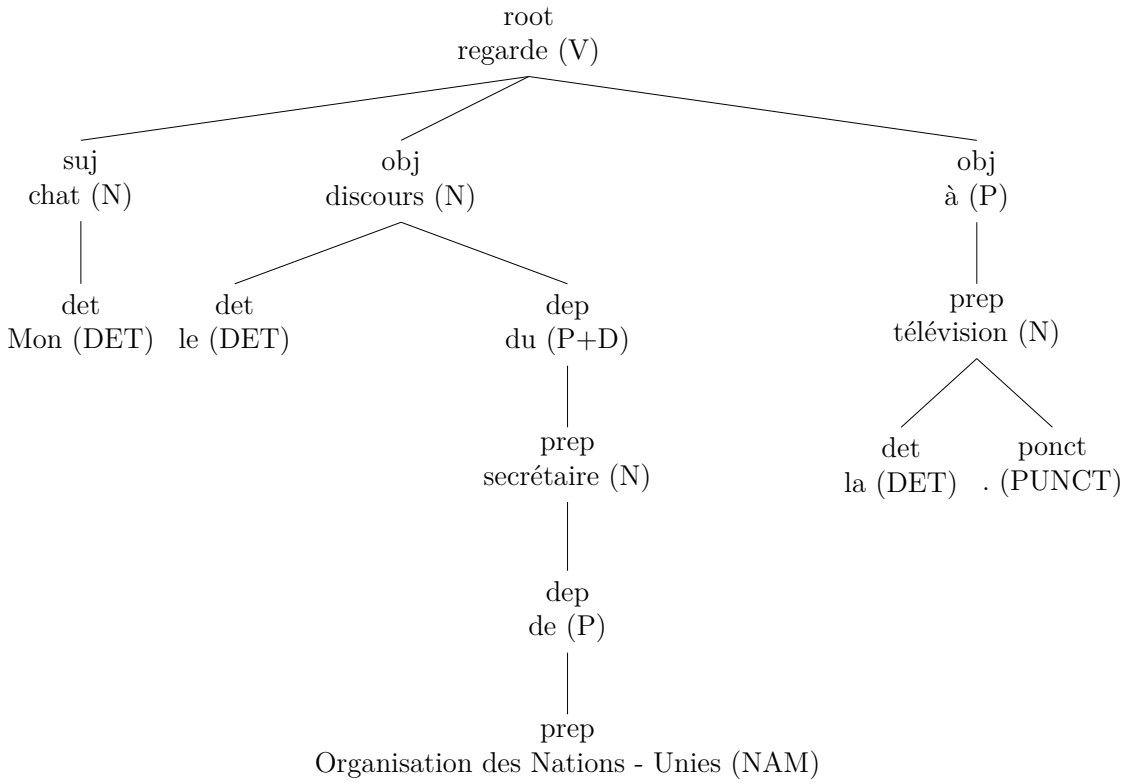

FIG. 3.2 : Sortie de Talismane avant correction (phrase 1).

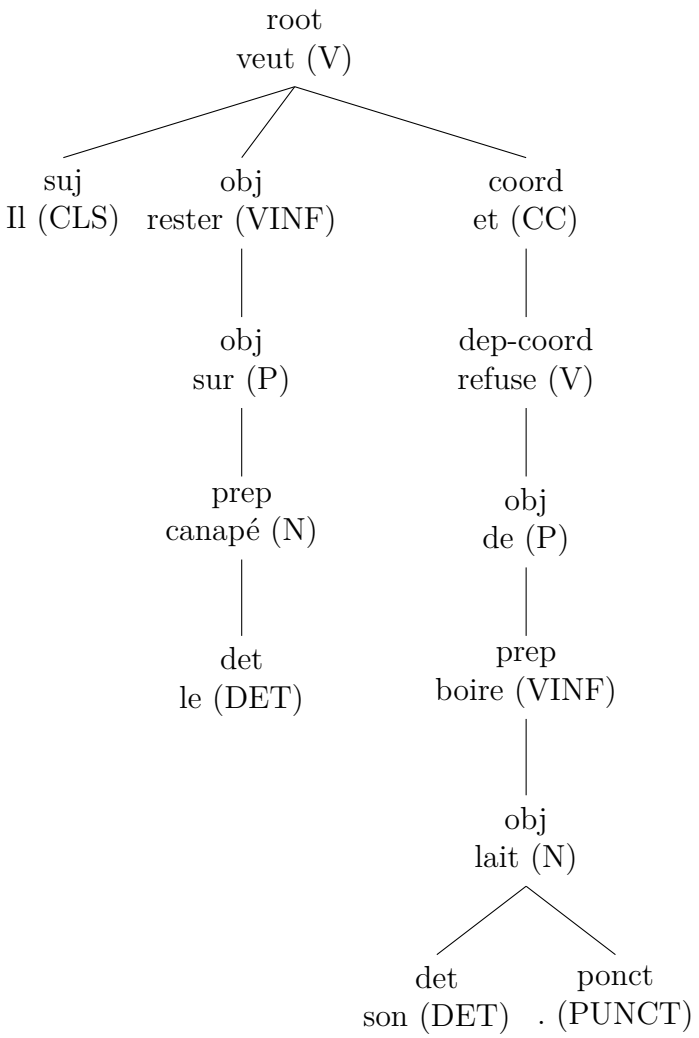

FIG. 3.3 : Sortie de Talismane avant correction (phrase 2).

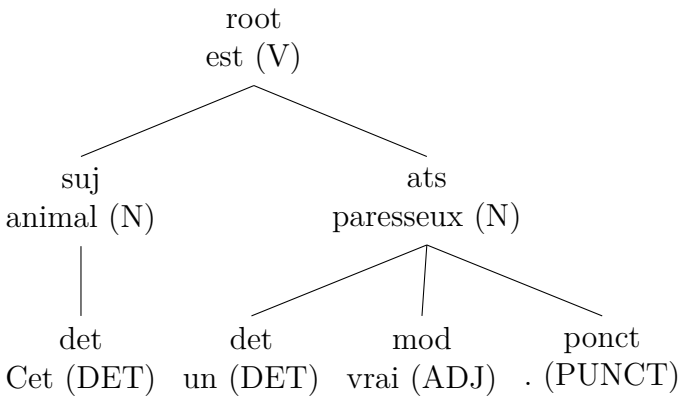

FIG. 3.4 : Sortie de Talismane avant correction (phrase 3).

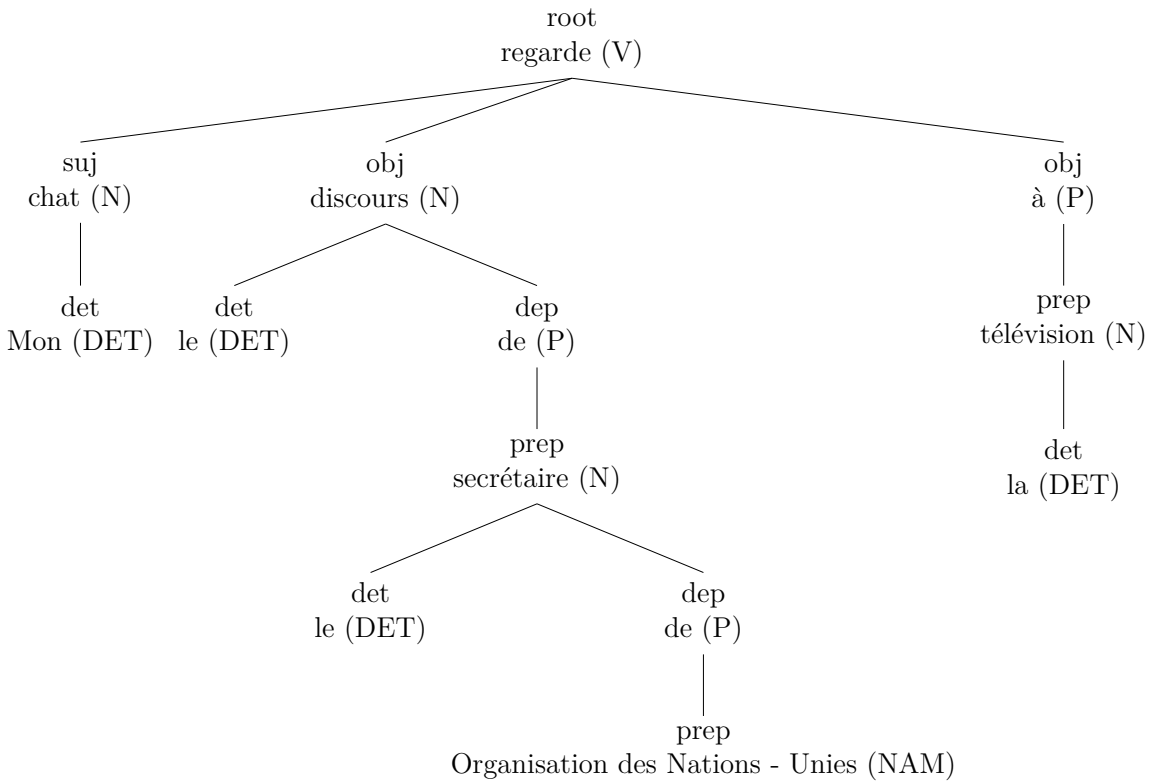

FIG. 3.5 : Sortie de Talismane après correction (phrase 1).

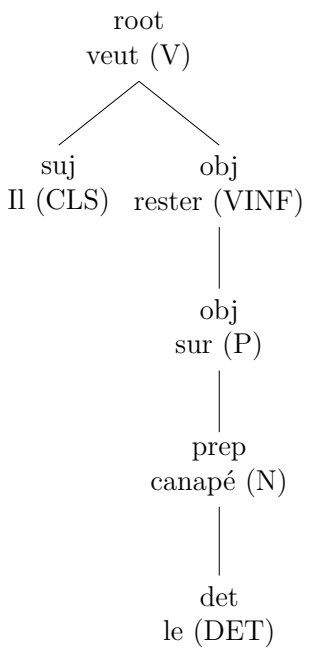

FIG. 3.6 : Sortie de Talismane après correction (phrase 2, premier segment).

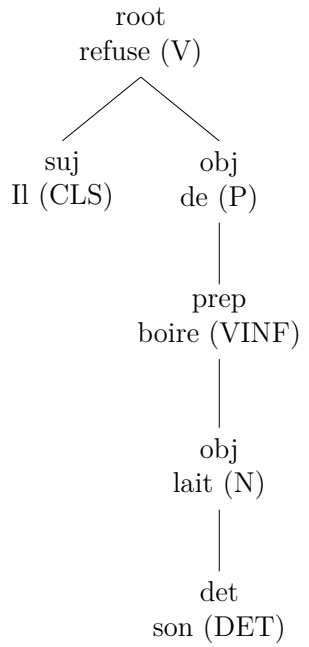

FIG. 3.7 : Sortie de Talismane après correction (phrase 2, deuxième segment).

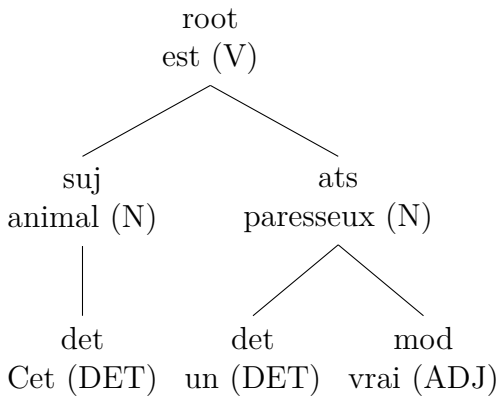

FIG. 3.8 : Sortie de Talismane après correction (phrase 3).

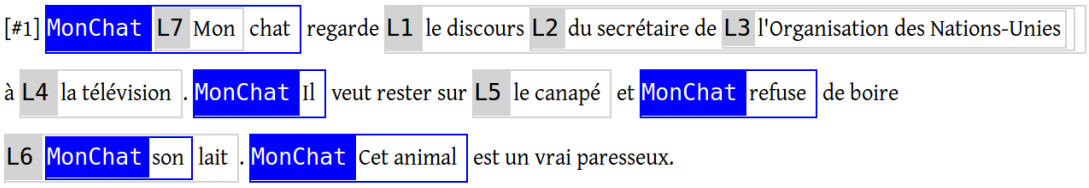

FIG. 3.9 : Le résultat dans SACR.

Chapitre 4

Besoins en ressources lexicales

Notre programme utilise des ressources lexicales pour deux tâches : la tokenisation et la résolution de la coréférence. La tokenisation ne peut se faire avec précision qu'avec l'aide de dictionnaires de noms communs et de noms propres. Ce sont en effet ces ressources qui permettent de définir les unités polylexicales, c'est-à-dire des unités lexicales qui s'écrivent en plusieurs « mots » : noms communs comme « petit pois », ou entité nommée comme « Ville lumière » (certaines de ces unités doivent cependant être cherchées à l'aide de règle : c'est le cas notamment des nombres, comme « trente quatre » ; il serait en effet difficile de créer un dictionnaire avec tous les nombres). Concernant la résolution de la coréférence, les ressources lexicales sont également utilisées pour la résolution des anaphores infidèles.

4.1 Les entités nommées

4.1.1 Motivation

Bien qu'il n'existe pas de définition « standard » des entités nommées (Nouvel, Ehrmann et Rosset 2015, p. 29), nous considérerons ici comme telles les *désignations propres de personnes, de lieux, d'organisations* (c'est la « triade traditionnelle »), mais aussi de divers objets (la « Pyramide du Louvre », « Les Misérable » par exemple). Ce sont donc des « noms *propres* », au sens où ils désignent une entité de façon univoque (le *Rhin* désigne une entité immédiatement identifiable, contrairement à *fleuve*).

Nous incluons également dans les entités nommées, comme on le fait traditionnellement (Nouvel, Ehrmann et Rosset 2015) les nombres et les dates.

La ressource que nous proposons d'élaborer aura plusieurs fonctions. D'abord elle permettra une tokenisation correcte du texte. Les entités nommées sont en effet un ensemble de « mots » (au sens d'un ensemble de lettres limité par des espaces ou des ponctuations) qu'il faut considérer comme une seule unité : par exemple, *Organisation des Nations-Unies* est un ensemble de « mots » qui ne doit former qu'un seul token, puisqu'il s'agit d'une entité nommée.

Cette tokenisation est indissociable de la tâche de la *détection des entités nommées*, bien étudiée dans le domaine du TAL (Nadeau et Sekine 2007 ; Nouvel, Ehrmann et Rosset 2015). On peut distinguer deux types d'approches. Les méthodes par apprentissage automatique utilisent des algorithmes statistiques[1] alors que les méthodes à base de règles utilisent des patrons prédéfinis : un nombre compris entre 1 et 31 suivi d'un nom de mois suivi d'un nombre de quatre chiffres sera ainsi reconnu comme une date. C'est cette dernière approche qui est retenue par Longo (2013, p. 267), dont le programme utilise « un système de règles symboliques, des listes de mots et des indices de surface ».

L'inconvénient de ces techniques, c'est qu'elles ne donnent aucune information sur les entités nommées, mise à part leur type (personne, organisation, lieu, etc.). Longo, par exemple, utilise des règles et des listes de prénoms (des « preuves internes ») pour identifier les noms de personnes : *Jacques Chirac* sera identifié comme une entité de type « personne » parce que le système identifiera le token *Jacques* comme un prénom suivi d'un nom en majuscule. Mais cela ne donne aucune indication, par exemple, sur le sexe de la personne (l'entité peut-elle être reprise par *il* ou bien par *elle* ?) ni sur ces possibles reprises anaphoriques (Chirac est-il un *président*, un *astronaute*, un *linguiste* ?). La ressource qu'il nous faut doit contenir toutes ces informations.

Nous venons d'évoquer le sexe de la personne, mais il doit aussi être possible de trouver le genre des organisations (l'*Organisation des Nations-Unies* sera reprise par *elle* alors que le *Real Madrid* sera repris par *il*) ou d'autres entités (le *Rhin vs.* la *Seine*).

Par ailleurs, il ne suffit pas de reconnaître les désignations d'entités nommées dans un texte, il faut encore pouvoir relier différentes désignations qui renvoient à la même entité, comme *Paris* et *Ville lumière* ou *ONU*, *Organisation des Nations-Unies* et *Nations-Unies*.

1. Nouvel, Ehrmann et Rosset (2015, p. 91-98) pour une description de quelques algorithmes et de leur fonctionnement.

Enfin, la ressource doit permettre d'identifier les différentes reprises possibles pour une même entité. Chirac, par exemple, pourra être repris par *président*, *ministre*, *maire*, etc. mais pas par *chimiste* ou *médecin*. Il est plus difficile de prévoir les reprises pour une personne qui n'est pas connue ou pour un personnage de fiction, c'est pourquoi il faut prévoir une liste de noms d'humains qui peut servir de liste de reprises anaphoriques possible pour une personne sur lesquelle on ne dispose pas d'information. Ce qui vaut pour les personnes vaut aussi pour les objets : le *Rhin*, par exemple, pourra être repris par *fleuve*, *cours d'eau*, etc., mais pas par *mer*, *montagne*, *journaliste* ou encore *meuble*.

Pour résumer, la ressource que nous cherchons doit pouvoir *identifier* les entités nommées, donner des *informations lexicales* (reprises par différents noms propres, comme *ONU vs. Organisation des Nations-Unis*), des *informations grammaticales* (reprises pronominales), des *informations sémantiques* (reprises nominales).

4.1.2 Ressources disponibles

Détection des entités nommées

Il existe de nombreux outils de détection d'entités nommées, mais ces outils ne font que *détecter*, ils n'ajoutent aucune connaissance, mis à part le type de l'entité (personne, organisation, lieu, nombre, date, etc.). C'est le cas par exemple de *JRC Names*[2], une ressources qui contient environ 250 000 entités. La particularité de cette ressource est qu'elle est multilingue et couvre les désignations d'une même entité selon des orthographes en plusieurs langues. Ainsi, le prénom de *Benjamin Netanyahu* sera reconnu aussi par *Binyamin, Bibi, Benyamín, Biniamin* ainsi que leur équivalents russes, arabes, etc. De même pour le nom de famille. Les désignations ont été agrégées à partir d'autres ressources, comme *Wikipedia*, *DBpedia* ou *Talk-Of-Europe*. *JRC Names* souffrent pourtant d'importants défauts : il ne reconnaît que les personnes et les organisations (pas les lieux, le nombres, les dates, les titres d'oeuvre, le groupes de musique, etc.) ; de simples tests montrent qu'il reconnaît « François Hollande », mais pas « Hollande » tout seul, ni « M. Hollande ». Par ailleurs, notre programme ne concerne que le français, il nous est donc peu utile d'avoir les désignations des entités en différentes langues.

CasEN[3] (Friburger et Maurel 2004 ; Maurel, Friburger, Antoine, Eshkol et Nouvel 2011) est un autre outil qui permet de détecter les entités nommées. Il fonc-

2. *European Commission's Joint Research Centre (JRC)* : https ://ec.europa.eu/jrc/en/language-technologies/jrc-names.

3. http ://tln.li.univ-tours.fr/Tln_CasEN.html

tionne uniquement à partir de règles appliquées en cascade (d'où son nom). Il détecte beaucoup de types d'entités, contrairement à *JRC Names* mais il est difficilement intégrable dans notre programme, puisqu'il ne fonctionne qu'au travers de la plateforme de traitement linguistique *Unitex*[4] : le format de sortie est peu commode et oblige à de multiples manipulations pour retrouver, dans le texte d'origine[5], la position exacte des entités nommées.

Un autre outil, *mXS*, présenté par Nouvel, Antoine, Friburger et Soulet (2011) permet l'identification des entités nommées. Mais il fait usage d'un étiqueteur morpho-syntaxique, *TreeTagger* (Schmid 1994, 1995), si bien que notre texte aurait été annoté deux fois : une première fois par *TreeTagger* pour satisfaire *mXS* et une autre fois par *Talismane*, que nous présentons au chapitre 5 : c'est une perte d'efficacité.

Ressources lexicales

Les limites de ces outils nous ont conduit à proposer notre propre outil de détection des entités nommées. Pleinement intégré à notre programme de détection des chaînes de référence, il apporte également, pour chaque entité nommée des informations grammaticales, lexicales et sémantiques. Pour ce faire, il nous a fallu élaborer une ressource qui contiennent toutes les informations décrites ci-dessus. Dans les limites de ce travail de Master, il nous a paru plus pertinent d'adapter une ressource existante plutôt que d'en construire une toute nouvelle.

La première étape a consisté à étudier les ressources disponibles. La première ressource à considérer est *BabelNet*[6], qui est un réseau sémantique et une ontologie, c'est-à-dire que pour, chaque concept, il donne un ensemble de concepts sémantiques reliés, que ces concepts soient dans une relation hiérarchique (de type *chat–animal*) ou non (de type *François Hollande–langue française*). *BabelNet* contient aussi nombre de propriétés pour chaque entrée, par exemple, pour François Hollande, il donne ses fonctions, ses lieu et date de naissance, etc. Il a été créé en associant *Wikipdia* et *WordNet* (Princeton University p.d.). Le défaut majeur de *BabelNet* est sa disponibilité : il n'est disponible que *via* une *API*[7], c'est-à-dire qu'on ne peut pas directement le télécharger et le

4. Logiciel disponible à l'adresse `http ://unitexgramlab.org/`.

5. Le principal problème, c'est qu'*Unitex* applique un prétraitement au texte d'origine, et donc le modifie.

6. `http ://babelnet.org`

7. Acronyme de *Application Programming Interface*. Il s'agit d'un ensemble de fonctions utilisables par d'autres programmes. Par exemple, on peut construire un programme qui demande à telle fonction (il faut lire la documentation pour savoir exactement laquelle) de l'API quelles sont les hyperonymes

modifier : il faut l'interroger à distance, en étant connecté à Internet. Mais pour cela, il faut avoir un compte, et le nombre de requêtes par jour est limité. Or notre programme doit être conçu pour être utilisable par tous et sur une grande quantité de textes, pas uniquement par ceux qui ont un compte *BabelNet* et pour un petit nombre de textes. De plus, nous programmons en *Python*, et l'*API* n'est disponible qu'en *http* ou *Java*.

Deux autres ressources sont extraites de *Wikipedia* : *Wikidata*[8], qui structure les informations de *Wikipedia* de façon relationnelle (par exemple, *François Hollande* a pour *nature* la propriété *être humain*, son *sexe* est *masculin*, etc.). Néanmoins, comme *BabelNet* (mais sans qu'il soit besoin d'avoir un compte, et sans limite de connexion), il ne peut être interrogé qu'à distance (*via* une *API http*). La deuxième ressource est *DBPedia*[9], qui pose le même problème d'accès uniquement à distance, en plus d'une interface peu intuitive.

C'est finalement *Yago3*[10] (Mahdisoltani, Biega et Suchanek 2014) que nous avons choisi. En effet, toutes les données peuvent être téléchargées sous un format texte très facile à utiliser (voir section 4.1.3). De plus, sa licence est très permissive, puisqu'il est possible de modifier et de redistribuer la ressource. Il est extrait de *Wikipedia*, *WordNet*, que nous avons déjà évoqués, ainsi que de *GeoNames*[11], une ressource qui donne des indications géographiques et lexicales (les différentes désignations, dans différentes langues, d'un même lieu, par exemple pour *Strasbourg* : *Argentoratum*[12], *Estrasborg, Estrasbou*, etc. Enfin, il contient toutes les informations dont nous avons besoin.

Mais comme il contient aussi beaucoup plus d'informations que celles dont nous avons besoin, il nous a fallu adapter cette ressource, c'est-à-dire en extraire les informations pertinentes et les réécrire sous une forme plus compacte et lisible par notre programme.

4.1.3 Élaboration d'une ressource adaptée

Extraction d'une ressource lexicale à partir de Yago

de « pomme », et la fonction donnera la réponse. Mais il n'est pas possible de savoir ce qui se cache « derrière » la fonction, ni en ce qui concerne le code informatique, ni en ce qui concerne les données sémantiques.

8. https ://www.wikidata.org
9. http ://wiki.dbpedia.org
10. https ://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
11. http ://www.geonames.org
12. Le nom du camp romain sur lequel a été construite l'actuelle ville.

Description de Yago *Yago* est un ensemble de près de 1,2 *milliards* de triplets, réparti dans 108 fichiers totalisant 91 Go de donnnées (soit 91 milliards de caractères). Ces fichiers sont facilement lisibles, puisqu'il sont en texte brut.

Chaque triplet est constitué d'un « sujet » (par exemple « Nirvana »), qui représente l'entité à laquelle on attribue le prédicat ; le « prédicat », qui indique le type de propriété attribuée à l'entité (par exemple « a créé ») ; « l'objet », qui est la valeur de la propriété (par exemple « Smells Like Teen Spirit »[13]). Le triplet est donc représentatif de la relation « Nirvana a créé *Smells Like Teen Spirit* ». Voici quelques exemples :

```
<Nirvana_(band)>  <created>  <Smells_Like_Teen_Spirit>
<fr/Emmanuel_Macron> <wasBornIn> <Amiens>
<Ansel_Adams>     <hasGender> <male>
```

Les triplets suivants listent les différentes désignations *françaises* (d'où le `@fra` à la fin) du festival de Cannes :

```
<Cannes_Film_Festival> <redirectedFrom> "Festival du film de Cannes"
   ↪ @fra
<Cannes_Film_Festival> <redirectedFrom> "Festival de cannes"@fra
<Cannes_Film_Festival> <redirectedFrom> "Festival international du film
   ↪ "@fra
<Cannes_Film_Festival> <redirectedFrom> "Festival international du film
   ↪  de Cannes"@fra
<Cannes_Film_Festival> rdfs:label  "Festival de Cannes"@fra
```

Difficultés Comprendre les relations n'est pas difficile : leur noms (`created`, `wasBornIn`, `hasGender`, `redirectedFrom`, `refs :label` dans les exemples précédents) sont transparents. Par contre, c'est leur nombre qui est déroutant : près de 1,2 milliards. Ces relations sont réparties dans une centaine de fichiers, distribués par thème et listés sur le site Internet de Yago, mais la documentation est rudimentaire et il nous a fallu observer les données pour comprendre comment Yago était organisé dans le détail. C'est donc la localisation des informations dont nous avions besoin qui nous a demandé le plus de temps.

Un autre problème est survenu avec le nombre d'entités nommées retenues (près de 6,4 millions de désignations différentes pour plus de 2,3 millions d'entités). Charger

13. Une célèbre chanson du groupe.

toutes ces données (avec toutes les informations lexicales et sémantiques) dans la mémoire de l'ordinateur n'était pas possible, il nous a donc fallu établir un système de base de données, qui charge uniquement les identifiants des entités nommées et accède rapidement par la suite aux autres données.

Méthodologie La première étape a consisté à extraire les désignations (*labels* en anglais). Celles-ci peuvent être la désignation propre à l'entité, ou bien extraite de pages de redirection de Wikipédia (lorsqu'on entre « ONU » dans la zone de recherche de Wikipédia, on aboutit sur la page « Organisation des Nations unies », avec l'indication *Redirigé depuis ONU* ; parfois il ne s'agit que d'une différence d'accent : *Karate Kid* redirige vers *Karaté Kid (film, 1984)*).

Mais il y a près de 45 millions de désignations, qui concernent une dizaine de langues. Toutes ces désignations ne nous sont pas utiles, d'abord parce qu'on risque fort peu de trouver un nom en hindi dans un texte en français (qui est la langue sur laquelle nous travaillons), et ensuite parce que toutes les entrées de Wikipédia n'existent pas dans toute les langues (la page de la Faculté de médecine de Strasbourg existe en français, mais pas dans les autres langues) : or nous avons surtout besoin des entités présente dans le Wikipédia français[14]. La difficulté vient donc de la sélection les désignations en français. Pour un certain nombre d'entre elles, un marqueur de langue est apposé (`@fra`), mais ce n'est pas le cas de toutes les désignations, même celles qui sont françaises[15]. Quand il n'y avait pas de marqueur (ni français, ni étranger), nous avons donc analysé les caractères de la désignation, en recherchant leur *script* Unicode[16] : nous avons gardé le script latin.

Nous avons ensuite créé des désignations supplémentaires pour les personnes. Pour chaque entités « personne », nous avons sélectionné les étiquettes existantes (par exemple *François Hollande*), puis nous avons cherché le prénom (exprimé avec la relation `<hasGivenName>`, donc ici *François*) et le nom de famille (exprimé avec la relation `<hasFamilyName>`, donc ici *Hollande*). Nous avons laissé le nom de famille comme une désignation à part entière. Puis nous avons calculé les initiales des prénoms (*F.*), que nous avons incluses comme désignation avec le nom de famille. Nous obtenons donc les désignations suivantes pour l'ancien président de la République : *François Hollande,*

14. Nous pourrions inclure d'autres langues, mais, afin que la ressource reste maniable, nous devons limiter le nombre d'entité nommées retenues.

15. Nous n'avons pas réussi à comprendre quand il y a avait un marqueur et quand il n'y en avait pas.

16. « Script » est un terme technique pour désigner, dans Unicode, un ensemble de caractères appartenant à un même système d'écriture.

Hollande, F. Hollande, ce qui correspondent aux différentes formes que l'on trouve dans les textes. Pour les prénoms composés, nous envisageons même plus de possibilités, avec ou sans trait d'union (*J. P.* et *J.-P.*) afin de couvrir un maxium de cas.

Nous avons également dû « nettoyer » certaines désignations. Par exemple, pour différencier entre des œuvres qui ont le même titre, Wikipédia ajoute une indication entre parenthèse (par exemple *Harry Potter (personnage), Harry Potter (films)*, ou bien *Karaté Kid (film, 1984), Karaté Kid (série de films)*, ou même pour différencier entre deux personnes : *Laurent Pernot (helléniste*[17]*), Laurent Pernot (artiste)*). Il faut bien sûr supprimer la parenthèse.

La deuxième étape à consister à construire l'ontologie. Nous avons dû faire de nombreux essais pour savoir quel fichier utiliser et comment les relations fonctionnent, parce qu'il n'y a aucune documentation sur le site Internet de Yago. Puisque toutes les relations de Yago sont des triplets (de type *A est une sous-classe de B*), il faut construire l'ontologie par transitivité : si A est une sous-classe de B et B une sous-classe de C, alors on peut établir la hiéarchie A > B > C. Néanmoins, les catégories de Yago peuvent être, entre autres, celles de Wikipédia et celle de WordNet. Ce sont celles de Wordnet qui nous intéressent surtout, car elles sont traduites en plusieurs langues (dont le français) : pour chaque catégorie il y a une étiquette, souvent plusieurs (parfois aussi aucune), par exemple :

```
<wordnet_football_player_110101634> rdfs:label "footballeur"@fra
<wordnet_football_player_110101634> rdfs:label "footballeuse"@fra
<wordnet_lake_114991004> rdfs:label "lac"@fra
<wordnet_lake_109328904> rdfs:label "loch"@fra
```

Noter que *lac* et *loch* ne sont pas rattaché à la même catégorie (les identifiants diffèrent par leur numéro), ce qui est normal puisqu'ils désignent deux réalités différentes. On remarque aussi que la catégorie `<wordnet_football_player_110101634>` a une traduction au masculin et au féminin, sans qu'on sache pourquoi (les autres catégories ont plutôt un lemme au masculin). Il s'agit sans doute d'un artifact dû à une traduction automatique.

Puisque Zinédine Zidane appartient à la catégorie `<wordnet_football_player_110101634>`, on peut facilement retrouver le fait que c'est un *footballeur*.

Mais les catégories de Wikipédia nous intéressent aussi, car bien qu'elles ne soient pas traduites en français (il y a en a plus de 8 000, une traduction manuelle est donc

17. Et professeur à la Faculté des Lettres de l'Université de Strasbourg.

difficilement envisageable), elles sont systématique reliées à une catégorie WordNet, qui, elle, est traduite.

On obtient donc une ontologie de ce type (seul les cinq premiers niveaux sont affichés) :

```
<wordnet_physical_entity_100001930> (entite physique)
  <wordnet_process_100029677> (processus physique, processus)
    <wordnet_irreversible_process_113503226>
    <wordnet_variation_113572324>
      <wordnet_covariation_106032752>
    <wordnet_dealignment_113455906>
    <wordnet_iteration_113503908>
    <wordnet_economic_process_113471206>
      <wordnet_market_forces_113511243>
      <wordnet_economic_growth_113471052> (croissance économique)
      <wordnet_globalization_113488110> (mondialisation,
        ↪ globalisation)
      <wordnet_demand_113461162> (demande (offre et demande), demande
        ↪ )
        <wordnet_consumption_113451804> (consommation)
      <wordnet_disinflation_113466849>
      <wordnet_inflation_113498828> (gonflage, inflation)
        <wordnet_cost-pull_inflation_113453737>
        <wordnet_reflation_113548350> (relance économique,
          ↪ réactivation)
        <wordnet_stagflation_113560738> (stagflation)
        <wordnet_demand-pull_inflation_113461390>
```

Il y a plus de 76 000 lignes de ce type, en incluant les catégories Wikipédia, et cela juste pour les cinq premiers niveaux. On remarque les traduction des catégories entre parenthèses, pas toujours exactes. Un système permet de corriger et supprimer manuellement les traductions données par Yago et d'en ajouter d'autres[18].

Chaque entité a donc sa hiérarchie ; par exemple, pour Zinédine Zidane :

```
<wordnet_football_player_110101634> (footballeur, footballeuse)
  <wordnet_athlete_109820263> (sportif, athlète)
    <wordnet_contestant_109613191> (challengeur, participant,
      ↪ concurrent, challenger, compétiteur, rival)
```

18. Nous avons entrepris ce travail, mais n'avons pas fini, et de loin.

```
        <wordnet_person_100007846> (quelqu'un, homme, un)
          <wordnet_causal_agent_100007347>
            <wordnet_physical_entity_100001930> (entite physique)
          <wordnet_organism_100004475> (créature, organisme)
            <wordnet_living_thing_100004258>
              <wordnet_whole_100003553> (tout)
                <wordnet_object_100002684> (objet, chose)
                  <wordnet_physical_entity_100001930> (entite
                    ↪ physique)
  <wordnet_player_110439851> (joueur, participant)
    <wordnet_contestant_109613191> (challengeur, participant,
      ↪ concurrent, challenger, compétiteur, rival)
      <wordnet_person_100007846> (quelqu'un, homme, un)
        <wordnet_causal_agent_100007347>
          <wordnet_physical_entity_100001930> (entite physique)
        <wordnet_organism_100004475> (créature, organisme)
          <wordnet_living_thing_100004258>
            <wordnet_whole_100003553> (tout)
              <wordnet_object_100002684> (objet, chose)
                <wordnet_physical_entity_100001930> (entite
                  ↪ physique)
```

ou pour l'*Énéide* :

```
<wordnet_book_106410904> (volume)
  <wordnet_publication_106589574> (publication)
    <wordnet_work_104599396> (ouvrage)
      <wordnet_product_104007894> (biens)
        <wordnet_creation_103129123>
          <wordnet_artifact_100021939> (objet fabriqué, objet
            ↪ façonné, artéfact, produit oeuvré, artefact, engin)
            <wordnet_whole_100003553> (tout)
              <wordnet_object_100002684> (objet, chose)
                <wordnet_physical_entity_100001930> (entite
                  ↪ physique)
```

Néanmoins, on voit bien que la hiérarchie remonte trop haut : il faudrait la couper lorsqu'un atteint une « catégorie principale » (`<wordnet_person_100007846>` pour Zidane, `<wordnet_work_104599396>` pour l'*Énéide*). Des options permettent de mar-

quer manuellement certaines catégories comme « catégories principales »[19] : elles deviennent le type de l'entité nommée (par exemple « personne » pour la catégorie `<wordnet_person_100007846>`, « lieu », etc.), et en plus arrête aussi la hiéararchie à ce niveau : `<wordnet_living_thing_100004258>` et `<wordnet_organism_100004475>`, par exemple, sont alors exclus de la hiérarchie. Ainsi Zinédine Zidane, déjà cité, est non seulement un `<wordnet_football_player_110101634>` mais aussi, au plus haut dans la hiérarchie, un `<wordnet_person_100007846>`, et son type d'entité sera « personne ». Ses catégories seront donc :

```
<wordnet_football_player_110101634> (footballeur, footballeuse)
<wordnet_contestant_109613191> (challengeur, participant, concurrent,
   ↪ challenger, compétiteur, rival)
<wordnet_athlete_109820263> (sportif, athlète)
<wordnet_person_100007846> (quelqu'un, homme, un)
<wordnet_player_110439851> (joueur, participant)
```

Avec ce principe, il est possible de trouver toutes les désignations communes (ou « hyperonymes ») pour chaque entité, en récupérant toutes les traductions des catégories *WordNet* à chaque niveau de la hiéarchie jusqu'à la catégorie principale. Par exemple, pour Zidane, on aura : *homme, athlète, sportif, footballeur, quelqu'un, participant, rival, concurrent, challenger, compétiteur, un, joueur, footballeuse*. On notera la présence de « un », dû à la catégorie `<wordnet_person_100007846>` : il s'agit là aussi, probablement, d'un artifact de traduction automatique[20] (comme pour « footballeuse »).

Nous avons cependant rencontré des problèmes lorsque des entités appartiennent à plusieurs catégories (*WordNet* s'avère être un « buisson » plutôt qu'un « arbre ») : laquelle faut-il alors choisir ? Il s'agit là d'un problème que nous n'avons pas eu le temps d'analyser en profondeur : pour l'heure le programme sélectionne inconditionnellement la première (ce qui revient à dire que le choix est aléatoire).

Une étape supplémentaire consiste à trouver le genre (puis le nombre) pour les entités nommées (ce qui donne une indication sur le type de reprise pronominale possible). Pour les personnes, il existe une relation `<hasGender>` ; pour les autres[21], il faut décomposer les désignations, chercher le nom tête, et trouver le genre dans un dictionnaire de formes

19. Là encore, nous sommes loin d'avoir achevé la détermination de toutes les catégories principales. Notre principal problème est d'en établir une liste.

20. C'est pourquoi il faut vérifier les traductions à la main. Nous avons prévu un système qui permet d'intégrer les corrections dans notre programme, mais n'avons pas encore fait toutes les vérifications elles-mêmes.

21. Cela n'a pas été implémenté dans la version actuelle. Néanmoins, on peut trouver le genre grâce à l'article, sauf quand il est élidé. Le nombre non plus n'est pour l'instant pas recherché. Mais cela

(et pas seulement de lemmes) : pour *Organisation des Nations-Unis*, il faut chercher le genre de *Organisation*.

La dernière étape consisterait[22] a extraire des informations des infoboxes, qui sont les « boîtes d'informations » que l'on trouve de plus en plus souvent sur les pages Wikipédia (en haut à droite de l'article) : pour les personnes, elles indiquent par exemple la date de naissance (et de mort, le cas échéant), les fonctions, etc., pour les entités d'autres informations, comme les coordonnées géographiques, etc. Ce qui nous intéresse surtout, ce sont les fonctions occupées par les humains. En effet, ce peut être des « fonctions » (*Président de la République française, Président du conseil général de Corrèze, Premier secrétaire du Parti socialiste*, etc. pour François Hollande), ou des « professions » (*professeur, historien* pour Laurent Pernot (mais on notera qu'il n'est pas fait mention de son élection à l'Académie française)), ou encore des « distinctions » (*Officier de l'Ordre de l'Empire britannique* pour Alan Turing), etc. Ces informations permettent donc de trouver des désignations communes (« hyperonymes ») pour chaque entité de type « personne ».

Plusieurs problèmes en perspective, cependant : d'abord, il faudrait établir la liste des attributs (de type `infobox/fr/profession`, `infobox/fr/fonction`). Ensuite, il n'y a pas forcément cohérence dans les identifiants d'entités nommées (ainsi, pour les infoboxes, Zinédine Zidane a pour identifivant `<Zinédine_Zidane>`, et non `<Zinedine_Zidane>` comme dans les autres parties de Yago. Par ailleurs, les termes doivent être « nettoyés » du wikicode qu'ils contiennent. Par exemple, l'une des fonction de François Hollande est décrite comme `Président du [[conseil général de la Corrèze]]`, il faut donc supprimer le wikicode (ici les double crochets)

Détection des entités nommées

Nous aborderons dans cette section la façon dont on détecte les entités nommées. C'est le module de tokenisation qui s'occupe de cette tâche.

La première étape utilise une liste de 6,4 millions de désignations différentes, extraites de Yago (il est cependant possible d'ajouter des entités nommées supplémentaires, en fonction du texte ou des besoins de l'utilisateur). Ces désignations différentes, segmentées en tokens, est comparée aux tokens présents dans le texte, par une simple recherche. Par exemple, `F . Hollande` est composé de trois tokens (un F, un point

pose des difficultés supplémentaires : les Beattles est à la fois pluriel (il y a plusieurs chanteurs) et singulier (c'est *un* groupe de musique).

22. Nous remettons ce travail à un temps ultérieure.

et un mot) : si cette désignation est présente dans le texte, alors ces trois tokens seront réunis pour former une entité. La recherche commence par les entités qui ont le nombre le plus élevé de tokens. À ce niveau, les informations lexicales, comme le type de l'entité, et grammaticales, comme le genre et le nombre, sont chargées en mémoire.

Les étapes suivantes utilisent un ensemble de règles, notamment :

- l'inclusion du déterminant : si l'entité est précédée d'un déterminant (*l'ONU*), celui-ci est inclu dans son token de l'entité ; cela permet de faire un véritable nom propre et de ne pas troubler l'analyseur syntaxique,
- l'utilisation du déterminant pour définir le genre et le nombre (*le Rhin* sera masculin, tout comme *le Real Madrid*, mais *la Seine* sera féminin),
- la désambiguïsation de différents types d'entités avec le déterminant[23]. En effet, certains types d'entités nommées ne prennent jamais de déterminant, d'autres souvent. C'est le cas de *Hollande* : lorsque le terme est précédée d'un déterminant ou de certaines prépositions, il s'agit du pays (*la Hollande*, *en Hollande*). Sinon, il s'agit de la personne (*Hollande était président de la République*),
- les titres de civilité (*M., Mme*) et les grades (*Dr., Prof.*) sont supprimés. On trouve souvent ces titres dans les articles de journaux (*M. Macron*), même pour parler de personnes connues. Ces désignations n'apparaissent bien sûr pas dans les listes extraites de Yago, il faut donc les traiter à part. Certains titres permettent de donner un genre à l'entité, s'il n'est pas déjà connu (*M., Mme*),
- les successions de lettres majuscules (suivies ou non par un point), qui ne sont pas présentent dans les listes utilisées à la première étape, sont considérées comme des noms d'organisation (*SNCF*), sauf si elles sont suivies d'un nom de famille ou d'une entité représentant une personne : ce sont alors des initiales de prénoms (*J. R. R. Tolkien*). Tous ces tokens sont fusionnés,
- les tokens qui apparaissent sur des listes de prénoms ou de noms de familles, et qui n'ont pas encore d'attribution, sont considérés comme des noms de personne : s'il y a un prénom, il est possible de récupérer de genre (*Marc, Marie*, sauf dans quelques cas de prénoms épicène : *Dominique*). Il faut cependant faire attention : ce n'est pas parce qu'un mot commence par une majuscule que c'est forcément un nom propre. Le cas de « Pierre » est fréquent : est-ce un prénom ou un nom

23. Cela n'est pas encore implémenté dans le programme, car les types d'entités qui demandent un déterminant doit être établis à la main.

commun ? Le système vérifie s'il est précédé d'un déterminant ou d'un adjectif avant de prendre une décision[24],

- les mots en majuscules restants sont considérés comme entités nommées de type personne[25].

Les désignations d'entités nommées issues des listes préétablies contiennent toutes un identifiant (le « lemme » de l'entité nommée, si l'on veut) : ainsi *François Hollande*, *F. Hollande* et *Hollande* seront associés. Mais cela n'est pas le cas pour les entités nommées trouvées par des règles : si un fait divers, par exemple, évoque d'abord *Pierre Dupont* pour le reprendre ensuite sous la forme *M. Dupont*, une étape supplémentaire est requise pour trouver qu'il s'agit de la même personne. Cette étape n'est réalisée que pour les entités de type « personne » et sur la base du dernier élément (nom de famille). Notre programme ne pourra donc pas associer ces deux désignations à *Pierre*, par exemple.

Informations grammaticales, lexicales et sémantiques

Les informations grammaticales et lexicales sont chargées en mémoire lors de la tokenisation, les informations sémantiques (hyperonymes) plus tard, au moment de la résolution de la coréférence. Il ne s'agit cependant pas ici des mêmes données : pour la tokénisation, ce sont des *désignations* qui sont chargées et cherchées (plusieurs désignations peuvent se rapporter à la même entité) : chaque désignation se voit attribuer l'identifiant de l'entité nommée. Les informations sémantiques, elles, sont chargées en fonction de l'*identifiant* : il n'est plus question ici de désignations.

Lorsqu'aucune information sémantique n'est trouvée dans les dictionnaires, des hyperonymes par défaut sont associées aux entités de type « personne ». La liste a été établie par le projet NHUMA (*NHUMA : Les noms d'humains : de la description linguistique aux applications lexicographiques* p.d.).

Nous avons vu, dans la section précédente, que certaines de ces informations étaient utilisées pour la tokenisation (un titre de civilité suivi d'une entité de type « personne » (telle est l'information lexicale) permettra de fusionner les deux tokens), et que d'autres

24. Il y a toujours des exceptions : *le petit Pierre* est un nom propre, même s'il est précédé d'un adjectif.

25. Nous les considérons comme personnes plutôt que comme organisations, parce que cela nous semble être le cas le plus fréquent ; mais cela peut être un biais de notre corpus d'étude (des articles du *Monde*, essentiellement).

sont au contraire obtenues grâce à une analyse des tokens (le genre déterminé par le déterminant, par exemple).

Ces informations sont également utilisées pour la résolution des relations anaphoriques (par opposition à la coréférence). Par exemple, le genre et le nombre permettent de définir si telle entité peut être l'antécédent de tel pronom (masculin *vs.* féminin, singulier *vs.* pluriel). De même, le type de l'entité joue un rôle similaire pour les pronoms de première et deuxième personne, qui ne peuvent[26] être associés qu'à des entités de type « personne ».

Lors de la résolution de la coréférence (par opposition aux relations anaphoriques), ce sont les informations sémantiques, c'est-à-dire les hyperonymes et désignations communes (*le Rhin... Ce fleuve*), qui sont surtout utilisées, bien que le type de l'entité serve aussi à n'associer des noms d'humains qu'à des entités de type « personne ».

4.1.4 Bilan et perspectives

Nous avons achevé l'ensemble des aspects automatiques, mais nous n'avons pas terminé les traitements manuels (détermination des catégories principales, vérification et correction des traductions des catégories *WordNet*, énumération des catégories d'entités qui requiert un article, genre et nombre des entités non humaines, etc.). Notre programme se veut un prototype : notre travail est donc, à ce stade du moins, suffisant pour qu'il soit fonctionnel.

Nous n'avons pas non plus inclus les propriétés des infoboxes, parce que cela requiert un travail d'observation plus approfondi : il n'y a pas de règles strictes qui président à la rédaction des infoboxes, et les propriétés peuvent donc varier, dans une certaine mesure, d'une entité à une autre.

Plus tard, il serait également possible de compléter les informations de Yago par celle du Glawi, dont nous parlerons ci-dessous. Le Glawi, en effet contient de nombreux noms propres spécifiquement français. Mais il faudrait d'abord vérifier s'il contient plus d'informations que ce qui est déjà contenu dans Yago.

Enfin, nous avons extrait des listes de prénoms et de noms de famille, mais nous avons préféré utiliser celles que nous avons obtenues à partir du Glawi (voir ci-dessous), car elles sont plus petites et plus « maniables ».

26. Sauf dans certaines fables, mais il est possible de désactiver cette option.

Il serait aussi possible d'extraire des noms d'humains à partir des infoboxes, mais pour l'heure nous utilisons la liste établie par le projet NHUMA (*NHUMA : Les noms d'humains : de la description linguistique aux applications lexicographiques* p.d.).

4.2 Dictionnaires

4.2.1 Motivation

L'utilisation de ressources lexicales est indispensable pour résoudre les anaphores infidèles, du type *un chat... cet animal...*. En effet, aucune règle ne permet de les résoudres sans un *dictionnaire d'hyperonymes et synonymes* (les hyponymes ne sont pas nécessaires, puisqu'un hyponyme a forcément un hyperonyme, et que la relation est donc incluse dans le dictionnaire d'hyperonymes).

Mais nous avons également besoin d'autres ressources lexicales. D'abord, une liste de mots composés pour la tokénisation, notamment pour les préposition, les conjonctions et les adverbes (*parce que, au coup par coup, à peine*). Ensuite, une liste de termes non référentiels (*jouer des coudes, se la jouer*). Enfin, des listes de prénoms et noms qui permettront de reconnaîtres les entités nommées de type « personne » qui ne sont pas dans Yago.

4.2.2 Ressources disponibles

Nous recherchons donc une ressource adaptable qui contienne une liste d'hyperonymes et de synonymes, mais aussi toutes les formes pour chaque lemme, ainsi que des informations grammaticales. De plus, elle doit être libre de droits.

Morphalou (Salmon-Alt, Romary et Pierrel 2005) contient toutes les informations morphologiques et grammaticales nécessaires, mais aucune informations sur les relations sémantiques. De plus, elle n'est pas tout à fait libre de droits pour l'usage que nous souhaitons en faire.

Le *Lefff* (Sagot 2010) est une ressource construite semi-automatiquement. Il contient des formes et leurs informations grammaticales, mais seulement des mots simples. De plus, il ne contient aucune information sémantique.

La troisième grande ressource à citer est *Wolf*, ou *WOrdnet Libre du Français* (Sagot et Fišer 2008). Il s'agit d'une traduction semi-automatique en français du *WordNet* de Princeton (qui est en anglais). La principale faiblesse de *Wolf* est qu'il reprend intégralement la structure hiérarchique de Wordnet, même lorsque cette structure ne correspond pas au vocabulaire français. Il y a donc des éléments vides en français, et des niveaux hiérarchiques français qui n'apparaissent pas, car ils n'ont pas d'équivalent anglais. Par ailleurs, notre exemple préféré, *chat*, n'a pas *animal* comme hyperonyme ; même s'il ne s'agit là que d'un seul exemple, il montre que la ressource a des défauts même là où le français correspond parfaitement à l'anglais (*cat–animal*). Enfin, il ne contient pas de formes (seulement des lemmes), et aucune information grammaticale.

Reste le *Glawi* (Hathout et Sajous 2016 ; Sajous et Hathout 2015). Il s'agit d'une mise au format XML du *Wiktionnaire*, le dictionnaire collaboratif libre construit sur le modèle de l'encyclopédie Wikipédia. Il contient toutes les informations dont on a besoin : aussi bien des synonymes que des hyperonymes (et même des hyponymes), mais aussi toutes les déclinaisons et conjugaisons, et les informations grammaticales pour chaque forme. Notre choix se porte donc sur cette dernière ressource.

4.2.3 Glawi

Présentation du Glawi

Le Glawi se présente comme un gros fichier XML. Il existe deux version :

- une version simple (le fichier fait 1,5 Go),
- une version dans laquelle toutes les données textuelles (étymologie, définitions, exemples) ont une analyse syntaxique effectué par Talismane (Urieli 2013) (le fichier fait 3,4 Go).

Voici le début de l'entrée *vase*, avec des commentaires dans le code XML (entre les balises `<!--` et `-->`) :

```
<!-- noter ici le titre de l'entrée -->
<title>vase</title>
<pageId>105128</pageId>
<meta>
  <!-- les catégories Wikipédia, qui s'appliquent à toutes l'entrée,
     ↪ quelle que soit la définition -->
  <category>Noms multigenres en français</category>
```

```
  <reference>DAF8</reference>
</meta>
<text>
  <pronunciations>
    <pron area="France Paris">vaz</pron>
    <pron>vaz</pron>
  </pronunciations>
  <etymology>
    <etym>
      <xml><i>(<innerLink ref="#fr-nom-1">nom commun 1</innerLink>)</i><
         ↪ century>fin du xv</century> du moyen néerlandais <
         ↪ foreignWord lang="dum">wase</foreignWord> qui donne le
         ↪ néerlandais <i><innerLink ref="waas">waas</innerLink></i>,
         ↪ apparenté à <i><innerLink ref="gazon">gazon</innerLink></i>,
         ↪  en ancien français <i>wason</i>.</xml>
      <txt>(fin du xve siècle) du moyen néerlandais wase qui donne le
         ↪ néerlandais waas, apparenté à gazon, en ancien français
         ↪ wason.</txt>
    </etym>
    <etym>
      <xml><i>(<innerLink ref="#fr-nom-2">nom commun 2</innerLink>)</i><
         ↪ century>xiii</century> du latin <foreignWord lang="la" sense
         ↪ ="vase, pot, vaisselle">vas</foreignWord>.</xml>
      <txt>(xiiie siècle) du latin vas (vase, pot, vaisselle).</txt>
    </etym>
  </etymology>
  <!-- ici commence un lemme; il peut y avoir plusieurs lemmes de la
     ↪ même catégorie avec le même genre et le même nombre (ce qui
     ↪ correspond à plusieurs entrées dans un dictionnaire papier) -->
  <pos type="nom" lemma="1" locution="0" homoNb="1" gender="f" number="
     ↪ s">
    <pronunciations>
      <pron>vz</pron>
      <pron>vaz</pron>
    </pronunciations>
    <!-- ici les différentes formes (déclinaison, conjugaison) -->
    <paradigm>
      <inflection form="vase" gracePOS="Ncfs" prons="vz;vaz"/>
```

```
    <inflection form="vases" gracePOS="Ncfp" prons="vz;vaz"/>
  </paradigm>
  <definitions>
    <definition>
      <!-- une première définition, sans étiquette sémantique -->
      <gloss>
        <xml><innerLink ref="bourbe">Bourbe</innerLink>, <innerLink
           ↪ ref="limon">limon</innerLink> qui se <innerLink ref="
           ↪ déposer">dépose</innerLink> au <innerLink ref="fond">
           ↪ fond</innerLink> de la mer, des <innerLink ref="fleuve">
           ↪ fleuve</innerLink>s, des <innerLink ref="étang">étang</
           ↪ innerLink>s, etc.</xml>
        <txt>Bourbe, limon qui se dépose au fond de la mer, des
           ↪ fleuves, des étangs, etc.</txt>
      </gloss>
      <!-- nous ignorons les exemples -->
      <example>
        <xml>
          <i>Le curage des <b>vases</b> de la Darse au moyen d'un
             ↪ dragueur à la vapeur de la force de 14 chevaux, a été
             ↪ complet en 1837.</i>
        </xml>
        <txt>Le curage des vases de la Darse au moyen d'un dragueur à
           ↪ la vapeur de la force de 14 chevaux, a été complet en
           ↪ 1837.</txt>
      </example>
      <!-- ... -->
    </definition>
  </definitions>
  <!-- des relations sémantiques, qui s'appliquent à toutes
     ↪ définitions du lemme (ici, il n'y a en qu'une seule, mais ce
     ↪ n'est pas le cas pour "chat", par exemple -->
  <subsection type="semRel">
    <item type="synonym">boue</item>
    <item type="synonym">fange</item>
    <item type="synonym">limon</item>
  </subsection>
  <subsection type="morpho">
```

```
    <item type="derivative">envaser</item>
    <item type="derivative">envasement</item>
    <item type="derivative">vaseux</item>
    <item type="derivative">vasière</item>
    <item type="derivative">vasard</item>
  </subsection>
<!-- ... -->
</pos>
<!-- ici, un deuxième lemme: un "homonyme" (voir l'attribut homoNb)
   ↪ -->
<pos type="nom" lemma="1" locution="0" homoNb="2" gender="m" number="
   ↪ s">
  <pronunciations>
    <pron>vz</pron>
    <pron>vaz</pron>
  </pronunciations>
  <paradigm>
    <inflection form="vase" gracePOS="Ncms" prons="vz;vaz"/>
    <inflection form="vases" gracePOS="Ncmp" prons="vz;vaz"/>
  </paradigm>
  <definitions>
    <!-- première définition de cet homonyme, sans étiquette -->
    <definition>
      <gloss>
        <xml><innerLink ref="récipient">Récipient</innerLink> pour
           ↪ contenir un <innerLink ref="fluide">fluide</innerLink>
           ↪ ou une <innerLink ref="substance">substance</innerLink>
           ↪ <innerLink ref="granuleux">granuleuse</innerLink>.</xml>
        <txt>Récipient pour contenir un fluide ou une substance
           ↪ granuleuse.</txt>
      </gloss>
      <example>
        <!-- ... -->
      </example>
    </definition>
    <!-- une définition avec une étiquette sémantique -->
    <definition>
      <gloss>
```

```
        <labels>
          <!-- ceci est l'étiquette sémantique -->
          <label type="sem" value="en particulier"/>
        </labels>
        <xml>Récipient de forme <innerLink ref="allongé">allongé</
           ↪ innerLink>e, posé <innerLink ref="debout">debout</
           ↪ innerLink>, utilisé pour contenir un liquide, des
           ↪ cendres, ou pour <innerLink ref="décorer">décorer</
           ↪ innerLink>.</xml>
        <txt>Récipient de forme allongée, posé debout, utilisé pour
           ↪ contenir un liquide, des cendres, ou pour décorer.</txt>
      </gloss>
      <!-- ... -->
    </definition>
    <!-- une autre définition avec une autre étiquette sémantique -->
    <definition>
      <gloss>
        <labels>
          <label type="domain" value="botanique"/>
        </labels>
        <xml><innerLink ref="calice">Calice</innerLink> de certaines
           ↪ fleurs.</xml>
        <txt>Calice de certaines fleurs.</txt>
      </gloss>
      <!-- ... -->
    </definition>
  </definitions>
  <!-- ici, des relations sémantiques -->
  <subsection type="semRel">
    <item type="hyponym">soliflore</item>
  </subsection>
  <!-- .. -->
</pos>
<!-- la suite concerne le terme "vase" en temps que verbe conjugué
   ↪ -->
<pos type="verbe" lemma="0" locution="0">
  <pronunciations>
    <pron>vz</pron>
```

```
      <pron>vaz</pron>
    </pronunciations>
    <!-- cette section permet de trouve le lemme de la forme verbale "
       ↪ vase" -->
    <inflectionInfos>
      <inflectedForm gracePOS="Vmip3s-" lemma="vaser"/>
      <inflectedForm gracePOS="Vmsp3s-" lemma="vaser"/>
    </inflectionInfos>
    <definitions>
<!-- ... -->
```

Avec l'analyse syntaxique, les définitions se présentent ainsi :

```
<gloss>
  <xml><innerLink ref="bourbe">Bourbe</innerLink>, <innerLink ref="limon">limon</innerLink> qui se <
      ↪ innerLink ref="déposer">dépose</innerLink> au <innerLink ref="fond">fond</innerLink> de la mer
      ↪ , des <innerLink ref="fleuve">fleuve</innerLink>s, des <innerLink ref="étang">étang</innerLink
      ↪ >s, etc.</xml>
  <txt>Bourbe, limon qui se dépose au fond de la mer, des fleuves, des étangs, etc.</txt>
  <parsed>1   Bourbe bourbe NC     NC     n=s|g=f|      0      root   0      root   100,00 68,74  44,69
2      ,      ,      PONCT  PONCT          1      ponct  1      ponct  100,00 88,47  95,19
3      limon  limon  NC     NC     n=s|g=m|      1      mod    1      mod    100,00 79,18  92,43
4      qui    qui    PROREL PROREL n=s|  6      suj    6      suj    100,00 97,67  99,80
5      se     se     CLR    CLR    n=p|p=3|      6      aff    6      aff    100,00 98,68  80,96
6      dépose déposer V      V      n=s|t=PS|p=13| 1     mod_rel 1     mod_rel 100,00 92,28 75,54
7      au     à      P+D    P+D    n=s|g=m|      6      mod    6      mod    100,00 97,99  61,80
8      fond   fond   NC     NC     n=s|g=m|      7      prep   7      prep   100,00 93,62  99,28
9      de     de     P      P             8      dep    8      dep    99,33  99,86  91,62
10     la     la     DET    DET    n=s|g=f|      11     det    11     det    100,00 99,77  99,43
11     mer    mer    NC     NC     n=s|g=f|      9      prep   9      prep   100,00 99,21  99,89
12     ,      ,      PONCT  PONCT          11     ponct  11     ponct  100,00 98,82  99,53
13     des    des    DET    DET    n=p|   14     det    14     det    100,00 84,66  89,73
14     fleuves fleuve NC     NC     n=p|g=m|      1      coord  1      coord  100,00 96,16  70,45
15     ,      ,      PONCT  PONCT          14     ponct  14     ponct  100,00 96,70  98,20
16     des    des    DET    DET    n=p|   17     det    17     det    100,00 67,63  96,93
17     étangs étang  NC     NC     n=p|g=m|      14     coord  14     coord  100,00 95,40  81,90
18     ,      ,      PONCT  PONCT          17     ponct  17     ponct  100,00 83,79  96,22
19     etc    _      ADV    ADV    _      1      mod    1      mod    100,00 66,75  74,23
20     .      .      PONCT  PONCT          19     ponct  19     ponct  100,00 100,00 98,05
</parsed>
</gloss>
```

La structure du code XML est bien décrite dans une DTD[27] disponible sur le site Internet du Glawi[28], et la documentation[29] est très détaillée, si bien que nous n'avons pas rencontré de difficultés pour comprendre la structure du Glawi.

27. *Document Type Definition*, un ensemble de déclarations qui décrivent la structure d'un document XML.

28. http ://redac.univ-tlse2.fr/lexiques/glawi/DTD_GLAWI_work.dtd

29. http ://redac.univ-tlse2.fr/lexiques/glawi/doc/index.html

Par contre, sa taille et le grand nombre d'informations qu'il contient nous a posé problème. Sur le plan méthodologique, il nous a d'abord fallu observer attentivement la ressource pour localiser les données dont nous avions besoin et les extraires de la manière la plus efficace. Sur le plan technique, nous avons dû travailler avec un analyseur SAX[30] et programmer un ensemble de classes python pour l'ensemble des tâches décrites précédemment.

Création d'une ressource lexicale pour la tokenisation

Extraire les données du Glawi pour la tokénisation requiert d'extraire non seulement l'ensemble des lemmes (*chat, jouer*), mais aussi l'ensemble des formes déclinées et conjuguées (*chats, chatte, chattes, joue, joues, jouons, jouez, jouent...*). De plus, il nous a paru important d'avoir des informations grammaticales associées à ces formes. La catégorie grammaticale permet d'utiliser des règles avant même le passage de l'étiqueteur automatique. Le genre et le nombre, eux, permettent de corriger, au besoin, la sortie de l'étiqueteur.

Nous avons décidé de créer un dictionnaire pour chaque catégorie grammaticale. Cela permet de choisir les éléments à inclure, et de retravailler à la main certaines catégories importantes pour la référence.

En ce qui concerne les noms, la reconnaissance des noms composés est parfois complexe : *table ronde*, par exemple, peut être compris comme un nom composé, mais aussi comme un nom suivi d'un adjectif. Notre système n'est pas capable de désambiguïser les deux cas, et donc l'utilisation du dictionnaire pour la tokénisation peut être un avantage comme un inconvénient. C'est pourquoi nous avons séparé les noms simples des noms composés (un attribut dans le code XML du Glawi indique s'il s'agit d'un mot simple ou composé[31]), et laisser à l'utilisateur du programme le choix des dictionnaires qu'il veut utiliser.

Il n'y a pas à proprement parler de « verbes complexes », mais il y a des locutions verbales. Il est important de les reconnaître car les syntagmes nominaux qu'elles contiennent ne sont pas référentiels. Cependant, le Glawi ne contient généralement que

30. Le fichier XML est lu par un *analyseur*, c'est-à-dire un algorithme qui transforme le texte en données compréhensibles par un ordinateur. Ici, le fichier est tellement gros qu'il faut le lire élément par élément, ce qui est complexe et demande beaucoup de temps.

31. D'après la documentation, cet attribut est déterminé, pour l'instant, par la présence d'un espace dans la forme du mot.

l'infinitif, c'est pourquoi, ces éléments sont reconnus selon une procédure particulière (voir section 4.2.3).

Nous avons suppléé les entrées du Glawi par celles du Lefff (Sagot 2010) en ce qui concerne les prépositions et les conjonctions, surtout pour les préposition et conjonctions composées (*jusqu'au moment où, à tel point que*, etc.). Le Lefff est en effet plus complet. Cette ressource se présente sous la forme d'une liste de termes, accompagnés du lemme et d'informations grammaticales comme la catégories, le genre, le nombre, le temps, etc. Comme le prépositions et conjonctions sont invariables, il nous a fallu uniquement repérer le code utilisé pour leur catégorie grammaticale et extraires les lignes correspondantes. Nous avons, pour le Lefff, 428 prépositions, 328 conjonctions de subordinations et 25 conjonctions de coordinations, à la fois simples et composées.

Parce que les déterminants et les pronoms jouent un rôle si important pour la résolution de la coréférence, nous avons construit les listes de ces termes à la main, en nous inspirant des données extraites du Glawi, mais en les adaptant fortement avec les remarques proposées par Riegel, Pellat et Rioul (2015). Nous avons particulièrement veillé à introduire des déterminants complexes (*une grande quantité de, toute la gamme de*, etc.), ainsi qu'à catégoriser déterminants et pronoms (définis, indéfinis, démonstratifs, possessifs, etc) : ces indications ne se trouvent pas dans le Glawi (ni dans le Lefff).

Pour les catégories déclinables et conjuguables, il nous a fallu extraire également les informations grammaticales. Cela nous a contraint à convertir les codes utilisés par le Glawi (GRACE[32]) en informations plus simples (genre, nombre et personne sont codés avec des lettres (par exemple *m* pour masculin), les temps et modes sont codés avec la catégorie grammaticale[33].

Ces informations grammaticales sont utilisées dans notre programme à plusieurs niveaux, notamment pour déterminer le genre d'une entité nommée grâce à l'article, ou la correction de la sortie de l'étiqueteur morpho-syntaxique.

32. On trouvera des informations sur ce format aux adresses suivantes : http ://presto.ens-lyon.fr/wp-content/uploads/2014/05/%C3%89tiquettes_Presto-2014-10-13.pdf et https ://alpage.inria.fr/statgram/frdep/Publications/FTB-GuideDepSurface.pdf.

33. Cela ne nous semble pas linguistiquement justifié, mais correspond au jeu d'étiquettes utilisé par Talismane.

Création d'une ressource lexicale pour identifier les expressions non référentielles

Certains syntagmes nominaux, que ce soit des noms ou des pronoms, ne sont pas référentiel. C'est le cas, par exemple, de *porter plainte*, *jouer des coudes*, ou encore *se la jouer*. Ce sont des syntagmes qui sont des compléments verbaux qui ont perdu leur valeur sémantique.

Nous avons donc extrait du Glawi 8 520 locutions verbales. Grâce au moteur de recherche sur les tokens que nous avons écrit, inspiré de CQL[34] (Christ, Schulze, Hofmann et Koenig 1999), et utilisé pour d'autres tâches dans nos travaux, il est possible d'écrire un ensemble de règles, telles que :

(16) `[lemma="porter"][pos="ADV"]*+([lemma="plainte"])`

pour identifier le mot *plainte* lorsqu'il est précédé du verbe *porter*, éventuellement avec un ou plusieurs adverbes après le verbe (*il ne porte pas (vraiment) plainte*)[35].

Création d'une ressource lexicale pour les relations sémantiques

Les relations sémantiques données par la ressources Notre idée de départ était d'utiliser la ressource collaborative *Wiktionaire*, mis en forme (XML) dans le Glawi. Cette ressources contient en effet des listes d'hyperonymes, d'hyponymes et de synonymes, établies manuellement par les interanutes. En outre, il est aussi possible d'utiliser les catégories du Wiktionaire (par exemple : *pomme* a entre autres pour catégorie *Fruits en français*, de même *dague* a pour catégorie *Arme blanche en français* ; l'analyse du nom de la catégorie permet donc de trouver facilement un hyperonyme, ici *fruit* ou *arme*).

Cependant, l'exploitation de ces données s'est révélée délicate, car les synonymes listés par les internautes sont associés à chaque lemme, et chaque lemme peut avoir plusieurs sens. De même, les catégories sont associées à toute l'entrée du dictionnaire, donc à tous les lemmes. Prenons un exemple. Dans l'entrée *chat*[36], il y a trois lemmes (c'est-à-dire trois homonymes, qui correspondrait à trois entrées dans un dictionnaire papier) :

34. *Corpus Query Language*.

35. Même si nous avons tous les éléments pour mettre en place cette fonctionnalité, nous n'avons pas eu le temps, pour l'instant, de l'implémenter.

36. Afin de faciliter la recherche des informations, nous prenons comme exemple la page en ligne (`https ://fr.wiktionary.org/wiki/chat`), consultée le 29 mai 2017.

- *Chat* au sens (entre autres) du félin, mais aussi du poisson (le poisson-chat) ou encore du jeu d'enfants.
- *Chat* au sens (entre autres) d'abri pour un bélier (pour enfoncer des portes), mais aussi au sens (entre autres) de navire.
- *Chat* au sens anglais de conversion par messages électroniques.

Comme synonymes, pour le premier lemme, nous avons *minet, matou*, mais aussi *jeu du loup*, et comme hyperonymes, *félidé*. Mais ces termes sont contradictoires entre eux (*matou* et *jeu du loup* ne sont pas synonymes, bien que chacun soit synonyme de *l'un* des sens du premier lemme). De plus, *félidé* ne s'applique qu'à l'un des sens du premier lemme. Comme catégories, on trouve les jeux tout comme les poissons. Or ces catégories s'appliquent à tous les lemmes, donc aussi bien au sens de *chat* comme navire. Bref, il devient rapidement impossible de savoir quel est le synonyme ou l'hyperonyme de quoi, et la ressource devient inutilisable car *navire* devient un synonyme de *poisson*, lui-même hyperonyme de *jeu*...

Il nous a donc fallu extraire les hyperonymes d'une autre façon.

Extraction de relations sémantiques utilisables L'observation de la ressource nous a amené à faire plusieurs constats. D'abord, les gloses (*i.e.* les définitions) sont rédigées de telle sorte que le premier nom est généralement un hyperonyme (il s'agit de définitions dites « par le genre prochain ») :

- chat :
  - ***Mammifère*** *carnivore félin de taille moyenne...*
  - ***Individu*** *mâle de cet animal.*
  - ***Félin****.*
  - et pour d'autres lemmes : ***Bélier*** *recouvert par un chat.*
  - ***Navire*** *servant au chargement et au déchargement...*
  - etc.
- pomme : ***Fruit*** *comestible du pommier...*
- linguiste : ***Spécialiste*** *de la linguistique.*
- table : ***Meuble*** *composé d'un plateau posé sur un ou plusieurs pieds...*
- taux de natalité : ***Rapport*** *entre le nombre de naissances et la population totale.*

Il est donc possible, en analysant les gloses, d'extraire des hyperonymes qui posent moins de problèmes que ceux définis manuellement.

Deuxième constat : les étiquettes sémantiques associées à chaque gloses permettent parfois d'avoir des hyperonymes ; c'est notamment le cas de l'étiquette « domaine », qui contient des informations telles que « île », « édifice », « fleur ». Pour reprendre notre exemple de *chat*, on trouve « poisson » pour l'étiquette « domaine » de la glose qui indique que *chat* est un synonyme de *poisson-chat*.

Un constat supplémentaire doit être fait à propos de l'ordre d'apparition des définitions et des étiquettes sémantiques attachées à ces définitions. Les définitions vont généralement de la plus générale à la plus spécialisée. Ainsi, les gloses de *chat* commencent par le mammifère dont on peut voir d'innombrables vidéos sur Internet, et termine par les discussions par messages életroniques (anglicisme), en passant par les poissons-chats, les abris de béliers et les navires, dans cet ordre. Les étiquettes sémantiques associées donne également de bons repères pour séparer les différents sens. Les deux premiers sens de *chat* (mammifère et félin) n'ont pas d'étiquettes particulières. Par contre, dès qu'on passe à la glose « félin », on a une étiquette « par extension »[37]. Ensuite toutes les gloses suivantes ont des étiquettes : « domaine : poissons »[38], « domaine : jeux », « domaine : mécanique », « attitude : vulgaire » (« sexe de la femme »), « diatopique : Lorraine » (« fourrure que les femmes portent au cou »), etc.

Ces jeux d'étiquettes vont nous permettre d'établir des *groupes sémantiques* : toutes les gloses qui ont les mêmes étiquettes sémantiques formeront un même groupe sémantique, dont les hyperonymes seront extraits des gloses qui composent le groupe. Le Glawi étant incomplet, il y a des mélanges dans les groupes sémantiques, mais cela permet déjà d'éviter les erreurs les plus grossières, comme celle qui ont été évoquées ci-dessus.

Dernier constat : les hyperonymes sont transitifs. Cela signifique que *chat* a pour hyperonyme *mammifère*, qui lui-même a pour hyperonyme *animal*. Donc *chat* a pour hyperonyme *mammifère*. On peut ainsi continuer : un *animal* est un *métazooaire*, etc.

Les hyperonymes et synonymes dont on dispose Ces constats nous amènes à proposer divers types d'hyperonymes (dans lesquels nous incluons les synonymes, pour simplifier).

37. On rappelle qu'un chat est un félin, mais que tous les félins ne sont pas des chats, sauf « par extension » de la définition.

38. Les étiquettes sémantiques sont composées d'un type d'étiquette, qui peut être *domain, diachronic, diatopic, sem*, etc. et d'une valeur.

- Les hyperonymes extraits des catégories.
- Les hyperonymes donnés par les auteurs du Wiktionnaire.
- Les hyperonymes extraits des étiquettes sémantiques.
- Les hyperonymes extraits des gloses.

Nous avons vu que les deux premiers ne sont pas fiables, puisqu'ils ne s'attachent pas à un sens, mais à un lemme (donc à plusieurs sens) ou même à une entrée (donc à plusieurs homonymes). C'est pourquoi nous ne calculons les hyperonymes par transitivité qu'à partir des deux derniers.

Extraction des informations Nous en avons tout d'abord établi la liste de des catégories du Wiktionnaire (avec nettoyage manuel des erreurs, par exemple des fautes). Nous en avons ensuite fait l'analyse automatique, en utilisant Talismane (Urieli 2013). Enfin, nous avons extrait les hyperonymes de la même façon que pour les hyperonymes des définitions (mais les définitions sont déjà analysées syntaxiquement, comme nous l'avons évoqué plus haut). Pour ce faire, nous avons utilisé un moteur de recherche sur les tokens, inspiré de CQL[39] (Christ, Schulze, Hofmann et Koenig 1999), et utilisé pour d'autres tâches dans nos travaux. Par exemple, pour chercher un nom éventuellement précédé ou suivi d'un adjectif, on peut utiliser le patron :

(17) `[pos="ADJ"]?[pos="NC"][pos="ADJ"]?`

Un tel patron est suffisant pour les catégories, dont les noms sont simples. Par contre, les définitions sont beaucoup plus riches, et nous avons tenté d'exploiter cette richesse pour obtenir le plus de termes possibles. Nous avons donc établi plusieurs patrons qui cherchent non seulement le premier nom commun, mais aussi le premier groupe nomimal (avec ces expansions). Cela permet de récupérer des noms composés (*mode de paiement*) qui serait sinon ignoré[40]

Nous avons également veillé à extraire tous les hyperonymes coordonnés, comme dans la définition de *prémium*, qui se présente comme une liste de synonymes : « Avantage, bénéfice, prérogative, privilège, faveur ».

39. *Corpus Query Language*.

40. Accessoirement, cela permet également de trouver des hyperonymes plus détaillés, comme *lame* et *petite lame* pour *lamelle*. Cela ne nous est, pour l'instant, pas utile, car nous considérons uniquement le mot tête du syntagme, mais permet d'étoffer la ressource sémantique que nous avons construite, et qui peut servir à d'autres usages que notre programme.

Parfois, cependant, le premier groupe nominal n'est pas le terme recherché, mais le deuxième. C'est le cas par exemple des définitions qui commencent par *sorte de, espèce de*, etc., comme dans la définition de *libocèdre* : « Sorte de conifère. » Dans ce cas-là, nous prenons le (ou les) deuxième groupe nominal comme hyperonyme.

Une fois les hyperonymes extraits pour chaque catégorie et chaque définition, nous avons extrait les hyperonymes des étiquettes sémantiques de type « domaine ». Certaines de ces étiquettes ont en effet pour valeur « poissons », « jeux », etc., ce qui permet facilement de trouver un hyperonyme en mettant simplement la propriété au singulier. Mais parfois les valeurs de ce type d'étiquettes désigne plutôt une activité, comme « maritime » ou « ichtyologie ». Il nous a donc fallu faire le tri manuellement, puis comparer, pour chaque définition, les étiquettes sémantiques pour trouver celles qui pouvaient réellement être utilisées comme hyperonymes.

Nous avons ensuite établi des groupes sémantiques, c'est-à-dire des définitions successives qui ont les mêmes étiquettes sémantiques. Nos regroupement d'hyperonymes se font en effet sur la base des étiquettes sémantiques plutôt que des lemmes (trop larges) ou des définitions prises individuellement (trop précises). Tous les hyperonymes trouvés dans les définitions appartenant à un même groupe sémantique sont donc réunis.

C'est en utilisant les groupes sémantiques et leur ensemble d'hyperonymes que nous avons cherché des hyperonymes par transitivité. L'un des principaux problèmes rencontrés est celui de la récursivité, notamment pour les noms abstraits. Si l'utilisation des définitions pour trouver les hyperonymes fonctionne bien pour les objets (*chat* > *mammifère* > *animal*), en ce qui concerne les noms abstraits, il s'agit plutôt de synonymes, et il n'est pas rare qu'un terme soit défini avec un autre terme, lui-même défini avec le premier terme. Ainsi, *groupe* renvoie à *ensemble*, qui renvoie *réunion*, qui renvoie à *collection*, qui lui-même renvoie à *réunion* et nous voilà dans une boucle sans fin...

Une autre difficulté est la présence, fort nombreuse dans le Glawi, de variantes orthographiques ou régionales. Ces variantes sont très mal identifiées : il y a des indices, comme une définition qui commence par « variante de », mais ils ne sont pas suffisants. En effet, les variantes peuvent être indiquées à plusieurs endroits :

- une entrée peut lister des variantes orthographiques au niveau global (tous lemmes confondus),
- chaque lemme peut lister d'autres variantes orthographiques (ou les mêmes),
- une entrée peut elle-même être une forme alternative avec une définition qui commence par « variante de ».

Nous avons essayé de trouver des règles, mais en vain : tous les termes qui ont au moins une définition en « variante de » ne sont pas repris dans le terme auquel il est fait mention, et l'inverse n'est pas vrai non plus. De plus, les entrées qui ont des définitions en « variante de » peuvent contenir aussi d'autres définitions, qui ne sont pas incluses l'entrée à laquelle il est renvoyée. Il n'y a donc aucune cohérence.

Par ailleurs, *tchat*, par exemple, est noté comme variante de *chat*, mais il n'est pas indiqué que cela concerne seulement l'un des lemmes (celui dont le sens est « conversation électronique »), et non tous les lemmes (notamment celui dont le sens « mammifère »). Pour éviter, donc, que *animal* soit considéré comme hyperonyme de *tchat*, nous avons traité les variantes comme des mots indépendants. Notre dictionnaire contient donc une entrée *chat* avec ses hyperonymes (dont ceux relatifs aux conversations), et une entrée *tchat* avec ses propres hyperonymes. Cela ne devrait pas poser problème, puisqu'il ne devrait pas y avoir de variation d'orthographe au sein d'un même texte (un auteur n'alternera problablement pas entre *tchat* et *chat*).

Paramètres Pour pallier ces difficultés, le calcul des hyperonymes peut être paramétré de plusieurs façons. D'abord, les hyperonymes peuvent apparaître ou non comme entrée du dictionnaire (c'est-à-dire qu'un hyperonyme n'est gardé que s'il s'agit d'un mot du dictionmaire). Cela permet d'éliminer les hyperonymes qui ne sont pas reconnus comme noms communs, mais cela peut éliminer à tort des hyperonymes valables, mais qui ne seraient pas dans le dictionnaire.

Lors de la recherche des hyperonymes par transitivité, il est possible de couper la hiérarchie en fixant certains termes. Par exemple, *chat* remonte à *métazooaire*. Mais tous les animaux sont des métazooaires : ce n'est donc pas un hyperonymes très pertinent, ni surtout très utilisé. Définir des hyperonymes trop généraux est un inconvénient pour un outil de détection automatique, car il y a trop de candidats. Il est donc possible de définir le terme le plus haut dans la hiérarchie, comme : « animal », « objet », « chose », « activité », « lieu », etc.

Il est aussi possible de limiter le niveau hiérarchique. Il n'est sans doute pas utile de remonter jusqu'à dix niveaux. Cela donne en effet des hyperonymes trop généraux, comme dans la suite *fossé* > *fosse* > *cavité* > *vide* > *espace*, mais facilite aussi l'apparition d'erreur, comme dans *fossé* > *fosse* > *cavité* > *creux* > *concavité* > *côté* > *région* > *territoire* > *aire* > *quantité*.

Enfin, il est possible d'éliminer des hyperonymes inconditionnellement. C'est le cas des hyperonymes les plus fréquents (dont on peut facilement établir la liste). Ce sont sou-

vent des hyperonymes peu significatifs, artifacts des gloses, comme *synonyme, variante, nom (de)*. D'autres sont des hyperonymes trop généraux pour être utiles, comme *type* ou *état*.

Nous avons créé un script particulier qui permet de parcourir les hyperonymes calculés par transitivité aléatoirement, en « faisant jouer » ces différents paramètres. Cela permet de tenter de trouver le meilleur compromis de façon intuitive.

Autres extractions à partir du Glawi

Le Glawi contient aussi d'autres informations que nous avons extraites. Nous avons créé une liste de prénoms, avec leur genre (masculin, féminin, ou épicène, ce qu'il a fallu calculer si le prénom apparaissait deux fois avec deux genres différents). Nous avons aussi créé une liste de noms propres.

Ces deux listes sont utilisées pour trouver des entités nommées de personnes qui n'apparaissent pas dans Yago.

4.2.4 Chargement des informations lexicales dans ODACR

Les données extraites du Glawi sont complexes, et nous avons voulu donner à l'utilisateur de notre programme la possibilité de choisir exactement celles qu'il voulait. Plusieurs options existent donc :

- Il peut choisir quelles sont les étiquettes sémantiques qu'il veut privilégier. Par exemple, dans un texte spécialisé dans l'ichtyologie, il pourra sélectionner ce domaine sémantique. Cela permettra de relier, par exemple, *poisson* et *chat*. Au contraire, dans un texte plus commun, il pourra choisir de privilégier les termes sans étiquettes sémantiques particulières : *poisson* et *chat* ne pourront alors pas être reliés.
- Certaines étiquettes peuvent être exclues, ou au contraire être exclusives (c'est-à-dire que celles qui ne sont pas mentionnées sont totalement exclues).
- Certaines étiquettes peuvent être fusionnées, par exemple si l'on veut traiter le registre courant et le registre familier avec la même priorité.
- Cette notion de priorité est utile si l'on ne veut garder que les n premiers groupes sémantiques (ceux-ci sont par défaut classer par ordre d'apparition dans le dictionnaire, c'est-à-dire par ordre du plus commun au plus spécialisé).

- Il est possible de ne garder que les hyperonymes des définitions, ceux trouvés par transitivité, ou bien ceux donnés par les internautes, ou bien un mélange.
- Il est également possible de supprimer les groupes sémantiques qui n'ont pas d'étiquettes sémantiques mais qui ne sont pas les premiers. Cela arrive notammment lorsqu'il y a plusieurs lemmes (correspondant à plusieurs entrées) dans le dictionnaire : les lemmes qui ne sont pas en premier position sont souvent spécialisés, même s'ils sont dépourvus d'étiquettes sémantiques.

4.2.5 Perspective

Nous pouvons envisager d'autres extractions à partir du Glawi, notamment le nombre important d'entités nommées, pour complémenter Yago. Les désignations communes (*Rhin... fleuve*) pourraient être extraites des définitions, comme nous l'avons fait pour les hyperonymes des noms communs (la définition de *Rhin* est en effet : « *Fleuve* prenant sa source dans les Alpes, et se jetant dans la mer du Nord »). Cela permettra à la fois de compléter les désignations trouvées dans Yago, et permettra d'ajouter de nouvelles entités.

4.3 Conclusion

Nous avons donc élaboré deux ressources lexicales pour le français. La première est consacrée aux entités nommées. Plus qu'une liste et une série de règles pour leur détection, il s'agit d'une véritable ressource sémantique qui inclut, pour chaque entité, un ensemble de désignations de l'entité (noms propres) et de désignations communes (« hyperonymes »). La deuxième est un dictionnaire d'hyperonymes extraits à partir des définitions du Wikitionnaire. Ces deux ressources sont les premières du genre pour le français.

Ces ressources ne sont toutefois pas totalement achevées. Si la partie automatique est fonctionnelle, il reste, dans le cas des entités nommées, à définir les catégories principales et à vérifier les traductions des catégories WordNet ; dans le cas des hyperonymes, à sélectionner les meilleurs paramètres pour que le nombre de termes retenus soit suffisant sans toutefois être trop important. Ces tâches sont manuelles et, sans être particulièrement difficiles, sont chronophrages.

Chapitre 5

Besoins en analyse syntaxique

5.1 Motivation

Un système *end-to-end* reçoit un texte brut en entrée. Sa première tâche est donc de repérer les expressions référentielles qui entreront (ou non) dans la constitution des chaînes de référence. Plusieurs techniques permettent de réaliser ce repérage :

- étiqueter le texte en morpho-syntaxe (c'est-à-dire mettre une catégorie grammaticale sur chaque mot), puis réunir les expressions référentielles par un ensemble de règles (par exemple : « DET + ADJ + N » peut former un groupe nominal et donc une expression référentielle). C'est l'approche utilisée dans RefGen, avec un étiquetage fait de façon automatique par TTL (Ion 2007),
- trouver les dépendances syntaxiques (un peu comme des arbres syntaxiques) et chercher toutes les têtes syntaxiques nominales ou pronominales ; l'expression référentielle est alors constitué de tout ce qui dépend de la tête (par exemple : si un déterminant et un adjectif dépendent d'un nom, alors trouver le nom suffit à récupérer toute l'expression référentielle). C'est l'approche que nous avons suivie[1]. Cette méthode nous permet de récupérer des informations syntaxiques (comme la fonction de l'expression référentielle, son niveau de dépendance[2], etc.).

1. En remplacement de TTL (Ion 2007), puisque celui-ci n'est pas libre de droit et n'est disponible que sous forme de service web limité.

2. Dans la proposition principale, ou bien dans une subordonnée, ou bien encore dans la subordonnée d'une subordonnée, etc.

5.2 Qu'est-ce qu'un analyseur syntaxique ?

Un analyseur syntaxique est un programme informatique qui cherche automatiquement les relations de dépendances syntaxiques entre différents tokens (des mots et des ponctuations). Il ne s'agit pas d'une analyse en constituants immédiats, mais plutôt de la désignation du rapport entre un élément et l'élément dont il dépend, à la manière des *stemmata* de Lucien Tesnière. Par exemple, l'analyseur indiquera que tel token de type *déterminant* sera dépendant de tel token de type *nom*.

L'analyseur accepte en entrée un texte *tokenizé*, c'est-à-dire découpé en tokens (mots (simples ou complexes) et ponctuations). Ces tokens sont ensuites *lemmatisés* et *étiquetés* (avec la catégorie grammaticale de chaque token). Enfin, l'analyseur peut construire les dépendances. Certains analyseurs ont des modules spécifiques qui réalisent les tâches préalables à l'analyse syntaxique, mais ces modules peuvent être remplacés par d'autres outils, parfois plus performants[3].

Ces différentes étapes se retrouvent dans la sortie des analyseurs. Voici un exemple de sortie (de Talismane[4]) pour la phrase « Je suis parti en vacances » :

```
1 Je je CLS CLS n=s|p=1| 3 suj 3 suj
2 suis être V V n=s|t=P|p=1| 3 aux_tps 3 aux_tps
3 parti partir VPP VPP n=s|g=m|t=K| 0 root 0 root
4 en en P P  3 mod 3 mod
5 vacances vacance NC NC n=p|g=f| 4 prep 4 prep
6 . . PONCT PONCT  5 ponct 5 ponct
```

Chaque ligne contient un token. Les colonnes sont séparées par des caractères de tabulation. La première colonne contient l'index (*i.e.* le numéro du token dans la phrase). Puis viennent successivement la forme, le lemme, la catégorie grammaticale, des informations morphologiques, l'index du token « parent », le type de dépendance. Trois colonnes sont répétées deux fois : c'est parce que ce format est un format standard (CoNLL-X[5]), qui peut contenir plus d'informations que celle fournies par Talismane. Il convient donc d'ignorer les colonnes répétées.

3. Nous tirons profit de cette possibilité avec Talismane, que nous utilisons seulement pour l'étiquetage et l'analyse syntaxique : nous faisons nous-même la tokenisation, comme nous l'expliquons ci-dessous.

4. Que nous présentons ci-dessous.

5. La description complète du format est disponible à l'adresse http ://web.archive.org/web/20161105025307/http ://ilk.uvt.nl/conll/.

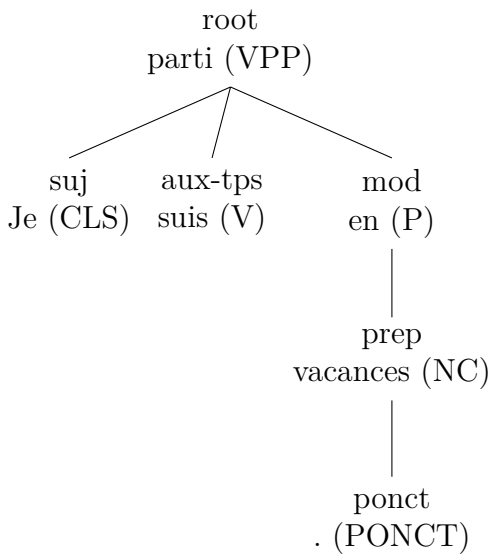

FIG. 5.1 : Arbre de dépendances pour « Je suis parti en vacances ».

Le token « parent », ou « tête », est simplement le token auquel est rattaché un autre token. Dans l'exemple précédent, par exemple, *Je* est rattaché à *parti* (sa colonne « parent » indique 3, qui est l'index de *parti*), verbe principal de la phrase, de même que *suis*, l'auxiliaire de *parti*. Chaque phrase contient un élément *racine* (*root* en anglais), noté 0 : c'est l'élément dont dépendent tous les autres (ici, il s'agit du verbe principal *parti*).

Afin de lire plus facilement la sortie de l'analyseur, on peut représenter les dépendances sous la forme graphique d'un arbre (fig 5.1 et 5.2) :

5.3 Ressources disponibles

Nous avons testé plusieurs analyseurs syntaxiques qui ont des ressources pour le français (nous n'avons pas cherché, dans le cadre de ce mémoire, à entraîner un nouveau modèle[6], et nous avons donc utilisé les modèles livrés par défaut avec les outils) :

6. Un modèle est le résultat de l'apprentissage : c'est ce qui permet de trouver les relations syntaxiques dans un texte nouveau. Les modèles varient donc en fonction du corpus d'entraînement (par exemple, on peut entraîner un modèle spécifique aux textes juridiques, ou aux textes médicaux, ou bien à l'ancien français, etc.)

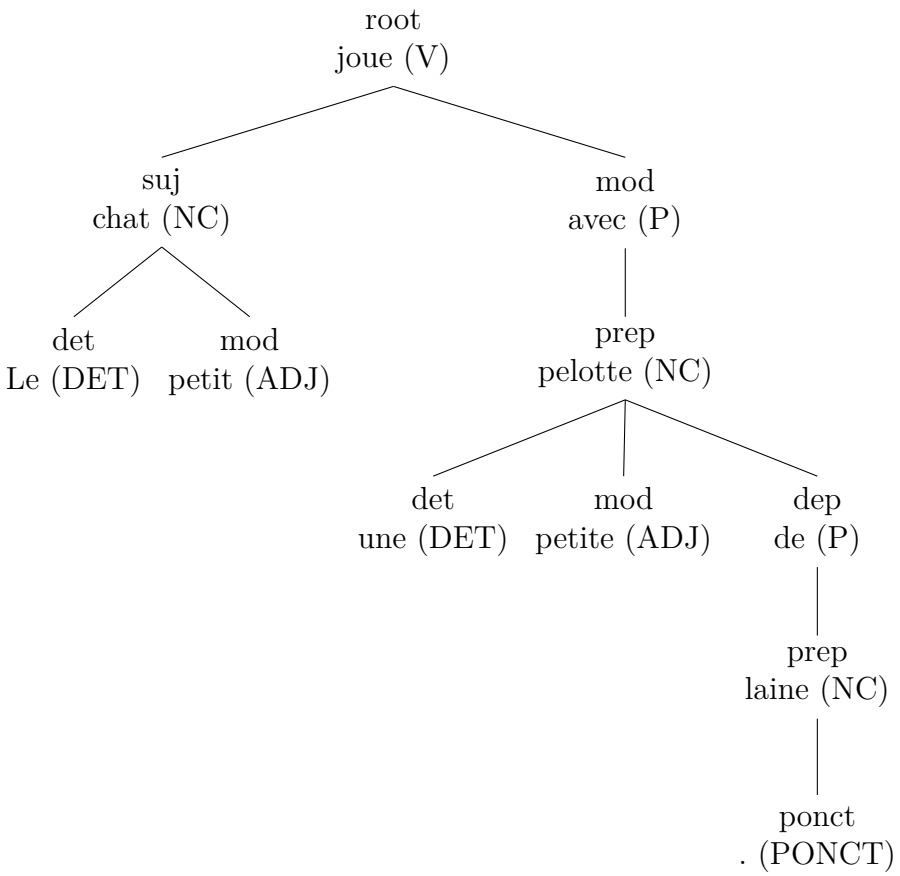

FIG. 5.2 : Arbre de dépendances pour « Le petit chat joue avec une petite pelotte de laine ».

- MSTParser (McDonald, Lerman et Pereira 2006),
- Berkeley Parser (Petrov, Barrett, Thibaux et Klein 2006),
- Fips (Wehrli 2007),
- l'analyseur de Bohnet, Nivre, Boguslavsky, Farkas, Ginter et Hajič (2013),
- Talismane (Urieli 2013),

Fips est le seul analyseur à base de règles. Il est un peu différent des autres outils parce qu'il est plus proche des théories de la grammaire générative. Alors que les autres outils se contentent de noter les relations, Fips donnent des informations supplémentaires, comme le « mouvement » (au sens générativiste) de certains éléments. Cependant, Fips ne fait pas une application stricte de la théorie X-bar : il en propose une adaptation simplifiée (par exemple, les arbres X-bar sont nécessairement binaires, ce qui n'est pas le cas des arbres de Fips). Malgré la précision de ces résultats, il échoue souvent à analyser des phrases complexes. Nous l'avons donc éliminé.

Il en va de même pour l'outil de Bohnet, Nivre, Boguslavsky, Farkas, Ginter et Hajič (2013), dont le modèle pour le français n'est pas libre et redistribuable.

Selon Urieli et Tanguy (2013), les trois analyseurs restants ont des résultats comparables ; nous avons préféré Talismane, car le plus ouvert en terme de droits et le plus facile à intégrer, du point de vue technique, à notre programme.

Notre programme est cependant conçu pour qu'un autre analyseur puisse être utilisé, sous réserve de fournir une conversion du jeu d'étiquettes morpho-syntaxique.

5.4 Intégration d'un parser

5.4.1 Tokénisation

L'intégration du parseur nous a cependant posé quelques problèmes. Le premier a été celui de la tokenisation.

- Talismane est mauvais pour la reconnaissance des unités qui se répartissent sur plusieurs tokens : « parce que », par exemple, est parfois reconnu comme un adverbe suivi d'un pronom relatif...

- Par ailleurs, nous avons vu que la reconnaissance des entités nommées est très importante pour la résolution de la coréférence. Or Talismane ne dispose d'aucun détecteur d'entités nommées : il reconnaîtra donc « Ville lumière » comme un nom suivi d'un adjectif.
- Talismane est perturbé par la présence de signes « parasites », comme les guillemets, les titres de civilités (*M., Mme*, etc.), les incises, les parenthèses, les puces, les numéros de paragraphes, etc.
- Enfin, il est nécessaire de garder trace de la position de chaque token pour pouvoir reproduire en sortie le texte *original* annoté. Même si Talismane dispose d'une option qui ajoute la position de chaque token dans le texte original, il devient impossible de prétraiter le texte (c'est-à-dire de modifier le texte avant de l'envoyer à Talismane).

Pour toutes ces raisons, nous avons écrit un module de tokénisation, qui s'occupe non seulement de tokéniser le texte, mais aussi de pré-annoter le texte, tout en tâchant de founir à l'analyseur syntaxique un texte « nettoyé » (sans parenthèse, guillemet, incise, etc.).

Le module commence donc par une tokénisation de base (découpage en paragraphes et en tokens simples). Il est possible de définir des tokens « spéciaux », par exemple des symboles chimiques avec des lettres grecques (cas fréquents dans les articles scientifiques).

Au cours de cette tokénisation, les parenthèses sont supprimées[7]. En effet, leur contenu est par définition « hors propos » : il s'agit de digressions ou de références qui ne font pas vraiment partie du texte. Nous avons donc fait le choix de les ignorer totalement, ce qui a l'avantage de ne pas perturber l'analyseur syntaxique inutilement, même si on court le risque de perdre de rares maillons de chaînes.

Puis le programme lit des dictionnaires afin d'identifier les unités polylexicales[8] et les entités nommées. Des informations morphologiques sont associées à chaque token dès cette étape (catégorie grammaticale, genre, nombre, type de l'entité nommée, etc.).

Les dictionnaires sont ceux créés à partir de Yago et du Glawi (comme décrit au chapitre 4), mais on peut ajouter des dictionnaires personnalisés pour ajouter, par

7. Elles ne sont supprimées que dans la représentation *interne* du texte. La sortie retrouve les parenthèses, mais leur contenu ne contiendra pas d'annotation.

8. La détection des unités polylexicales est parfois complexes : *table ronde* peut être un mot composé mais aussi un nom suivi d'un adjectif. L'utilisation de dictionnaire de noms composés est donc optionnelle.

exemple, une entité nommée qui ne se trouve pas dans Yago, ou pour forcer l'utilisation de tel ou tel homonyme (deux personnes différentes de même nom, par exemple, auront des fonctions différentes : le programme ne peut pas décider lesquelles utiliser de lui-même parce qu'il ne saura pas à laquelle des deux personnes le texte fait référence).

Puisque les déterminants et les pronoms sont particulièrement importants dans la détection de la coréférence, nous avons veillé à écrire des dictionnaires spécialisés pour ces deux parties du discours, en incluant la personne (première, deuxième, troisième), le type du déterminant ou du pronom (démonstratif, possessif, etc.).

Il est également possible de définir des expressions régulières ou des règles. Les expressions régulières permettent d'associer des tokens en fonction de leur forme (par exemple : « Chapitre 5 », « chap. 5 » et « ch. 5 » peuvent être reconnus comme trois formes qui doivent avoir le même lemme). Les règles permettent d'associer des tokens en fonction de leurs propriétés et de leur contexte (par exemple *les deux* suivi d'un nom sera déterminant, alors que s'il est suivi par un mot d'une autre catégorie grammaticale, il sera pronom). Un ensemble de règles permettent d'associer les dates et les nombres.

Un algorithme tente ensuite de localiser les incises (voir l'exemple 18). Cela a deux objectifs :

- déplacer l'incise avant la citation afin d'en faire une phrase à part qui sera analysée indépendamment par l'analyseur syntaxique (les analyseurs sont en général très mauvais avec les phrases qui contiennent des incises),
- noter l'incise comme du texte qui n'est pas du discours direct.

Notre algorithme de résolution de la coréférence prend en effet en compte le discours direct. Les pronoms de première et deuxième personnes, quand ils sont dans du discours direct, vont alors être associés avec un nom qui n'est pas dans du discours direct. On passera donc de :

(18) a. Paul se retourna. « Je ne crois pas, dit-il, que Pierre a fait cela. »

à :

b. Paul se retourna. Dit-il. « Je ne crois pas que Pierre a fait cela. »

Ce qui permettra à l'algorithme de former la chaîne suivante, en veillant à associer un pronom de troisième personne au pronom de première personne du discours direct[9].

c. $[\text{Paul}]_i$ se retourna. Dit-$[\text{il}]_i$. « $[\text{Je}]_i$ ne crois pas que Pierre a fait cela. »

soit, en sortie[10] :

d. $[\text{Paul}]_i$ se retourna. « $[\text{Je}]_i$ ne crois pas, dit-$[\text{il}]_i$, que Pierre a fait cela. »

Une heuristique permet ensuite d'enlever les guillemets et d'en typer le contenu. Cela vise encore une fois le double objectif d'une pré-annotation des tokens et d'un nettoyage du texte fournit à l'analyseur syntaxique. Des règles identifient les guillemets comme :

- indiquant un titre (c'est le cas par exemple s'il y a une entité nommée entre les guillemets),
- indiquant du discours direct (s'il y a des pronoms de première ou deuxième personnes, s'il y a des crochets d'édition, si les guillemets sont précédés de deux points ou s'ils ouvrent une phrase, etc.),
- indiquant une mise en relief, une ironie, un euphémisme, etc. (dans les autres cas). Les guillemets sont alors simplement supprimés et le texte n'est pas annoté.

Des listes de prénoms et de noms de famille permettent de former des entités nommées de type *personne* qui ne sont pas dans le dictionnaire des entités nommées.

Suit une heuristique qui tente de former des entités nommées avec les mots en majuscules (s'ils sont en début de phrases, ils doivent n'être reconnus par aucun dictionnaire) : soit des acronymes (considérés comme des entités nommées de type *organisation*), soit des personnes.

Une fois ce travail réalisé sur les tokens, le module va séparer le texte en différentes phrases selon un ensemble de règles qui se basent sur la ponctuation, en essayant d'éviter les points qui ne délimitent pas des phrases (point des acronymes, des abréviations, etc.).

Les tokens, groupés en phrases, sont alors envoyés à l'analyseur syntaxique.

9. L'algorithme ne permet pas de retrouver les chaînes des pronoms de première et deuxième personnes dans les dialogues, puisque le tokéniseur ne reconnaît pas, pour l'heure, les tours de parole (cela pose des problèmes, notamment lorsqu'il y a plus de deux locuteurs dans un même dialogue).

10. L'exemple (c) est une représentation *interne* du texte. La sortie (en (d)), elle, contient le texte original, sans modifications autres que les annotations.

5.4.2 Règles de correction

La plupart des analyseurs syntaxiques sont à base d'apprentissage automatique. C'est le cas de Talismane, l'analyseur que nous avons choisi pour nos expérimentations[11]. Or l'analyse syntaxique résultante est souvent erronée. Il ne s'agit pas seulement de phrases ambiguës (comme « Paul a vu la femme avec un téléscope ») que seules des connaissances extérieures permettent de comprendre, mais d'erreurs grossières comme faire dépendre une proposition relative d'un verbe ou encore coordonner une préposition et un adverbe. Aucun contrôle n'est effectué par l'analyseur syntaxique (en tout cas par Talismane).

Nous avons donc écrit un ensemble de règles de correction. Chacune des règles a accès à l'arbre syntaxique construit par l'analyseur et modifié par les règles précédentes. Il est donc possible de rattacher une proposition relative à un nom ou de tenter de coordonner des éléments de même catégorie grammaticale (au sens large : un adjectif peut être coordonné à une proposition relative, par exemple, mais pas à un verbe). Parfois, il est même possible de changer le type d'un token, par exemple de désambiguiser entre le *que* pronom relatif et le *que* conjonction de subordination.

Cela est nécessaire pour plusieurs raisons :

- une bonne délimitation des expressions référentielles (si un adjectif ou une relative est rattaché au mauvais élément, les mentions ne seront pas bien délimitées),
- une bonne précision syntaxique pour l'algorithme de résolution de la coréférence, qui utilise la syntaxe pour déterminer la fonction des candidats et leur niveau de dépendance, mais aussi et surtout pour rechercher tous les antécédents candidats possibles d'un pronom,
- former les groupes (*Pierre et Paul*),
- séparer les phrases coordonnées et juxtaposées, et définir les sujets zéros.

Nous n'avons pas la place d'énumérer l'ensemble des règles que nous avons élaborées, nous donnerons donc seulement quelques exemples.

Pronom impersonnel

Le premier ensemble de règles permet d'identifier les pronoms impersonnels (le *il* dans « il pleut ») afin de les marquer, de ne pas les considérer comme expressions référentielles

11. Notre programme peut toutefois être connecté à un autre analyseur syntaxique.

et de ne pas leur chercher d'antécédent[12]. Pour ce faire, nous avons écrit une sorte de moteur d'expressions régulières qui recherche des motifs dans une suites de tokens (par exemple un déterminant suivi d'un adjectif suivi d'un nom). La syntaxe est inspirée de CQL[13] (Christ, Schulze, Hofmann et Koenig 1999). Par exemple, le patron suivant :

```
([lower="il"])[pos="ADV"]*+[lemma="faire"][pos="ADV"]*+
  [lemma="@WEATHER_ADJ_FOR_DUMMY"]
```

capturera un pronom *il* (il est dans un groupe capturant, ce qui signifie que c'est ce token qui va être marqué comme pronom impersonnel) suivi de zéro, un ou plusieurs adverbes, suivi du verbe *faire* conjugué, suivi de zéro, un ou plusieurs adverbes, suivi d'un adjectif de météorologie (*beau*, *chaud*, *venteux*, etc.).

Ainsi, ce patron capturera les phrases suivantes et annotera le pronom comme impersonnel :

(19) a. Il fait beau.

b. Il fait chaud.

c. Il fait très chaud.

d. Il ne fait pas très chaud.

etc.

Relations syntaxiques

Règles La plupart des règles concernent la correction des relations syntaxiques.

- Certaines vont simplement faire du nettoyage et simplifier l'arbre en éliminant les ponctuations de fin, en supprimant les adverbes (qui ne sont pas importants pour la résolution de la coréférence), éventuellement en reportant des informations sur la tête syntaxique (par exemple en marquant le verbe comme négatif s'il s'agit d'un adverbe de négation), etc. Il faut noter que les auxiliaires verbaux, qui n'entrent

12. Nous avons eu accès au fichier XML qui défini les patrons définis par Longo (2013) et utilisés pour RefGen. Le code XML est quelque peu cryptique ; nous nous sommes pourtant efforcé de retrouver les phrases que les règles capturaient. Les listes d'adjectifs, de participes passés, de verbes, etc., étaient aussi définis dans ce fichier. Nous les avons reprises telles quelles.

13. *Corpus Query Language*.

pas dans la détection de la coréférence, sont également supprimés (*j'ai mangé* devient *j' mangé*, et le participe est marqué comme verbe).

- D'autres veilleront à ce qu'il n'y ait qu'une seule racine[14], et que cette racine soit un verbe plutôt qu'un nom (sauf dans le cas des phrases nominales).
- D'autres vont corriger des erreurs grossières : les déterminants sont attachés à des noms (et non à des verbes, des adverbes, etc.), les pronoms relatifs à des verbes ou des prépositions (et non à des noms, des adjectifs, etc.), les noms objets directs sont déplacés lorsqu'il y en a plusieurs attachés au même verbe (un verbe ne peut avoir qu'un seul objet direct), tout comme les noms modifieurs d'adjectifs (ce qui n'est pas possible), les prépositions sans régime cherchent à adopter un nom ou un infinitif voisin (les propositions peuvent ne pas avoir de régime, mais seulement avant une ponctuation : *Je suis contre.*).
- D'autres règles vont modifier la catégorie grammaticale d'un token. Par exemple, deux verbes dont le premier est dans une relative et le deuxième est subordonné et introduit par la conjonction *que* ne peuvent pas être coordonnés. S'ils sont coordonnés, c'est en fait que le *que* est un pronom relatif, et non une conjonction. Cela est particulièrement important pour la détection des expressions référentielles, car le pronom relatif en est une, la conjonction non.
- D'autres règles vont modifier la structure interne de certains tokens, comme les amalgames qui sont éclatés (*du* va être transformé en *de le*).
- D'autres règles vont « améliorer » la structure de l'arbre :
  - Les éléments nominaux (mais seulement eux) coordonnés vont être réunis en un nouvel élément, un *groupe*. En effet, un groupe peut être repris (voir l'exemple 23 et il est donc important de les annoter.
  - L'analyseur syntaxique ne fait pas trop d'erreurs lorsque deux éléments sont coordonnés par une conjonction de coordination (souvent, toutefois, il ne choisit pas les bons éléments à coordonner, mais cela ne pose pas de problème, puisque c'est facile à corriger). Mais il arrive que des éléments soient coordonnés par une ponctuation (généralement une virgule). C'est notamment le cas lorsque trois éléments sont coordonnés (*A, B et C*). Certaines règles vont tenter de retrouver ces situations et de les corriger. Cependant, ces règles in-

14. Certains analyseurs veillent à ce qu'il n'y ait qu'une seule racine (c'est le cas de l'analyseur de Bohnet, Nivre, Boguslavsky, Farkas, Ginter et Hajič (2013)), mais Talismane ne respecte pas cette contrainte.

```
def add_relative_info_to_verb(s) : # s is a sentence object
    for t in s.tokens :
        if t.is_relative_pronoun :
            parent = s.get_parent(t)
            assert parent
            if parent and parent.is_verb :
                parent.feat.mod_type = VERB_MOD_RELATIVE
            # ex. : dans lequel
            elif parent and parent.is_preposition :
                grandparent = s.get_parent(parent)
                if grandparent and grandparent.is_verb :
                    grandparent.feat.mod_type = VERB_MOD_RELATIVE
```

FIG. 5.3 : Une « règle » en Python.

troduisent parfois d'autres erreurs (comme dans : *F. Hollande, président et candidat*[15]).

– Enfin, d'autres règles cherchent à séparer en phrases distinctes les verbes principaux (ou « radicaux », selon le vocable des syntacticiens, qui est plus juste) coordonnés. Cela est linguistiquement plus valide (les deux verbes sont « radicaux » et donc chacun doit être une racine), plutôt que de faire dépendre le second verbe du premier (comme le font les analyseurs[16]). Cela facilite la recherche des antécédents.

Ces règles utilisent généralement les relations de dépendance, c'est-à-dire qu'elles « parcourent » l'arbre syntaxique. Nous avons tenté d'élaborer un formalisme qui permettent d'écrire ce type de règles, mais cela se révèle particulièrement complexe, si bien que le rendement est minimum : l'utilisateur doit apprendre un formalisme complexe pour écrire des règles. Pour cette raison, nous avons décidé de définir les règles qui utilisent les relations de dépendance directement en Python. La figure 5.3 donne un exemple d'une fonction qui ajoute l'indication « relatif » aux verbes des propositions relatives.

15. Si *président* et *candidat* sont étiquetés comme noms, alors nous aurons trois noms coordonnées. L'exemple est inventé, bien sûr.

16. Certains divisent les phrases comme nous le faisons : c'est le cas de Fips (WEHRLI 2007), mais pas de Talismane.

Cependant, quand les relations de dépendance ne sont pas utilisés, il est possible d'utiliser le formalisme décrit ci-dessus pour les pronoms impersonnels. Le motif suivant permet de récupérer le pronom objet *l'* suivi d'un verbe conjugué à un temps composé ; l'analyse du genre du participe passé permet de donner un genre au pronom[17] :

(20) `[form="l'"][lemma="avoir"][pos="VPP"]`

Exemples Prenons un premier exemple, qui illustrera la façon dont la sortie de Talismane peut être corrigée. Soit la phrase :

(21) Le livre que Pierre a lu l'a intéressé.

Talismane donne la sortie suivante :

```
1 Le le DET DET n=s|g=m| 2 det 2 det
2 livre livre NC NC n=s| 0 _ 0 _
3 que que CS CS  2 dep 2 dep
4 Pierre Pierre NPP NPP _ 6 suj 6 suj
5 a avoir V V n=s|t=P|p=3| 6 aux_tps 6 aux_tps
6 lu lire VPP VPP n=s|g=m|t=K| 3 sub 3 sub
7 l' l' CLO CLO n=s|p=3| 9 obj 9 obj
8 a avoir V V n=s|t=P|p=3| 9 aux_tps 9 aux_tps
9 intéressé intéresser VPP VPP n=s|g=m|t=K| 6 mod 6 mod
10 . . PONCT PONCT  9 ponct 9 ponct
```

que l'on peut illustrer avec l'arbre de la figure 5.4[18].

Cette sortie a beaucoup d'erreurs : la racine de l'arbre est un nom, et non le verbe *a intéressé*, verbe principal qui est mis en modifieur du verbe relatif *a lu*. En outre, *que* est considéré comme une conjonction, et non comme un pronom relatif. L'analyse de la phrase est donc totalement fausse.

L'application des règles que nous avons définies permettent de corriger l'arbre parfaitement (figure 5.5). D'abord, le programme va détecter que la racine n'est pas un verbe, et qu'il y a justement un verbe qui n'est ni relatif ni subordonné : il va le prendre comme racine. Ensuite, il va détecter que le verbe subordonné est attaché à un nom, et non à un verbe. Or la conjonction est *que*, et comme elle fréquemment confondue

17. Le participe passé conjugué avec l'auxiliaire *avoir* s'accorde en genre et en nombre avec l'objet direct, si celui-ci est placé avant le verbe, selon la règle bien connue.

18. Notre programme calcule automatiquement les arbres de toutes les phrases du texte avant et après correction, dans un but de vérification et de correction des règles.

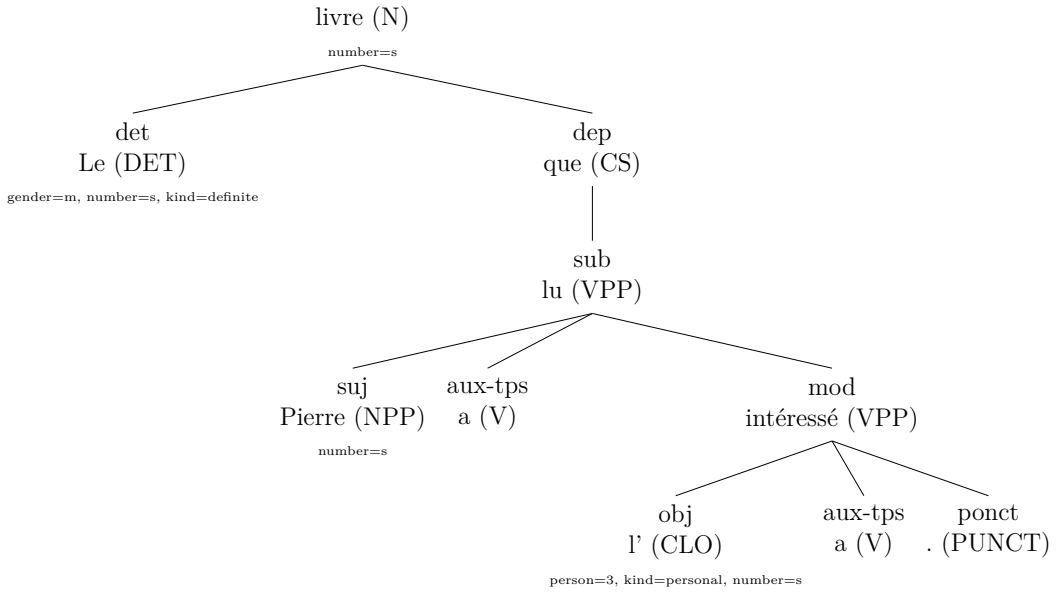

FIG. 5.4 : L'arbre illustrant de la sortie de Talismane.

avec le pronom relatif, le programme va la transformer en pronom relatif, et va inverser les positions du verbe et du pronom. Le résultat est parfait, et permet non seulement une identification de toutes les expressions référentielles, et d'autre part une bonne délimitation de celles-ci (la relative comprise dans l'expression référentielle du *livre*).

On notera aussi que la sortie de Talismane ne donne pas de genre pour *livre* (car le nom peut être masculin ou féminin), mais que l'arbre corrigé indique le genre correct (masculin) : celui-ci a été trouvé grâce au déterminant.

Prenons un autre exemple pour illustrer la formation des groupes. Soit la phrase :

(22) Paul voit le petit chat, le gros chien et la belle girafe.

La sortie de Talismane est illustrée dans l'arbre 5.6. Non seulement cette sortie n'est pas linguistiquement valide (un verbe ne peut pas avoir deux compléments directs), mais en outre le chat, le chien et la girafe ne sont pas réunis en un seul groupe. Cela pose des problèmes pour l'identification des expressions référentielles (nous avons choisi d'annoter le groupe et chacun de ses constituants), mais aussi pour le calcul de la coréférence. En effet, les groupes peuvent être repris, comme dans :

(23) $[[\text{Paul}]_j$ et $[\text{Pierre}]_k]_i$ $[\text{s}]_i$'aiment.

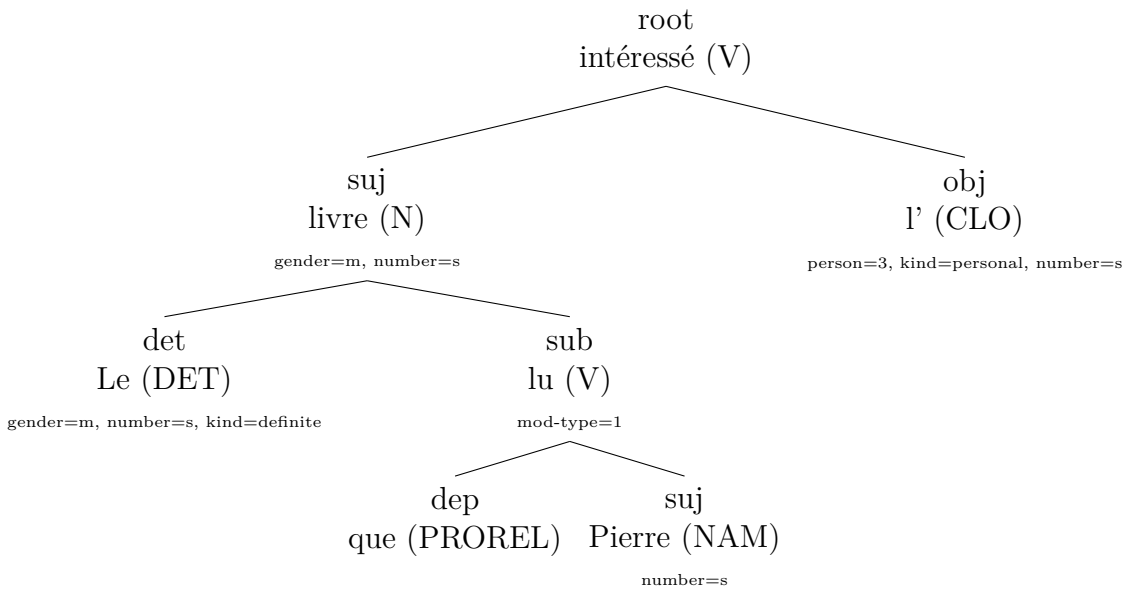

FIG. 5.5 : L'arbre illustrant la correction de la sortie de Talismane. `mod-type=1` indique un verbe relatif. Noter que les auxiliaires ont été supprimés afin de rendre le parcours de l'arbre plus aisé. De plus, nous utilisons un jeu d'étiquette légèrement différent de celui de Talismane, c'est pourquoi le `NPP` de Talismane a été transformé en `NAM`.

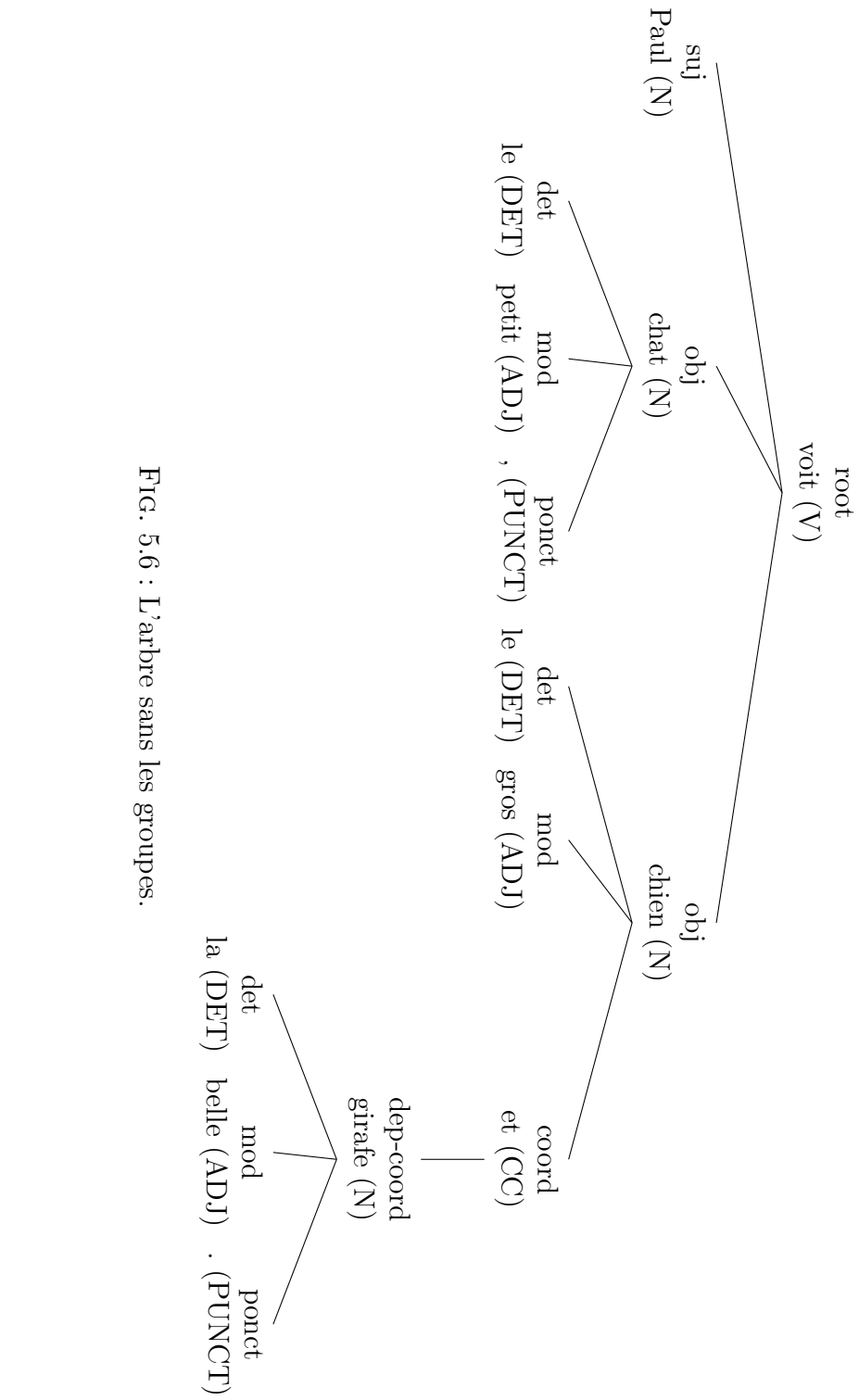

FIG. 5.6 : L'arbre sans les groupes.

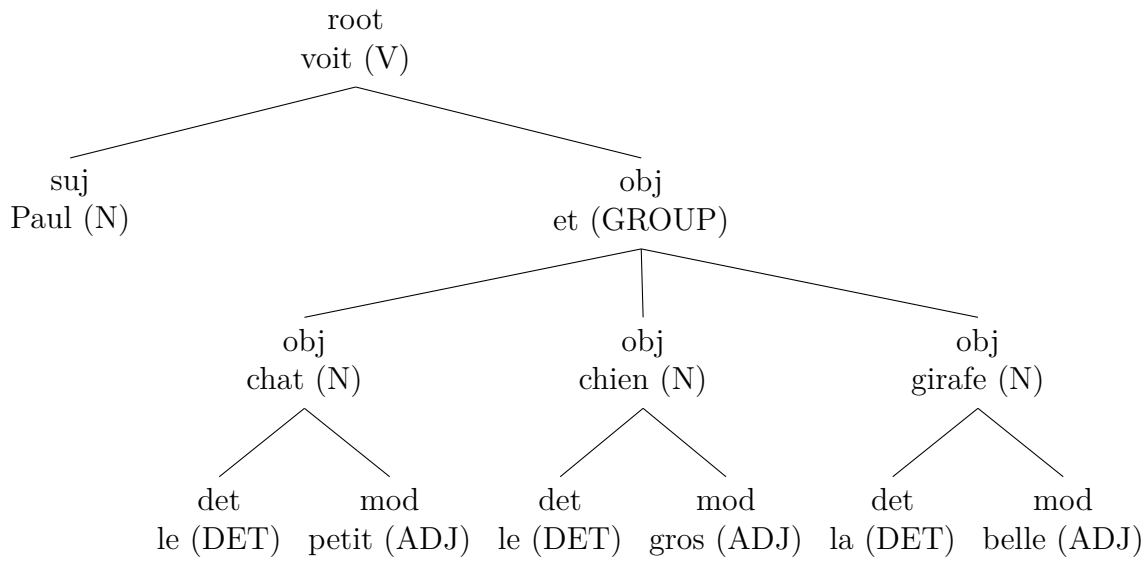

FIG. 5.7 : L'arbre avec les groupes.

La correction est illustrée dans l'arbre 5.7. On voit que les groupes nominaux du complément d'objet ont été réunis sous un seul élément, étiqueté `GROUP`. Cela permet de le considérer comme un syntagme à part entière (il peut donc servir d'antécédent à un pronom et/ou avoir une coréférence avec un autre nom) et de correctement délimiter l'expression référentielle qu'il forme.

Pour finir, nous proposons un exemple de découpe d'une coordination (figure 5.8) de deux verbes en deux phrases distinctes (figure 5.9), avec la phrase :

(24) Pierre mange des pommes et sirote de l'eau.

Cela est utile pour noter les sujets zéros, ce qui est le cas dans certain manuel d'annotation, comme celui de Democrat. Cela permet aussi de faciliter la recherche d'antécédents.

Genre et nombre

Enfin, certaines règles tentent de déterminer le genre ou le nombre d'un token épicène. Ce peut être fait à partir du participe passé. Dans :

(25) Je *l*'ai assumée

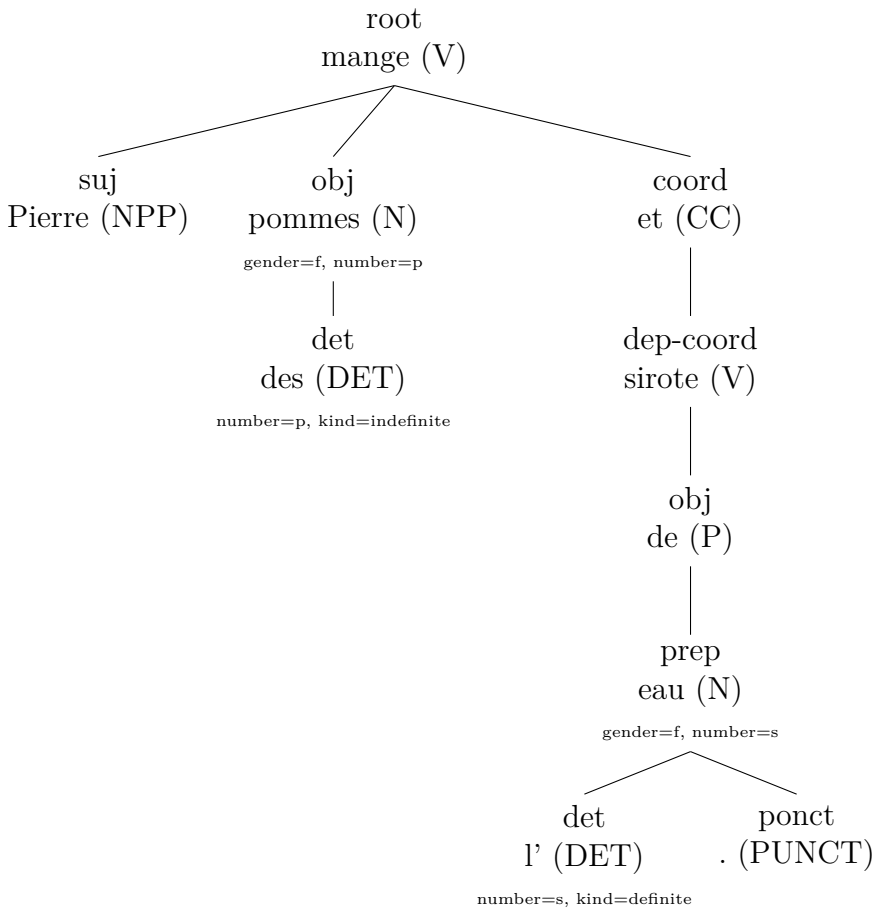

FIG. 5.8 : L'arbre illustrant la sortie de Talismane.

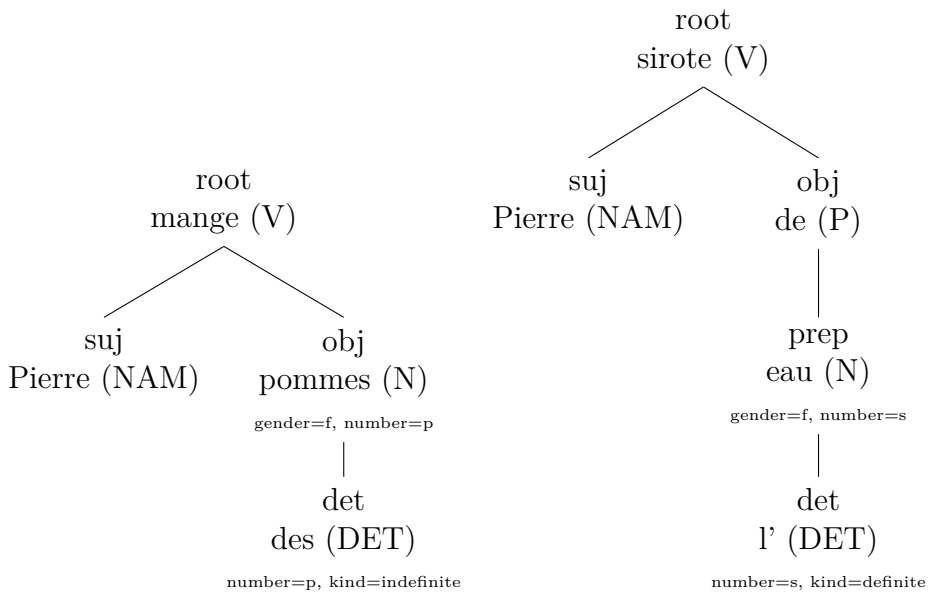

FIG. 5.9 : Les deux arbres résultant de la segmentation de la coordination, avec sujet zéro qui est ici répété à l'identique (*Pierre*).

on sait que que le pronom objet est féminin en regardant la terminaison du participe passé. On sait donc que l'antécédent doit être féminin, et non masculin.

Ce peut aussi être fait à partir du déterminant ou d'adjectif non épicène, comme dans :

(26) a. Le géographe.

b. La géographe.

Les analyseurs syntaxiques s'efforcent parfois de deviner le genre d'un nom épicène, mais sans regarder le déterminant. Ces règles de correction vont donc écraser ce que l'analyseur aura deviné statistiquement.

Enfin, le genre et le nombre peuvent aussi être déterminés grâce à l'attribut du sujet. Dans :

(27) Ce n'est pas un beau vélo.

le pronom démonstratif *ce* est de genre masculin. Cependant, le genre ne correspond pas nécessairement à celui de l'antécédent :

(28) Le véhicule... Ce n'est pas une belle voiture.

5.5 Bilan

Nous avons décrit ici deux modules de notre programme : le tokéniseur et la correction de la sortie de l'analyseur syntaxique. Un important travail est déjà effectué lors de la tokénisation afin de repérer les unités polylexicales et les entités nommées, ainsi que de pré-annoter le texte (discours direct, par exemple) et de le simplifier en vue d'un meilleur traitement par l'analyseur syntaxique. La correction de la sortie de celui-ci permet d'abord de corriger ses erreurs les plus grossières, ensuite de modifier la structure des arbres de dépendance (séparation des phrases coordonnées), enfin d'ajouter des éléments nouveaux (groupes).

La tâche est cependant difficile et nos règles (tant au niveau de la tokénisation que de la correction) sont pour l'heure surtout exploratoires. Il conviendrait de mieux analyser les sorties de l'analyseur (d'un point de vue qualitatif et quantitatif) afin de les améliorer.

Chapitre 6

Algorithme de détection des chaînes

6.1 Rappel des règles de *RefGen*

Pour introduire cette section, nous aimerions rappeler l'algorithme utilisé par RefGen (Longo 2013), que nous avons présenté à la section 0.2.2, puisqu'il s'agit du seul système à base de règles linguistiques opérationnel pour le français.

CalcRef, le module de résolution de la coréférence de RefGen, commence par chercher les expressions qui peuvent être des *premiers maillons*. Une expression peut être un premier maillon si elle a un score supérieur ou égale à 190 : ce score étant calculé à partir de l'échelle d'accessibilité d'Ariel (1990)[1], de la fonction syntaxique[2] et du genre textuel d'occurrence[3]. Concrètement, chaque expression référentielle reçoit d'abord un « score d'accessibilité » en fonction de sa nature (un groupe nominal indéfini reçoit un score de 230, un nom propre complet avec un modifieur reçoit un score de 220, etc. jusqu'au déterminant possessif qui reçoit un score de 130). Des points sont ensuite ajoutés à ce score initial selon la fonction grammaticale de l'expression (100 points pour le sujet, 50 points pour l'objet direct, etc.). Enfin, certains types d'expressions peuvent recevoir 50 points en plus selon le genre textuel d'occurrence (par exemple, pour le genre de « l'analyse politique », les noms propres reçoivent 50 points supplémentaires, car dans ces textes les noms propres sont souvent en début de chaîne).

1. Pour faire bref : plus une expression complète (précise), plus elle a de chance d'être un « premier maillon ».

2. Le sujet, par exemple, obtient un score plus élevé que l'objet.

3. Par exemple, dans le genre de « l'analyse politique », comme le dit Longo, les premiers maillons sont le plus souvent des noms propres : ces derniers voient donc leur score augmenter.

À partir de cette listes de « premiers maillons » établis, le système cherche toutes les expressions qui ont besoin d'un antécédent (c'est-à-dire d'un premier maillon). Ce sont les expressions qui ont un score inférieur à 190[4]. Pour tester les paires antécédent candidat–anaphore, un jeu de contraintes est utilisé :

- contraintes fortes (éliminatoires) :
  - pas d'imbrication des expressions coréférentielles[5] (mais cela n'est possible que parce que RefGen n'annote pas le syntagme, mais seulement une partie de celui-ci, comme dans *[le véhicule]$_i$ qu'il a acheté* (sans le relatif), plutôt que *[le véhicule [qu]$_i$'il a acheté]$_i$*,
  - les têtes lexicales doivent être les mêmes (*Le jeune homme… cet homme*), mais cela interdit les anaphores infidèles ;
- contraintes faibles :
  - compatibilité de genre et de nombre,
  - proximité (l'antécédent doit se trouver dans la même phrase ou dans la phrase précédente). Cette contrainte est utilisée pour les possessifs,
  - même fonction syntaxique pour l'antécédent candidat et l'anaphore,
  - fréquence de co-occurrence[6]dans un corpus préalablement analysé (mais seulement pour des articles d'informatique),
  - présence sur une liste de noms de fonctions pour les antécédents qui sont des entités nommées de type « personne ».

Le candidat retenu est celui qui valide le plus de contraintes faibles.

À ce stade, il y a des *paires* coréférentielles : les chaînes sont ensuite formées par transitivité : si A et B sont coréférentiels, et B et C le sont aussi, alors A, B et C sont dans la même chaîne.

---

4. De plus, la recherche est limitée dans l'espace : seules sont considérées les expressions dont la distance par rapport à leur antécédent potentiel est inférieure à la distance intermaillonnaire moyenne observée pour le genre d'occurrence.

5. Il peut y avoir imbrication des expressions référentielles, comme dans *[[le frère]$_i$ de [Jacques]$_j$]*, mais pas d'expressions *co*-référentielles.

6. Cela permet à RefGen, par exemple, de désambiguïser le pronom *il* dans la phrase :

(29) Il envoie des messages de publicité.

Il y a en effet deux candidats possibles pour le pronom dans les phrases précédentes : *virus* et *ordinateur*. Or il y a plus souvent co-occurrence entre *virus* et *envoyer* qu'entre *ordinateur* et *envoyer* (Longo 2013), donc l'antécédent de *il* est *virus*.

6.2 Algorithme d'ODACR

L'algorithme que nous proposons est tout à fait différent. Il effectue plusieurs passes, chacune s'occupant d'un type de relations différent.

6.2.1 Un algorithme en plusieurs passes

Après avoir corrigé la sortie de l'analyseur syntaxique (voir section 5), le programme détecte les expressions référentielles : ce sont les noms et les pronoms qui ne sont pas marqués comme non référentiels (du type *il pleut*), mais aussi les groupes (*Pierre et Marie*).

La résolution des anaphores et des coréférences se fait en plusieurs étapes, en fonction de la difficulté de la tâche. Premièrement, le programme résout les « anaphores liées », c'est-à-dire les anaphores qui peuvent se résoudre grâce à la syntaxe. Deuxièmement, le programme cherche à résoudre les autres anaphores, comme les pronoms, à partir de syntaxe *et* de sémantique. Troisièmement, le programme résout la coréférence, à partir de données sémantiques seulement. On passe donc progressivement de la syntaxe à la sémantique.

6.2.2 Les « anaphores liées »

Corblin (1995, p. 157) évoque, sans développer, ces « chaînes liées », caractérisées par le fait que « la construction de la chaîne est exclusivement régie par la syntaxe ». La notion s'appuie sur celle « d'anaphore liée », issue de la syntaxe générative. On dit qu'il y a anaphore liée lorsque « l'antécédent... est choisi en vertu d'un calcul purement syntaxique » (Corblin (1995, p. 157), voir aussi les développements plus théoriques de Corblin (1985b) et Milner (1982)).

Le cas le plus évident est celui des pronoms réfléchis : leur antécédent est toujours le sujet de la proposition dans laquelle ils apparaissent. Il n'y a besoin d'aucune analyse supplémentaire, ce seul fait syntaxique permet de résoudre l'anaphore dans tous les cas.

Nous résolvons aussi à ce stade les pronoms relatifs, qui renvoient toujours au nom dont dépend la proposition relative (éventuellement par l'intermédiaire d'une préposition,

puisque le pronom relatif peut être le régime d'une préposition). Ce n'est pas forcément le nom qui précède le relatif, comme dans :

(30) Le *doyen* de la Faculté des Lettres, *qui* s'appelle Frédéric Chapot, n'est pas le directeur de l'*Institut* de Latin, *qui* est Agnès Arbo-Molinier.

Néanmoins, cela n'est pas un problème qui doit être résolu à ce niveau : c'est une tâche qui incombe à l'analyseur syntaxique.

Par ailleurs, il est assez facile de trouver l'antécédent des pronoms disjoints tels que *moi, je*, ou *lui-même, il* : *moi* coréfère avec *je* et *lui-même* avec *il*.

Nous considérons les appositions nominales (*Le président, E. Macron*) comme des expressions référentielles à part entière, mais, puisqu'elles sont apposées, justement, leur coréférence est très facile à établir.

6.2.3 Les autres anaphores

La résolution des autres anaphores requiert à la fois des connaissances syntaxiques et des connaissances sémantiques. Les connaissances syntaxiques permettent de savoir où chercher l'antécédent, et où ne pas le chercher. Par exemple, il ne peut pas y avoir dans la même rection verbale (pour reprendre le terme de Blanche-Benveniste (1995)) deux fois le même référent :

(31) *$[\text{Il}]_i$ donne un cadeau à $[\text{Pierre}]_i$

sauf dans le cas des réfléchis, qui peuvent avoir la forme *se*, mais une forme disjointe :

(32) $[\text{Il}]_i$ $[\text{se}]_i$ donne un cadeau à $[\text{lui-même}]_i$

La syntaxe permet aussi de détecter les cas de cataphore : le pronom doit être dans une subordonnée en début de phrase et doit coréférer au sujet de la proposition principale (de plus, afin d'écarter les faux positifs, nous limitons les cas de cadratifs au début du paragraphe) :

(33) Quand $[\text{il}]_i$ était petit, $[\text{Paul}]_i$ jouait au foot.

La sémantique que nous utilisons ici est surtout affaire de saillance. Nous calculons en effet pour chaque lemme présent dans le texte un « rang de saillance », qui prend en compte les paramètres suivants :

- le nombre d'occurrences (4),

- le nombre de mots sémantiquement reliés dans le texte, calculé à partir du dictionnaire d'hyperonymes (4),
- la moyenne du « score d'accessibilité » d'Ariel (1990) des différentes occurrences[7] (4),
- une des occurrences est-elle le sujet principal[8] de la première phrase du texte (4) ?
- une des occurrences est-elle l'un des objets principaux de la première phrase du texte (4) ?
- une des occurrences est-elle dans la première phrase du texte (3) ?
- le nombre d'occurrences qui sont sujet principal de la première phrase d'un paragraphe (4),
- le nombre d'occurrences qui sont l'un des objets principaux de la première phrase d'un paragraphe (4),
- le nombre d'occurrences qui sont dans la première phrase d'un paragraphe (3),
- le nombre d'occurrences qui sont sujet principal (3),
- le nombre d'occurrences qui sont l'un des objets principaux (3),
- la moyenne du niveau hiérarchique syntaxique[9] (1),
- le lemme est-il un nom d'humain ou une entité nommée de type « personne » (une option permet d'activer ou non ce paramètre, en fonction du type de texte) (4).

Les lemmes sont triés, c'est-à-dire qu'ils sont comparés deux à deux. Le vainqueur de chaque « duel » est déterminé par une sorte de jeu où chaque candidat remporte des points (ce sont les chiffres entre parenthèses dans la liste précédente) pour chaque paramètre où il est le meilleur. Par exemple si le lemme A a dix occurrences, et que le lemme B n'en a que 8, alors le lemme A remportera un nombre déterminé de points, et ainsi de suite pour chaque paramètre. Ces points sont déterminés par expérimentation et susceptibles d'être modifiés en fonction du corpus/du texte. Certes, l'usage de poids est discutable (Landragin 2004), mais dès que l'on a affaire à plus d'un paramètre, il y a toujours des poids. Ne pas faire usage de poids revient en fait à utiliser des poids équivalents (une moyenne non pondérée est en fait pondérée par des coefficients égaux

7. Il s'agit d'une version très simplifiée : nous ne comptons que quatre niveau : nom propre expansé, nom propre, nom commun expansé, nom commun. Il n'y a pas lieu ici de considéré les pronoms.

8. C'est-à-dire de la proposition radicale.

9. L'occurrence était-elle dans la rection principale d'un verbe (niveau 0), ou bien dans la rection d'un verbe subordonné (niveau 1), ou dans la rection d'un verbe subordonné à un autre verbe subordonné (niveau 2), etc.

à 1). Donc, à moins de n'utiliser qu'un seul paramètre ou de ne pas faire usage de la saillance du tout, l'usage des poids est *de facto* indispensable.

Ce « rang de saillance », calculé pour l'ensemble du texte et pour chaque paragraphe, est l'un des paramètres utilisés pour classer les candidats possibles d'une anaphore. Ces candidats sont cherchés dans la phrase de l'anaphore et dans les phrases précédentes (on s'arrête dès qu'on a trouvé des candidat, la recherche s'effectue au minimum dans la phrase courrante et la phrase précédente), jusqu'au début du paragraphe : s'il n'y a pas de candidat et que le début du paragraphe est atteint, alors, et seulement alors, la recherche continue dans la paragraphe précédent.

Chaque expression référentielle est examinée comme candidat potentiel (c'est-à-dire candidat au statut de candidat). Pour que l'expression accède au statut de candidat, elle doit d'abord pouvoir être un antécédent ; par exemple un pronom réfléchi ne peut pas être antécédent. Ensuite, il y a un certain nombre de configurations éliminatoires de la paire candidat potentiel–anaphore. Par exemple, un pronom de première ou deuxième personne ne peut avoir pour antécédent qu'un pronom de même personne (si on est dans le même niveau de discours direct/indirect) ou un nom d'humain[10] (si on n'est plus dans le même niveau). Ainsi dans :

(34) Le charpentier dit : « Pierre, je suis fatigué »

je ne peut pas coréférer à *Pierre*, car on est dans le même niveau de discours direct/indirect. Autre contrainte : le genre et le nombre. Un pronom doit avoir le même genre et nombre que son antécédent[11]. Un déterminant possessif, lui, s'accorde en genre et en nombre avec le nom tête de son syntagme, mais change de radical en fonction du nombre du possesseur : *son/sa/ses* si le possesseur est singulier, *leur/leurs* s'il est pluriel. Autre contrainte que l'on peut citer en exemple : l'antécédent d'un pronom démonstratif[12] doit être un nom, et doit être le dernier nom dominant (c'est-à-dire qu'il ne peut être dominé (au sens de la grammaire générative) par un nom situé après lui) de la phrase précédente :

(35) ? ? [Le chat]$_i$ mange le pâté que ma voisine a préparé. [Celui-ci]$_i$ est gourmand.

On s'attendrait plutôt à

10. Une option permet de désactiver cela pour les textes où des animaux (voire des objets) parlent, comme les fables.

11. Il existe des exceptions d'autant plus célèbres qu'elles sont rares. Nous les ignorons.

12. Voir par exemple Kleiber (1994) et Charolles (1995).

(36) Le chat mange [le pâté]$_i$ que ma voisine a préparé. [Celui-ci]$_i$ est fait à base de boeuf.

Les paramètres utilisés pour classer les candidats sont les suivants :

- saillance dans le texte[13] (2),
- saillance dans le paragraphe (3),
- nombre d'antécédents pronominaux avant d'arriver à un antécédent nominal (1),
- le candidat est-il le sujet principal, ou l'un des objets principaux de la phrase (2) ?
- le candidat est-il sujet ou objet, mais d'un verbe subordonné (1) ?
- le niveau hiérarchique syntaxique (1),
- candidat et anaphore ont-ils la même fonction (2) ?
- distance en nombre de tokens entre candidat et anaphore (plus elle est réduire, mieux c'est) (2),
- cadidat et anaphore sont-ils de la même catégorie grammaticale (3) ?

Les candidats sont triés, c'est-à-dire qu'ils sont comparés deux à deux, en fonction des points indiqués entre parenthèses. Les points, comme tout à l'heure, sont déterminés par intuition (on fera donc les mêmes remarques que ci-dessus). Mais ici les points sont modulés en fonction du type de l'anaphore. Par exemple, pour un déterminant possessif, la proximité jouera un rôle plus important que pour un pronom personnel sujet.

Le candidat qui arrive le premier est retenu comme antécédent de l'anaphore.

6.2.4 Les entités nommées

À ce stade, tous les liens anaphoriques sont résolus, c'est-à-dire qu'on a toutes les « chaînes-A », pour reprendre l'expression de Corblin (1995), du texte. Il ne reste donc plus qu'à assembler celles qui sont coréférentes.

Le programme commence par assembler tous les « bouts de chaînes » qui sont coréférentiels à la même entité nommée, dont l'identifiant a été trouvé lors de la tokenisation. Il n'y a ici aucun problème.

13. La saillance dans le texte et dans le paragraphe est déterminée par le « rang de saillance » du lemme, dont le calcul a été décrit p. 6.2.3.

6.2.5 La coréférence

Par contre, la construction des autres chaînes est bien plus problématique. Le programme tente d'assembler les groupes nominaux (et leurs anaphores calculées précédemment) au fur et à mesure, ce qui permet de tester la viabilité de la chaîne sur l'ensemble des maillons. Prenons un exemple. Soit le texte suivant :

(37) Mon chat boit du lait à côté de mes chiens. Cet animal est gourmand.

Le premier groupe nominal trouvé est « chat ». Comme c'est le premier, il devient automatiquement une « chaîne » (ici, « chaîne » a le sens d'ensemble de maillons, quel que soit le cardinal de cet ensemble), appelons la la chaîne A. Le deuxième groupe est « du lait ». Il serait possible de le rattacher à la chaîne A, mais il n'y a pas de relation sémantique (hyperonymie) entre « lait » et « chat »[14]. Une nouvelle chaîne est donc créée : la chaîne B. Le groupe suivant est « mes chiens ». Il serait possible de le rattacher aux chaînes déjà créées : « chien » (le lemme) va être comparé aux derniers groupes nominaux de ces chaînes (« chat » et « lait »). Comme il n'y a aucune relation sémantique[15], une nouvelle chaîne est créée (C). Le groupe suivant est « cet animal ». Il est possible de le rattacher aux chaîne A, B ou C. « Animal » va être comparé aux derniers groupes nominaux de ces chaînes : il y a une relation sémantique avec « chat » et « chien ». Ces deux termes candidats vont être triés en fonction des paramètres suivants (selon le même principe que décrit précédemment) :

- saillance dans le texte[16] (2),
- saillance dans le paragraphe (3),
- le candidat est-il le sujet principal, ou l'un des objets principaux de la phrase (2) ?
- le candidat est-il sujet ou objet, mais d'un verbe subordonné (1) ?
- le niveau hiérarchique syntaxique (3),
- candidat et anaphore ont-ils la même fonction (2) ?
- distance en nombre de tokens entre candidat et anaphore (plus elle est réduire, mieux c'est) (2),

14. Mais imaginons un instant qu'il y ait une relation sémantique entre les deux noms : la chaîne {chat–lait} serait testée... et considérée comme impossible car deux expressions coréférentes ne peuvent pas être dans la rection d'un même verbe (sauf pronoms réfléchis).

15. Imaginons encore un instant qu'il y ait une relation sémantique : des chaînes ({chat–chien} et {lait-chien}) seraient créées et testées, mais il y aurait incompatibilité de nombre, en plus de la contrainte citée dans la note précédente.

16. La saillance dans le texte et dans le paragraphe est déterminée par le « rang de saillance » du lemme, dont le calcul a été décrit p. 6.2.3.

- cadidat et anaphore sont-ils de la même catégorie grammaticale (3) ?

En fonction des points accordés pour chaque paramètre, c'est « chat » ou « chien » qui sera classé premier. Mettons que ce soit « chien » pour les besoins de la démonstration. Le programme va donc commencer par attacher « cet animal » à la chaîne C (celle de « mes chiens »). Un certain nombre de contraintes vont être vérifiées, comme le nombre (mais pas le genre, puisqu'on peut avoir *la souris... cet animal...*), les déterminants (par exemple, un indéfini devra être en début de chaîne et ne pourra pas être attaché à une autre chaîne), les modifieurs (les modifieurs *restrictifs*, identifiés par l'absence de virgule, doivent être les mêmes), le fait que deux noms soient dans la même phrase (ce que nous considérons comme impossible : dans *le chat... cet animal...*, les deux noms seront dans des phrases distinctes[17]), etc. Ici, c'est la contrainte du nombre qui est violée : « mes chiens » ne peut pas être rattaché à la chaîne C. Donc le programme va étudier la possibilité de l'attacher à la chaîne de « chat », donc la chaîne A. Ici, aucune des contraintes n'est violée, et la chaîne A sera étendue avec « cet animal ».

Après avoir parcouru toutes les expressions référentielles nominales, toutes les chaînes (singletons, paires ou chaînes) sont formées.

6.3 Conclusion

Notre projet de départ était d'améliorer RefGen. Finalement, nous avons écrit un algorithme tout à fait différent, beaucoup plus complexe, mais aussi plus performant (voir le chapitre 7), bien que plus difficile à paramétrer. L'originalité de notre approche est de trouver les relations entre les expressions référentielles en différentes phases, des relations les plus faciles aux plus complexes. La faiblesse de notre algorithme réside dans l'attribution de points dont la détermination est discutable et discutée, mais inévitable.

17. Nous entendons ici « phrase » au sens grammatical et non typographique : deux « phrases » peuvent être séparées par une simple virgule ou une conjonction de coordination : *Je vais au cinéma et mange du pop-corn* comporte deux « phrases ».

Chapitre 7

Évaluation

7.1 Présentation des principales métriques

L'évaluation des systèmes qui détectent la coréférence automatiquement est complexe, puisqu'il faut tenir compte de plusieurs paramètres. Par exemple, une expression peut être annotée alors qu'elle ne le devrait pas (parce qu'elle n'est pas référentielle, comme les pronoms impersonnels), une expression référentielle peut au contraire avoir été « oubliée », ou bien n'avoir pas été incluse dans la bonne chaîne. Une chaîne, à son tour, peut avoir été scindée en deux, ou même en plusieurs, segments, etc. (voir par exemple Trouilleux 2001 pour une discussion des problèmes de ce genre). C'est pourquoi il existe plusieurs métriques (c'est-à-dire plusieurs façon de mesurer la performance d'un système) qui comparent la production de l'algorithme évalué (c'est ce que nous appelerons dans la suite la « réponse ») avec un corpus de référence annoté à la main (c'est ce qui nous appelrons dans la suite la « clé » (*key* en anglais)). Quatre d'entre elles sont utilisées couramment. Pour notre présentation, nous nous appuierons sur Longo (2013) (voir aussi Lassalle (2015)).

Nous devons cependant d'abord définir les termes « précision », « rappel » et « f-mesure », puisque les métriques courante calculent à la fois la précision et le rappel. La précision est une *mesure de la qualité*. Imaginons une corbeille de fruits, et un système qui doit aller chercher des pommes dans cette corbeille. Plus un système aura une bonne précision, plus il ramènera des fruits qui sont des pommes, *et pas autre chose* ; plus le système ramènera d'autres fruits parmi les pommes, plus sa précision baissera.

Le rappel est une *mesure de la quantité*. C'est l'inverse de la précision : plus le système aura un bon rappel, plus il ramènera de pommes, *sans en oublier dans la corbeille* ; plus le système laissera de pommes dans la corbeille, plus son rappel baissera.

Bonne précision et bon rappel sont affaire de compromis, et un bon système doit être bon dans l'un *et* dans l'autre. Il est en effet facile d'être bon dans l'un *ou* dans l'autre : à la rigueur, un système qui prend tous les fruits de la corbeille aura un rappel maximal (mais une précision nulle), car il n'aura laissé aucune pomme dans la corbeille ; et la situation est inversée pour un système qui ne prendrait aucun fruit de la corbeille.

Avec le rappel et la précision, on a deux chiffres ; la *f-mesure* est une sorte de « moyenne »[1] des deux.

7.1.1 La mesure MUC

MUC (Vilain, Burger, Aberdeen, Connolly et Hirschman 1995), developpé dans le cadre de la *Message Undestanding Conference*, s'attache à l'analyse des *liens* entre mentions et non aux mentions elles-mêmes. Le rappel se définit ainsi :

$$\text{MUC}_\text{r} = \frac{\text{nbr de liens commun à } R \text{ et } K}{\text{nbr de liens dans } K}$$

où R est la réponse du système (*i.e.* le texte annoté par le programme de détection des chaînes) et K l'annotation manuelle (la clé, *key* en anglais). La précision, elle, se définit par :

$$\text{MUC}_\text{p} = \frac{\text{nbr de liens commun à } R \text{ et } K}{\text{nbr de liens dans } R}$$

Cette mesure souffre de deux défauts. D'abord, elle favorise les systèmes qui calculent des chaînes longues : identifier une seule chaîne pour toutes les expressions d'un texte d'obtenir un score meilleur que tous les systèmes existants. Ensuite, elles ne tient compte que des liens de coréférence, et ne dit donc rien sur les singletons.

1. En réalité, il s'agit bien d'une vraie moyenne, mais d'une moyenne *harmonique*, qui est différente de la moyenne *arithmétique* que les enseignants utilisent pour calculer la moyenne (arithmétique, donc) de leurs élèves.

7.1.2 La mesure B^3

Cette mesure (*b-cubed* ou B^3) (Bagga et Baldwin 1998) inclut les singletons. Pour chaque mention m, on définit R_m la chaîne contenant m dans la réponse du système, et K_m la chaîne contenant m dans la clé. Rappel et précision se définissent comme :

$$B^3_r = \frac{\text{nb de mentions communes à } R_m \text{ et } K_m}{\text{nbr de mentions dans } K}$$

$$B^3_p = \frac{\text{nb de mentions communes à } R_m \text{ et } K_m}{\text{nbr de mentions dans } R}$$

Le résultat final est la moyenne des mesures pour toutes les mentions.

7.1.3 La mesure CEAF

La mesure CEAF, pour *Constrained Entity Aligned F-Measure* (Luo 2005) se base non sur les liens de coréférence, ni sur les mentions, mais sur l'entité référentielle. Pour ce faire, elle considère les chaînes dans leur ensemble. L'algorithme cherche, pour chaque chaîne de R, la chaîne de K la plus proche, c'est-à-dire celle qui permet d'avoir le plus grand nombre de maillons en commun. La particularité de cette mesure, c'est qu'une même chaîne de R ne peut être utilisée qu'une seule fois, ce qui n'est pas le cas des deux autres métriques. Cependant, cela signifie que des liens de coréférence corrects seront ignorés : c'est le cas notamment si une chaîne est « scindée » en deux segment par le système ; seul l'un des segments sera considéré, le deuxième ignoré.

7.1.4 La mesure BLANC

Enfin, BLANC, pour *BiLateral Assessment of Noun-phrase Coreference* est la mesure la plus récente (Recasens 2010). Son calcul est plus complexe, puisque les formules diffèrent pour la coréférence et la non-coréférence (*i.e.* les singletons), ce qui permet de tester un système non seulement pour la détection de la coréférence, mais aussi pour la non-coréférence (le système sera pénalisé s'il inclus dans une chaîne une expression qui est en fait un singleton, ce que ne font pas les autres métriques).

7.2 L'évaluation de RefGen

Notre programme doit se positionner par rapport à RefGen, le seul système de détection des chaînes de référence à base de règles linguistiques qui soit opérationnel pour le français à l'heure actuelle. Or l'évaluation de RefGen telle que présentée par Longo (2013) présente deux problèmes majeurs. D'une part, les métriques que nous avons évoquées ont été calculées automatiquement à partir d'un script. Mais Longo a utilisé une version qui souffre d'un bug pour le calcul de trois des quatre métriques. Puisque nous avons obtenu le corpus d'évaluation, avec à la fois les réponses du systèmes et les clés (c'est-à-dire les textes annotés manuellement), nous proposons ci-dessous de recalculer les différentes métriques. Nous avons également quelques doutes sur la pertinence de l'annotation des clés. C'est ce que nous examinerons ensuite.

7.2.1 L'évaluation « déboguée » de RefGen

Longo (2013) a utilisé un script perl nommé `scorer`[2] pour calculer automatiquement les quatre métriques. Mais cette version (1.08) de 2011 souffre d'un bug qui a été plus tard corrigé dans la version (8.01) officielle (elle a été utilisée lors de la campagne d'évaluation CoNLL 2012) de `scorer`[3] (Pradhan, Luo, Recasens, Hovy, V. Ng et Strube 2014).

Puisque nous avons obtenu le corpus d'évaluation (d'environ 2 500 tokens), nous avons simplement repassé les réponses de RefGen et les clés au travers des deux versions. Les chiffres que nous obtenons avec la version 1.08 (tableau 7.1) correspondent à ceux de Longo, aux arrondis près. Ceux obtenus avec la version 8.01 (tableau 7.2) sont différents pour trois métriques : B^3, CEAF et BLANC. Nous ne prendrons en considération ici que la f-mesure (qui synthétise sous forme de moyenne harmonique la précision et le rappel).

Les résultats enregistrent une chute parfois très importante : CEAF « institutionnel » passe ainsi de 81,3 % à 16,1 %. Si l'on fait la moyenne de toutes les métriques et de tous les corpus, on obtient un score « moyen » de 30,7 % au lieu de 55 %.

2. http ://www.lsi.upc.edu/~esapena/downloads/index.php ?id=3
3. https ://github.com/conll/reference-coreference-scorers

corpus	id	MUC	B^3	CEAF	BLANC
littéraire	78.4	57.8	58.9	41.3	59.2
journalistique	53.5	36	69	50.9	63.5
institutionnel	18.6	14.3	81.3	68.8	55.7
moyenne	50.2	36	69.3	53.7	59.5

TAB. 7.1 : Évaluation (f-mesures) de RefGen avec le corpus utilisé par LONGO (2013) et la version 1.08 de `scorer`. « id » signifie « identification des mentions ». Ce tableau correspond au tableau présenté par LONGO (2013, p. 306), exception faite des lignes « journalistique » et « littéraire » que Longo a inversées, ainsi que de la métrique CEAF pour le littéraire (Longo a utilisé une variante « CEAFm » seulement pour ce genre de texte, pas pour les autres).

corpus	id	MUC	B^3	CEAF	BLANC
littéraire	78.4	57.8	43.5	40.4	41.1
journalistique	53.5	36	36.6	33.29	26.3
institutionnel	18.6	14.3	15.6	16.1	7.4
moyenne	50.2	36	31.9	29.9	24.9

TAB. 7.2 : Évaluation (f-mesures) de RefGen avec le corpus utilisé par LONGO (2013) et la version 8.01 de `scorer`. « id » signifie « identification des mentions ».

7.2.2 Le problème de l'annotation des clés

Prenons le texte littéraire du corpus qui a servi à évaluer RefGen, un extrait des *Trois Mousquetaires*, dont voici le premier paragraphe :

(38) Tout le long de la route, [le duc]$_{i}$ se fit mettre au courant par [d'Artagnan]$_{j}$ non pas de tout ce qui s'était passé, mais de ce que [d'Artagnan]$_{k}$ savait. En rapprochant ce qu'[il]$_{i}$ avait entendu sortir de la bouche du jeune homme de ses souvenirs à [lui]$_{i}$, [il]$_{i}$ put donc [se]$_{i}$ faire une idée assez exacte d'une position de la gravité de laquelle, au reste, la lettre de la reine, si courte et si peu explicite qu'elle fût, [lui]$_{i}$ donnait la mesure. Mais ce qui [l]$_{i}$'étonnait surtout, c'est que le cardinal, intéressé comme il l'était à ce que le jeune homme ne mît pas le pied en Angleterre, ne fût point parvenu à l'arrêter en route. Ce fut alors, et sur la manifestation de cet étonnement, que [d'Artagnan]$_{l}$ [lui]$_{i}$ raconta les précautions prises, et comment, grâce au dévouement de [ses]$_{l}$ trois amis qu'[il]$_{l}$ avait éparpillés tout sanglants sur la route, [il]$_{l}$ était arrivé à en être quitte pour le coup d'épée qui avait traversé le billet de la reine, et qu'[il]$_{l}$ avait rendu à [M. de Wardes]$_{m}$ en si terrible monnaie. Tout en écoutant ce récit, fait avec la plus grande simplicité, [le duc]$_{i}$ regardait de temps en temps le jeune homme d'un air étonné, comme s'[il]$_{i}$ n'eût pas pu comprendre que tant de prudence, de courage et de dévouement s'alliât avec un visage qui n'indiquait pas encore vingt ans.

L'annotation nous semble avoir quelques erreurs. Ainsi, les deux premièrs occurrences de « d'Artagnan » sont annotées en singleton, non des chaînes ; et ce sont les seuls singletons annotés. Soit il faut annoter tous les singletons et les paires (« chaînes » de deux maillons), soit aucun. Ensuite, « le jeune homme » n'est pas rattaché à « d'Artagnan », alors que les deux termes sont coréférentiels et que « le jeune homme » n'est pas un nom propre (RefGen recommence les chaînes à chaque nom propre) ; et ce d'autant plus que l'expression « le jeune homme » réapparaît un peu plus loin (pas annotée non plus) et est reprise par « il » (pas annoté non plus). Le « ses » de « ses souvenirs » n'est pas annoté alors qu'il s'agit d'un possessif qui renvoie au duc. Il faut enfin faire une remarque au sujet des pronoms relatifs de la « chaîne » {le coup d'épée—qui—qu'} : certes RefGen ne les annote pas, mais il perd alors des chaînes que pour notre part nous annotons : c'est un point à garder à l'esprit si on veut comparer RefGen et ODACR : le deuxième système annote plus de chaînes que le premier, et donc sa tâche est plus difficile.

Un peu plus loin dans le texte, on trouve :

(39) [Le duc]$_u$ [s]$_u$'approcha de l'autel, s'agenouilla comme eût pu faire un prêtre devant le Christ ; puis [il]$_u$ ouvrit le coffret. « Tenez, [lui]$_v$ dit-[il]$_u$ en tirant du coffre un gros noeud de ruban bleu tout étincelant de diamants ; tenez, voici ces précieux ferrets avec lesquels j'avais fait le serment d'être enterré. La reine me les avait donnés, la reine me les reprend : sa volonté, comme celle de Dieu, soit faite en toutes choses. »

Ici, on se demande pourquoi la chaîne {la reine—la reine—sa} n'est pas annotée. On remarquera aussi que le « je » n'est pas annoté et n'est pas relié avec la chaîne du duc. Là encore, RefGen ne le fait pas, mais notre programme tente de relier les pronoms de première et deuxième personne à des éléments hors du discours direct : c'est aussi une tâche délicate que RefGen évite.

On retrouve des problèmes de ce genre ailleurs dans ce texte et dans les autres textes du corpus d'évaluation (chaîne non annotée, maillon non inclus dans les chaînes). Cela s'explique en partie par le contexte dans lequel ces annotations ont été faites : l'annotation des chaînes de référence n'était à l'époque pas encore fixée, et de nombreuses questions étaient encore en discussion, sans réponse claire. RefGen lui-même a évolué au cours du temps (y compris après la fin de la thèse de Longo[4]). Néanmoins, on aurait pu s'attendre à ce que l'annotation des clés correspondent aux choix d'annotation de RefGen au moment où l'évaluation a été faite. Par exemple, si Longo a choisi de faire recommencer les chaînes à chaque nom propre, il y a, pour RefGen, continuité référentielle entre les noms communs (on ne recommence pas les chaînes à chaque nom commun : sinon, ce ne sont plus des chaînes de référence, mais des « chaînes anaphoriques », comme les appelle Corblin (1995)). Il en va de même pour les possessifs, par exemple. De même, RefGen n'annote pas les singletons : pourquoi alors les laisser dans le texte ?

Ces problèmes d'annotation biaisent le résultat de l'évaluation de RefGen, c'est pourquoi nous avons réannoté le texte littéraire du corpus d'évaluation en respectant les choix d'annotation que Longo évoque (il n'y a pas de description précise) dans sa thèse, notamment : pas d'annotation des relatifs mais annotation des autres pronoms et des déterminants possessifs ; on recommence les chaînes à chaque nom propre[5], mais pas aux noms communs ; on retire tous les singletons et toutes les paires, pour ne laisser que les chaînes. Nous avons aussi éliminé les singletons et les paires (qui ne sont pas des

4. Même si nous prenons comme référence l'évaluation faite par Longo (2013).

5. Il faut remarquer tout de même que dans la sortie de RefGen que nous avons, les deux premiers « d'Artagnan » de l'extrait 38 sont annotés coréférentiellement par RefGen. Mais ce sont les deux seules occurrences de d'Artagnan qui sont annotées ainsi. Peut-être que RefGen considère ces deux premières occurrences comme des noms communs.

chaînes) de la sortie de RefGen, puisque le programme ne le fait pas automatiquement (il semble annoter certains singletons et certaines paires, mais de loin pas tous, sans que l'on comprenne pourquoi).

annotation	id	MUC	B^3	CEAF	BLANC	moyenne
d'origine	78.4	57.8	43.5	40.4	41.1	45.7
refaite	56.3	44.4	26	27.2	23.9	30.3

TAB. 7.3 : Évaluation (f-mesures) de RefGen avec le texte littéraire : avec l'annotation d'origine et l'annotation refaite par nos soins. « id » signifie « identification des mentions ».

Les résultats sont présentés dans les tableau 7.3. On constate donc que les résultats baissent à tous les niveaux, et alors que la moyenne avec l'annotation d'origine (qui, encore une fois, n'est pas le fait de Longo) était de 45.7 % pour ce texte littéraire, elle est plutôt de l'ordre de 30.3 % avec l'annotation refaite.

7.3 Évaluation d'ODACR

7.3.1 Une évaluation préliminaire

Nous présentons ici une *évaluation préliminaire*. En effet, nous n'avons pas encore implémenté un certain nombre de règles, ni défini les points des paramètres, et nous avons noté des bugs que nous n'avons pas encore corrigés. De plus, les ressources linguistiques ne sont pas encore vérifiées et corrigées. Cela signifie que les performances indiquées doivent être considérées *a minima*.

Nous commencerons par présenter les résultats d'ODACR en exposant et commentant les annotations obtenues automatiquement dans deux textes. Puis nous décrirons l'évaluation quantitative, que nous discuterons et mettrons en perspective par rapport à RefGen d'une part, CROC (DÉSOYER, LANDRAGIN, TELLIER, LEFEUVRE et ANTOINE 2015) d'autre part.

7.3.2 Présentation qualitative

Examinons la sortie d'ODACR pour le texte que nous présentions dans l'exemple 2, qui montrait la sortie de RefGen et que nous reprenons dans l'exemple 41 :

(40) [Abdelmalek Benbara, retrouvé mort mercredi dans le coffre de [son]$_i$ véhicule]$_i$, a reçu plusieurs coups de pied ou de poing au visage et trois coups de couteau qui [lui]$_i$ ont été portés alors qu'[il]$_i$ agonisait. L'un de ces coups [lui]$_i$ a brisé le larynx, provoquant [son]$_i$ asphyxie. Les deux autres coups ont touché le cœur et le thorax. Toujours selon les constatations du légiste, la mort de [M. Benbara]$_i$ est intervenue rapidement. La date de [sa]$_i$ mort serait contemporaine à celle de [sa]$_i$ disparition. En effet, [sa]$_i$ barbe n'a pas eu le temps d'apparaître.

(41) [Abdelmalek Benbara]$_i$, retrouvé mort mercredi dans le coffre de [son]$_i$ véhicule, a reçu plusieurs coups de pied ou de poing au visage et trois coups de couteau qui lui ont été portés alors [il]$_i$ agonisait. L'un de ces coups lui a brisé le larynx, provoquant [son]$_i$ asphyxie. Les deux autres coups ont touché le cœur et le thorax. Toujours selon les constatations du légiste, la mort de M. Benbara est intervenue rapidement. La date de sa mort serait contemporaine à celle de sa disparition. En effet, sa barbe n'a pas eu le temps d'apparaître.

Notre texte annote parfaitement la chaîne de Benbara[6], sans oublier aucun maillon et sans inclure d'expressions référentielles en trop. Il détecte bien l'entité nommée, y compris sa reprise avec un titre de civilité (*M.*). On pourrait se demander s'il n'y a pas une chaîne des « coups » que le système n'a pas détecté, mais l'annotation d'une telle chaîne est délicate, même pour un annotateur humain : on ne sait pas trop quel est l'antcédént de *qui* (tous les coups, ou seulement les coups de couteaux ? car s'il agonisait, c'est qu'il avait déjà reçus des coups). Ni ODACR ni RefGen ne détecte en tout cas une telle chaîne.

Prenons un deuxième exemple, lui aussi issu du corpus d'évaluation de RefGen. Nous donnons la sortie d'ODACR en 42 et celle de RefGen en 43.

(42) [Lionel Jospin]$_i$ [se]$_i$ livre en revanche à une longue analyse de [[son]$_i$ échec du 21 avril]$_j$. « [Ma]$_i$ part de responsabilité dans [l'échec]$_j$ existe forcément. [Je]$_i$ l'ai assumée en quittant la vie politique », explique-t-[il]$_i$ d'emblée. « Des erreurs dans la campagne, [j']$_i$en ai commises ». Notamment, « [j']$_i$ai sans doute, comme d'autres, sous-évalué les risques du premier tour, gardant pour plus tard une partie de [mes]$_i$ forces ». Mais [l'ancien Premier ministre]$_i$ voit plutôt le « cœur de l'explication » de [cet échec]$_j$ dans la division de la gauche plurielle : « sans candidat MDC (Mouvement des citoyens, le parti de Jean-Pierre Chevènement) et PRG (parti radical de

6. En incluant dans le premier maillon le participe apposé, comme nous avons choisi de le faire.

gauche), l'issue du premier tour était différente et la face de l'élection présidentielle était changée », affirme [Lionel Jospin]$_i$. [Il]$_i$ [s']$_i$en prend en particulier à la campagne « politiquement destructrice » et « nocive » de Jean-Pierre Chevènement.

(43) [Lionel Jospin]$_i$ se livre en revanche à une longue analyse de [son]$_i$ échec du 21 avril. « Ma part de responsabilité dans l'échec existe forcément. Je l'ai assumée en quittant [la vie]$_j$ politique », explique-t-il d'emblée. « Des erreurs dans la campagne, j'en ai commises ». Notamment, « j'ai sans doute, comme d'autres, sous-évalué les risques du premier tour, gardant pour plus tard une partie de mes forces ». Mais [l'ancien Premier ministre]$_k$ voit plutôt le « cœur de l'explication » de cet échec dans la division de la gauche plurielle : « sans candidat MDC (Mouvement des citoyens, le parti de Jean-Pierre Chevènement) et PRG (parti radical de gauche), l'issue du premier tour était différente et la face de l'élection présidentielle était changée », affirme [Lionel Jospin]$_l$. [Il s']$_l$en prend en particulier à la campagne « politiquement destructrice » et « nocive » de Jean-Pierre Chevènement.

Notre programme a détecté correctement toutes les chaînes. Il faut faire trois remarques. D'abord, notre outil a la capacité de bien détecter les antécédents des pronoms de première (et deuxième) personne : ici, tant *je* que *mon* sont notés comme coréférents à Jospin, ce qui est correct. Ensuite, l'anaphore infidèle *Jospin—l'anciem premier ministre* est bien détectée. Enfin, le système a bien vu que le *l'* de *je l'ai assumée* ne renvoyait pas à l'échec (mais à *la part de responsabilité*). Cela n'est possible que parce que le programme a attribué un genre au pronom (féminin), grâce l'analyse du genre du participe passé.

7.3.3 Évaluation quantitative

La difficulté d'une évaluation quantitative tient dans l'annotation du corpus de référence. Réutiliser des corpus déjà annotés est délicat, puisque généralement les choix d'annotations sont différents. Nous avons donc décidé de réannoter des textes selon les choix que nous avons faits pour ODACR. Nous avons utilisé des textes du corpus d'évaluation de RefGen (un extrait des *Trois Mousquetaires* et des faits divers), et des textes du corpus utilisé pour le projet Allusif. Au total, nous avons à peu près 3 250 tokens (c'est un corpus un peu plus important que celui utilisé pour évaluer RefGen, mais bien moins grand que celui qui a servi à évaluer CROC). Nous avons également utilisé le premier chapitre du *Capitaine Fracasse*, qui a été annoté dans le cadre du pro-

jet Democrat. Mais nous le considérons à part, car nous avons réutilisé telles quelles les annotations de Democrat, qui ne correspondent pas toujours à nos choix d'annotations.

Non seulement il y a quatre métriques, mais on peut donner différents éléments à mesurer à ces métriques. Nous avons décidé d'extraire les annotations sous quatre formes :

- L'ensemble des expressions référentielles et des expressions coréférentielles. Cela inclut donc *tous* les singletons, les paires et les chaînes, donc tous les ensembles de un, deux, trois ou plus éléments. Cela est important car les métriques sont prévues pour mesurer la *coréférence* (deux éléments ou plus) et non les chaînes (trois éléments ou plus). Comme nous l'avons vu plus haut, ces métriques tiennent donc compte des singletons, et il est donc pertinent de prendre en compte ces singletons[7]. D'autant que la plupart des systèmes (dont CROC) sont évalués avec singletons et paires, et non seulement avec les chaînes. Cela leur permet d'avoir de meilleurs résultats, car il est plus facile de détecter une paire (deux éléments) qu'une chaîne (trois éléments ou plus).
- Cependant, RefGen a été évalué avec des chaînes, et non des singletons et des paires[8]. Afin de comparer ODACR à RefGen, il est donc plus pertinent de n'utiliser que les chaînes ; tandis que pour comparer ODACR à CROC, nous utiliserons singletons, paires et chaînes.

Chacun de ces deux types d'annotations a ensuite été comparé aux clés selon deux méthodes :

- avec toute la mention,
- en ne considérant que le premier token de la mention.

En effet, les métriques sont très sensibles aux délimitations des mentions : si les mentions ne correspondent pas parfaitement, alors elles sont considérées comme différentes. Cela peut être un problème si, par exemple, le programme a omis d'inclure un adjectif dans la mension (par exemple, s'il a oublié *rousse* à la fin de *Le petit chat de la voisine rousse*).

Le tableau 7.4 montre les résultats. Nous ne donnons que la f-mesure. Le tableau 7.5 indiquent les moyennes des quatre métriques pour le rappel, la précision et la f-mesure.

7. Au-delà de la satisfaction des métriques, la prise en compte des singletons et des paires est utile pour calculer, par exemple, la densité référentielle.

8. Du moins c'est ce que nous trouvons dans les sorties de RefGen que nous avons.

	id	MUC	B^3	CEAF	BLANC
tout	83.62	58.1	64.53	69.28	53.08
tout (début)	91.09	53.68	68.13	71.64	59.86
chaînes	73.88	60.97	41.19	39.58	43.2
chaînes (début)	75.95	61.47	41.75	39.59	45

TAB. 7.4 : Évaluation (f-mesures) d'ODACR. « Id » signifie « identification des mentions ». « Tout » signifie que singletons, paires et chaînes ont été inclus (sinon seulement les chaînes). « Début » signifie que seul le premier token de chaque mention a été utilisé (sinon toute la mention).

	rappel	précision	f-mesure
tout	63.325	61.597	61.248
tout (début)	57.533	60.263	63.328
chaînes	47.325	46.69	46.235
chaînes (début)	47.693	47.805	46.953

TAB. 7.5 : Évaluation (moyennes) d'ODACR. « Id » signifie « identification des mentions ». « Tout » signifie que singletons, paires et chaînes ont été inclus (sinon seulement les chaînes). « Début » signifie que seul le premier token de chaque mention a été utilisé (sinon toute la mention).

Il faut faire plusieurs remarques.

Détection des expressions référentielles Notre programme détecte bien les expressiosn référentielles, puisqu'il a un taux de 91 %. Par contre, il est moins bon pour les expressions référentielles qui entrent dans des chaînes (seulement 76 %).

Délimitation des expressions référentielles Seules 83 % des mentions sont non seulement bien *détectées* mais aussi bien *délimitées*. L'analyseur syntaxique et les règles de correction semblent donc avoir un peu de mal à regrouper les éléments en fin de mentions (adjectifs, participes, etc.). Cela vient sans doute de notre volonté d'inclure les appositions dans les mentions. Cependant, en ce qui concerne les expressions qui entrent dans ces chaînes, il y a peu de différence (74 % *vs.* 76 %). C'est la raison pour laquelle les moyennes ne présentent pas une grande différence selon que l'on considère la mention dans son ensemble ou seulement le premier token.

Singletons, paires et chaînes Comme on pouvait s'y attendre, notre programme est meilleur quand on mesure singletons, paires et chaînes que quand on ne prend en compte que les chaînes : détecter les relations entre deux éléments est plus facile qu'entre trois éléments ou plus. Dans le premier cas, nous obtenons un résultat de l'ordre de 62 %, dans le second cas, de l'ordre de 46 %.

Rappel et précision Sauf dans quelques applications spécialisées[9], un programme est d'autant meilleur qu'il trouve un équilibre entre précision et rappel[10]. Notre outil réussit bien dans ce domaine, puisqu'il y a au maximum trois points d'écart entre précision et rappel.

Nous avons ensuite évalué notre programme avec le premier chapitre du *Capitaine Fracasse*. Ce texte a été annoté pour le projet Democrat : il a l'avantage d'être long (près de 9 100 tokens), alors que les autres textes utilisés pour l'évaluation sont brefs. Mais les choix de Democrat ne sont pas nos choix d'évaluation ; par exemple, Democrat annote certains attributs, nous en annotons aussi quelques-uns, mais pas les mêmes ! De plus, nous annotons les pronoms réfléchis, pas Democrat. Nous annotons aussi les appositions, ce que ne fait pas Democrat, et nous avons fait d'autres choix pour la délimitation des expressions référentielles. De plus, Democrat ne considère que les chaînes et non les singletons ou les paires (les annnotateurs peuvent *ou non* les annoter).

9. Par exemple en médecine, où l'on cherche souvent à optimiser soit le rappel soit la précision.

10. Un tel programme permet de trouver suffisamment d'informations (c'est le rappel, mesure de la quantité), et des informations pertinentes (c'est la précision, mesure de la qualité).

En ne considérant que les chaînes, et que le premier token de chaque expression référentielle, nous arrivons à une moyenne pour les quatre métriques de 22.67 % (MUC : 26.4 %, B^3 : 17.41 %, CEAF : 27.32 %, BLANC : 19.59 %). Le résultat est faible, mais il faut encore une fois souligner que les choix d'annotation sont différents.

7.3.4 Comparaison avec RefGen et CROC

Nous reprenons dans le tableau 7.6 les chiffres de RefGen et les mettons en regard de ceux d'ODACR (la ligne *chaînes* du tableau 7.4, puisque cela correspond le plus à la sortie de RefGen). (Il faut noter que seul quelques textes sont communs : il ne s'agit donc pas d'une stricte comparaison.)

	id	MUC	B^3	CEAF	BLANC	moyenne
RefGen	50.2	36	31.9	29.9	24.9	30.67
ODACR	73.88	60.97	41.19	39.58	43.2	46.23

TAB. 7.6 : Comparaison de RefGen et ODACR. « Id » signifie « identification des mentions ».

La comparaison est intéressante car RefGen (30.67 %) est un système à base de règles linguistiques, comme le nôtre (46.23 %). Mais nous avons ajouté des ressources linguistiques et nous avons complexifié l'algorithme : cela a permis d'améliorer nettement les résultats.

En ce qui concerne la comparaison avec CROC, il faut d'abord rappeler que si ODACR (tout comme RefGen) détecte et délimite les expressions référentielles en plus de chercher la coréférence, CROC ne fait que la deuxième tâche. Comme nous arrivons à détecter environ 91 % des mentions, il faudrait, pour que la comparaison soit « équitable », baisser les résultats de CROC de 9 %. Le tableau 7.7 donne les résultats pour les deux systèmes (nous utilisons la ligne *tout (début)* du tableau 7.4, car cela correspond le mieux à l'évaluation de CROC). (Il faut noter que l'évaluation de CROC a été fait sur un corpus bien plus important que le nôtre.)

	MUC	B^3	CEAF	BLANC	moyenne
CROC	63.45	83.76	79.14	67.43	73.45
ODACR	53.68	68.13	71.64	59.86	63.328

TAB. 7.7 : Comparaison de CROC et ODACR.

À l'heure actuelle, notre programme (63.328 %) est donc moins performant que CROC (73.45 %). Néanmoins, il accepte du texte tout venant (sans avoir besoin d'annotations préalables) et fonctionne pour l'écrit (alors que CROC est orienté vers l'oral).

7.4 Bilan

Notre programme présente une performance meilleure que celle de RefGen, mais moins bonne que celle de CROC. Les moyennes des ses performances sont comprises entre 46 % et 63 %, selon que l'on ne mesure que le taux de détection des chaînes ou bien l'ensemble des expressions référentielles et de la coréférence.

Nous devons cependant souligner encore une fois qu'il ne s'agit que d'une évaluation préliminaire : un certain nombre de règles n'ont pas encore été implémentées, les points des paramètres n'ont pas encore été déterminés avec précision, et des bugs n'ont pas encore été corrigés. De plus, les ressources linguistiques ne sont pas encore vérifiées et corrigées. Cela signifie que les performances indiquées doivent être considérées *a minima*.

Conclusion

7.5 Apport

Notre objectif était de proposer une amélioration de RefGen (Longo 2013), un programme de détection automatique des chaînes de référence à base de règles linguistiques, notamment par l'ajout de ressources lexicales. Néanmoins, le code source du programme n'étant pas accessible, nous avons été obligé de réécrire tout le programme : cela nous a incité à proposer un nouvel algorithme, plus complexe mais plus performant.

Notre apport s'inscrit dans quatre perspectives différentes. D'abord, nous avons proposé deux ressources linguistiques. La première concerne les entitées nommées, et a consisté à élaborer, à partir de *Yago* (Mahdisoltani, Biega et Suchanek 2014), un dictionnaire d'entitées nommées qui contient des informations grammaticales (comme le genre et le nombre), lexicales (les différentes désignations possibles pour une même entités) et sémantiques (les reprises anaphoriques possibles du type *le Rhin... ce fleuve*). Cette ressource permet aussi bien la détection des entités nommées (6,4 millions de désignations différentes) que la résolution de la coréférence (2,3 millions d'entités). Mais notre travail sur les entités nommées ne s'est pas arrêté là : nous avons aussi considéré un ensemble de règles pour la détection des entités qui ne seraient pas la ressource (notamment les dates et les nombres, mais aussi, et surtout, les noms de personnes).

La deuxième ressource linguistique que nous proposons est un dictionnaire d'hyperonymes extraits principalement à partir des définitions d'un dictionnaire généraliste, le *Glawi* (Hathout et Sajous 2016). Ce dictinnaire, qui contient des hyperonymes pour 160 000 termes, sert, dans le cadre de notre programme, à détecter les anaphores infidèles (du type *le chat... cet animal*). Du *Glawi*, nous avons aussi extraits des informations pour la tokenisation, notamment celles des unités polylexicales, pour lesquelles nous avons également utilisé les informations du *Lefff* (Sagot et Fišer 2008).

Ensuite, nous avons établi un ensemble de règles de correction pour les analyseurs syntaxiques, notamment ceux par apprentissage automatique. Nous avons travaillé avec *Talismane* (Urieli 2013), mais d'autres analyseurs peuvent être utilisés, et les règles que nous avons écrites s'y appliqueront également. Ces règles sont fondées sur des observations linguistiques, et visent à rendre l'arbre « linguistiquement valide », en interdisant, par exemple, que l'antécédent d'un pronom relatif soit un verbe.

Enfin, nous avons élaboré un algorithme de détection des chaînes de référence. Cet algorithme se décomposent en trois passes : la première détecte les anaphores liées, la seconde les anaphores libres, la troisième les coréférences entre noms. Les anaphores liées sont résolues à partir d'informations syntaxiques ; les anaphores libres à partir de syntaxe mais aussi d'informations sur la saillance des éléments dans le texte ; la coréférence à partir d'un dictionnaire d'hyperonymes et de contraintes pesant sur les maillons des chaînes. De plus, notre programme offre des particularités intéressantes, comme la détection des groupes (*Pierre et Marie*), qui peuvent être les antécédents d'une anaphore ; ou bien la relation entre un pronom de première ou deuxième personne dans un discours direct et un antécédent nominal en-dehors du discours direct.

Ces recherches, dont aucune n'avaient été développée de façon approfondie par Longo (2013), nous ont permis de proposer un système de détection des chaînes de référence qui obtient de meilleures performances que RefGen. Nous n'arrivons pas cependant aux performances affichées par CROC (Désoyer, Landragin, Tellier, Lefeuvre et Antoine 2015 ; Désoyer, Landragin, Tellier, Lefeuvre, Antoine et Dinarelli 2016), d'abord parce que ce dernier programme ne réalise qu'une partie des tâches (la détection de la coréférence, mais pas celle des expressions référentielles, qui n'a rien de triviale), mais aussi parce que notre travail est avant tout *une recherche exploratoire*.

7.6 Limites et perspectives

Nous avons exploré de nombreuses pistes et achevé un certains nombres de tâches, mais il reste encore beaucoup à faire. Nous avons notamment terminé le développement de toutes les parties automatiques, que ce soit pour les constructions des ressources lexicales, la correction de l'analyseur syntaxique ou bien la résolution de la coréférence. Mais le temps nous a manqué pour achever les tâches manuelles, les plus chronophages.

En ce qui concerne les entités nommées, il reste à définir les « catégories principales », les catégories qui requièrent un article ; il reste également à vérifier les traductions

des catégories *WordNet* et à extraire les noms de fonctions à partir des infoboxes de *Wikipédia*. Pour les hyperonymes, il faut vérifier manuellement la ressource, et trouver les valeurs les plus pertinentes pour les paramètres d'extractions des données lexicales.

Il faut souligner que ces deux ressources lexicales peuvent être développées et améliorées pour des projets indépendants de ceux de la résolution de la coréférence. En effet, il n'existe pas à l'heure actuelle de telles ressources de qualité (et libres) pour le français.

Pour ce qui est de l'analyse syntaxique, il faudrait reprendre les erreurs des analyseurs de façon plus systématique, afin de réécrire des règles plus efficaces et plus pertinentes. Là encore, cette amélioration pourra servir à d'autres projets que la détection des chaînes de référence, puisque tous les analyseurs syntaxiques à base d'apprentissage automatique (ce sont les plus courants) font des erreurs qu'il est utile de corriger, quelle que soit la tâche pour laquelle on réalise l'analyse syntaxique.

Enfin, si le programme de détection automatique des chaînes lui-même est parfaitement fonctionnel, beaucoup d'éléments ne sont pas implémentés, ou ne le sont que partiellement. C'est le cas, par exemple, de la détection des expressions non référentielles (pronoms impersonnels et locutions verbales), le traitement des groupes (*Pierre et Marie*), des poids (les « points ») des règles pour la détection de la coréférence.

Nous avons donc élaboré un *prototype*, qui montre que les systèmes de détection par règles peuvent être performants et améliorés, mais il faudrait encore beaucoup de travail (nous n'avons eu que quelques mois) pour réaliser un système qui puisse rivaliser avec les meilleurs systèmes *end-to-end* de la littérature (ces systèmes sont pour l'anglais).

Ce travail peut également servir de base à une réflexion sur l'hybridation entre règles et ressources linguistiques d'une part, et apprentissage automatique d'autre part. En effet, la plupart des systèmes développés actuellement sont à base d'apprentissage automatique, c'est-à-dire que ce sont des algorithmes statistiques qui cherchent des régularités un corpus préalablement annoté en chaînes de coréférence. Néanmoins, non seulement ces systèmes font des erreurs qu'il est possible de corriger à l'aide de règles linguistiques (un peu comme nous l'avons fait avec la sortie de *Talismane*), mais de plus, certaines tâches ne peuvent être réalisées qu'à l'aide de ressources linguistiques externes. C'est le cas de la détection des anaphores infidèles, par exemple : aucun système d'apprentissage ne pourra « apprendre » les différentes possibilités de reprises anaphoriques par des synonymes ou des hyperonymes, car aucun corpus ne contiendra toutes les combinaisons possibles en quantité suffisante. Il en va de même pour les entités nommées et leurs reprises pronominales et nominales : aucun système ne pourra « apprendre » toutes les entités nommées, leur sexe ou genre, leur nombre, leur reprises nominales, etc. Notre travail, tant du point du vue des ressources linguistiques produites que des

règles linguistiques établies, pourra donc servir également à l'amélioration de système de détection automatique des chaînes de référence à base d'apprentissage automatique.

Nous commencerons l'année prochaine un travail de recherche doctorale sur l'hybridation et nous explorerons certaines de ces pistes. Nous nous appuierons notamment sur l'analyse des erreurs des systèmes existants (tant à base de règles qu'à base d'apprentissage) pour cibler au mieux les besoins en ressources linguistiques dont nous aurons besoins, et auxquelles les travaux que nous avons effectués dans ce mémoire serviront de base. Cela nous permettra par ailleurs de déterminer les connaissances linguistiques à intégrer à un système d'apprentissage automatique. Notre travail de master servira également de support à la déterminantion de ces connaissances, qui pourront aider l'apprentissage sous forme de règles (en amont ou en aval) ou sous forme de sélection des traits d'entraînement les plus pertinents.

Bibliographie

Abeillé, Anne, François Toussenel et Martine Chéradame (2015). *Corpus arboré pour le français (FTB) : Annotations en constituants (Les syntagmes)*. URL : http://ftb.linguist.univ-paris-diderot.fr/fichiers/public/guide-constit.pdf.

Ariel, Mira (1990). *Accessing Noun Phrase Antecedents*. Routledge.

Bagga, Amit et Breck Baldwin (1998). « Algorithms for scoring coreference chains ». In : *The first international conference on language resources and evaluation workshop on linguistics coreference*. T. 1, p. 563–566.

Blanche-Benveniste, Claire (1995). *Le français parlé. Etudes grammaticales*.

Bohnet, Bernd, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter et Jan Hajič (2013). « Joint morphological and syntactic analysis for richly inflected languages ». In : *Transactions of the Association for Computational Linguistics* 1, p. 415–428.

Capin, Daniéla (2014). « Chaînes de référence dans les textes médiévaux non-narratifs : les Year Books ou l'élaboration d'une écriture juridique ». In : *Langages* 195.3, p. 61–78.

Charolles, Michel (1988). « Les plans d'organisation textuelle : périodes, chaînes, portées et séquences ». In : *Pratiques* 57, p. 3–13.

— (1995). « Comment repêcher les derniers ? Analyse des expressions anaphoriques en *ce dernier* ». In : *Pratiques : linguistique, littérature, didactique* 85, p. 89–113.

Chastain, Charles (1975). « Reference and context ». In : *Language, mind, and knowledge*. Sous la dir. de Keith Gunderson. University of Minnesota Press.

Chomsky, Noam (1981). *Lectures on Government and Binding*. Foris Publications.

Christ, Oliver, Bruno M Schulze, Anja Hofmann et Esther Koenig (1999). *The IMS Corpus Workbench : Corpus Query Processor (CQP) : User's Manual*.

Constant, Matthieu, Isabelle Tellier, Denys Duchier, Yoann Dupont, Anthony Sigogne et Sylvie Billot (2011). « Intégrer des connaissances linguistiques dans

un CRF : application à l'apprentissage d'un segmenteur-étiqueteur du français ». In : *TALN*.

Corblin, Francis (1985a). « Les chaînes de référence : analyse linguistique et traitement automatique ». In : *Intellectica* 1.1, p. 123–143.

— (1985b). « Remarques sur la notion d'anaphore ». In : *Revue québécoise de linguistique*.

— (1987). *Indéfini, défini et démonstratif : constructions linguistiques de la référence*. Droz.

— (1995). *Les formes de reprise dans le discours. Anaphores et chaînes de référence*. Pr. Univ. de Rennes.

Désoyer, Adèle, Frédéric Landragin, Isabelle Tellier, Anaïs Lefeuvre et Jean-Yves Antoine (2015). « Les coréférences à l'oral : une expérience d'apprentissage automatique sur le corpus ANCOR ». In : *Traitement Automatique des Langues* 55.2, p. 97–121.

Désoyer, Adèle, Frédéric Landragin, Isabelle Tellier, Anaïs Lefeuvre, Jean-Yves Antoine et Marco Dinarelli (2016). « Coreference Resolution for French Oral Data : Machine Learning Experiments with ANCOR ». In : *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing'2016)*.

Dupont, Michel (2003). « Une approche cognitive du calcul de la référence ». Thèse de doct. Caen.

Friburger, Nathalie et Denis Maurel (2004). « Finite-state transducer cascades to extract named entities in texts ». In : *Theoretical Computer Science* 313.1, p. 93–104.

Glikman, Julie, Céline Guillot-Barbance et Vanessa Obry (2014). « Les chaînes de référence dans un corpus de textes narratifs médiévaux : traits généraux et facteurs de variation ». In : *Langages* 195.

Haghighi, Aria et Dan Klein (2009). « Simple coreference resolution with rich syntactic and semantic features ». In : *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 3-Volume 3*. Association for Computational Linguistics, p. 1152–1161.

Halliday, Michael Alexander Kirkwood et Ruqaiya Hasan (1976). *Cohesion in english*. Routledge.

Hartrumpf, Sven (2001). « Coreference resolution with syntactico-semantic rules and corpus statistics ». In : *Proceedings of the 2001 workshop on Computational Natural Language Learning*.

Hathout, Nabil et Franck Sajous (2016). « Wiktionnaire's Wikicode GLAWIfied : a Workable French Machine-Readable Dictionary ». In : *Proceedings of the Tenth*

International Conference on Language Resources and Evaluation (LREC 2016). Sous la dir. de Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk et Stelios Piperidis. Portorož, Slovenia : European Language Resources Association (ELRA).

Heiden, Serge, Jean-Philippe Magué et Bénédicte Pincemin (2010). « TXM : Une plateforme logicielle open-source pour la textométrie - conception et développement ». In : *10th International Conference on the Statistical Analysis of Textual Data - JADT 2010*. Sous la dir. de Luca Giuliano Sergio Bolasco Isabella Chiari. T. 2. 3. Rome, Italy : Edizioni Universitarie di Lettere Economia Diritto, p. 1021–1032.

Ion, Radu (2007). « Word sense disambiguation methods applied to English and Romanian ». Thèse de doct. Romanian Academy, Bucharest.

Kleiber, Georges (1994). *Anaphores et pronoms*. Duculot.

Landragin, Frédéric (2004). « L'utilisation de scores numériques en sémantique computationnelle ». In : *Journées Scientifiques de Sémantique et Modélisation (JSM'04)*.

— (2005). « Traitement automatique de la saillance ». In : *Actes de la douzième conférence sur le traitement automatique des langues (TALN 2005)*. LIMSI, p. 263–272.

— (2011). « Une procédure d'analyse et d'annotation des chaînes de coréférence dans des textes écrits ». In : *Corpus* 10, p. 61–80.

Lassalle, Emmanuel (2015). « Structured learning with latent trees : A joint approach to coreference resolution ». Thèse de doct. Univeristé Paris Diderot Paris 7.

Lee, Heeyoung, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu et Dan Jurafsky (2011). « Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task ». In : *Proceedings of the Fifteenth Conference on Computational Natural Language Learning : Shared Task*. Association for Computational Linguistics, p. 28–34.

Longo, Laurence (2013). « Vers des moteurs de recherche intelligents : un outil de détection automatique de thèmes. Méthode basée sur l'identification automatique des chaînes de référence ». Thèse de doct. Université de Strasbourg.

Longo, Laurence et Amalia Todirascu (2009). « Une étude de corpus pour la détection automatique de thèmes ». In : *Texte et Corpus* 4, p. 143–155.

— (2014). « Vers une typologie des chaînes de référence dans des textes administratifs et juridiques ». In : *Langages* 195.3, p. 79–98.

Luo, Xiaoqiang (2005). « On coreference resolution performance metrics ». In : *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, p. 25–32.

Mahdisoltani, Farzaneh, Joanna Biega et Fabian Suchanek (2014). « Yago3 : A knowledge base from multilingual wikipedias ». In : *7th Biennial Conference on Innovative Data Systems Research*.

Maurel, Denis, Nathalie Friburger, Jean-Yves Antoine, Iris Eshkol et Damien Nouvel (2011). « Cascades de transducteurs autour de la reconnaissance des entités nommées ». In : *Traitement automatique des langues* 52.1, p. 69–96.

McDonald, Ryan, Kevin Lerman et Fernando Pereira (2006). « Multilingual dependency analysis with a two-stage discriminative parser ». In : *Proceedings of the Tenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, p. 216–220.

Milner, Jean Claude (1982). *Ordres et raisons de langue*. Seuil.

Mitkov, Ruslan (1998). « Robust pronoun resolution with limited knowledge ». In : *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. Volume 2*. Association for Computational Linguistics, p. 869–875.

— (1999). *Anaphora Resolution : The State Of The Art*.

— (2002). *Anaphora resolution*. Routledge.

Muzerelle, Judith, Anaïs Lefeuvre, Jean-Yves Antoine, Emmanuel Schang, Denis Maurel, Jeanne Villaneau et Iris Eshkol (2011). « ANCOR, premier corpus de français parlé d'envergure annoté en coréférence et distribué librement ». In : *TALN'2013, 20e conférence sur le Traitement Automatique des Langues Naturelles*, p. 555–563.

Nadeau, David et Satoshi Sekine (2007). « A survey of named entity recognition and classification ». In : *Lingvisticae Investigationes* 30.1, p. 3–26.

Ng, Vincent (2010). « Supervised noun phrase coreference research : The first fifteen years ». In : *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, p. 1396–1411.

Ng, Vincent et Claire Cardie (2002). « Improving machine learning approaches to coreference resolution ». In : *Proceedings of the 40th annual meeting on association for computational linguistics*.

NHUMA : Les noms d'humains : de la description linguistique aux applications lexicographiques (p.d.). Projet dirigé par Catherine Schnedecker. url : http://nomsdhumains.weebly.com/.

Nouvel, Damien, Jean-Yves Antoine, Nathalie Friburger et Arnaud Soulet (2011). « Recognizing named entities using automatically extracted transduction rules ». In : *Language & Technology Conference (LTC'11)*.

Nouvel, Damien, Maud Ehrmann et Sophie Rosset (2015). *Les entités nommées pour le traitement automatique des langues*. ISTE editions.

Oberlé, Bruno (2016). « Étude des chaînes de référence dans les articles de recherche de format IMRaD : problèmes d'annotation, analyse quantitative et qualitative ». Mém.de mast. Université de Strasbourg.

Petrov, Slav, Leon Barrett, Romain Thibaux et Dan Klein (2006). « Learning accurate, compact, and interpretable tree annotation ». In : *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, p. 433–440.

Pradhan, Sameer, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng et Michael Strube (2014). « Scoring Coreference Partitions of Predicted Mentions : A Reference Implementation ». In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*. Baltimore, Maryland : Association for Computational Linguistics, p. 30–35. url : http://www.aclweb.org/anthology/P14-2006.

Princeton University (p.d.). *About WordNet*. url : http://wordnet.princeton.edu.

Raghunathan, Karthik (2010). « Simple Coreference Resolution with Rich Syntactic and Semantic Features : Is it enough ? » Thèse de doct. Stanford University.

Recasens, Marta (2010). « Coreference : Theory, Annotation, Resolution and Evaluation. » Thèse de doct. Universitat de Barcelona.

Riegel, Martin, Jean-Christophe Pellat et René Rioul (2015). « Grammaire méthodique du français ». In : *Linguistique nouvelle*.

Sagot, Benoît (2010). « The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French ». In : *7th international conference on Language Resources and Evaluation (LREC 2010)*. Valletta, Malta.

Sagot, Benoît et Darja Fišer (2008). *Building a free French wordnet from multilingual resources*. OntoLex.

Sajous, Franck et Nabil Hathout (2015). « GLAWI, a free XML-encoded Machine-Readable Dictionary built from the French Wiktionary ». In : *Proceedings of the of the eLex 2015 conference*. Herstmonceux, England, p. 405–426.

Salmon-Alt, Susanne, Laurent Romary et Jean-Marie Pierrel (2005). *Morphalou 2.0*.

Schmid, Helmut (1994). « Probabilistic part-of-speech tagging using decision trees ». In : *Proceedings of International Conference on New Methods in Language Processing*.

— (1995). « Improvements in part-of-speech tagging with an application to German ». In : *In proceedings of the ACL SIGDAT-Workshop*.

Schnedecker, Catherine (1997). *Nom propre et chaînes de référence*. Metz : Librairie Klincksieck.

— (2005). « Les chaînes de référence dans les portraits journalistiques : éléments de description ». In : *Travaux de linguistique* 51.2, p. 85–133.

— (2011). « La notion de "saillance" : problèmes définitoires et avatars ». In : *Saillance. Aspects linguistiques et communicatifs de la mise en évidence dans un texte. Vol. 1*. Sous la dir. d'O. Inkova. Recherches en Linguistique. Presses Universitaires de Franche Comté, p. 23–43.

— (2014). « Chaînes de référence et variations selon le genre ». In : *Langages* 195.3, p. 23–42.

— (2015). « Les anaphores à nom général humain dans les chaînes de référence renvoyant à des personnes : contraintes d'emploi et rendements ». In : *Travaux de linguistique* 70.1, p. 39–72.

Schnedecker, Catherine et Frédéric Landragin (2014). « Les chaînes de référence : présentation ». In : *Langages* 195.3, p. 3–22.

Schnedecker, Catherine et Laurence Longo (2012). « Impact des genres sur la composition des chaînes de référence : le cas des faits divers ». In : *3ième Congrès Mondial de LInguistique Française*.

Soon, Wee Meng, Hwee Tou Ng et Daniel Chung Yong Lim (2001). « A machine learning approach to coreference resolution of noun phrases ». In : *Computational linguistics* 27.4, p. 521–544.

Trouilleux, François (2001). « Identification des reprises et interprétation automatique des expressions pronominales dans des textes en français ». Thèse de doct. Université Blaise-Pascal de Clermont-Ferrand.

Urieli, Assaf (2013). « Robust French syntax analysis : reconciling statistical methods and linguistic knowledge in the Talismane toolkit ». Thèse de doct. Université de Toulouse II le Mirail.

Urieli, Assaf et Ludovic Tanguy (2013). « L'apport du faisceau dans l'analyse syntaxique en dépendances par transitions : études de cas avec l'analyseur Talismane ». In : *Actes de la 20e conférence sur le Traitement Automatique des Langues Naturelles (TALN'2013)*. Les Sables d'Olonne, France, p. 188–201.

Uryupina, Olga (2006). « Coreference resolution with and without linguistic knowledge ». In : *Proceedings of LREC*, p. 893–898.

— (2007). « Knowledge acquisition for coreference resolution ». Thèse de doct. Universität des Saarlandes.

— (2010). « Corry : A system for coreference resolution ». In : *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, p. 100–103.

VICTORRI, Bernard (2005). « Le calcul de la référence ». In : *Sémantique et traitement automatique des langues*. Sous la dir. de Patrice ENJALBERT. Hermès.

— (2011). *Analec : logiciel d'annotation et d'analyse de corpus écrits*. URL : http://www.lattice.cnrs.fr/-Analec-.

VILAIN, Marc, John BURGER, John ABERDEEN, Dennis CONNOLLY et Lynette HIRSCHMAN (1995). « A model-theoretic coreference scoring scheme ». In : *Proceedings of the 6th conference on Message understanding*. Association for Computational Linguistics, p. 45–52.

WEHRLI, Eric (2007). « Fips, a deep linguistic multilingual parser ». In : *Proceedings of the workshop on deep linguistic processing*. Association for Computational Linguistics, p. 120–127.

WEISSENBACHER, Davy (2008). « Influence des annotations imparfaites sur les systèmes de Traitement Automatique des Langues, un cadre applicatif : la résolution de l'anaphore pronominale ». Thèse de doct. Université Paris-Nord.

WEISSENBACHER, Davy et Adeline NAZARENKO (2007). « Identifier les pronoms anaphoriques et trouver leurs antécédents : l'intérêt de la classification bayésienne ». In : *Traitement Automatique des Langues Naturelles (TALN'07)*. ATALA, p. 145–155.

Table des matières