ANTELOPE - Une plateforme industrielle de traitement linguistique

Francois-Regis Chaumartin

Antelope : une plate-forme industrielle de traitement linguistique

François-Régis Chaumartin *,**

** Société PROXEM (« Procédures Sémantiques »)*
7, impasse Dumur
92110 Clichy

frc@proxem.com

*** Équipe ALPAGE (INRIA)/Lattice (Université Paris 7)*
30, rue du Château-des-Rentiers
75013 Paris

fchaumartin@linguist.jussieu.fr

RÉSUMÉ. *La plate-forme de traitement linguistique Antelope, en partie basée sur la Théorie Sens-Texte (TST), permet l'analyse syntaxique et sémantique de textes sur des corpus de volume important. Antelope intègre plusieurs composants préexistants (pour l'analyse syntaxique) ainsi que des données linguistiques à large couverture provenant de différentes sources. Un effort d'intégration permet néanmoins d'offrir une plate-forme homogène. Notre contribution directe concerne l'ajout de composants d'analyse sémantique et la formalisation d'un modèle linguistique unifié. Cet article présente la plate-forme et la compare à d'autres projets de référence. Il propose un retour d'expérience d'un éditeur de logiciel vers la communauté du TAL, en soulignant les précautions architecturales à prendre pour qu'un tel ensemble complexe reste maintenable.*

ABSTRACT. *The Antelope linguistic platform, inspired by Meaning-Text Theory, targets the syntactic and semantic analysis of texts, and can handle large corpora. Antelope integrates several pre-existing (parsing) components as well as broad-coverage linguistic data originating from various sources. Efforts towards integration of all components nonetheless make for a homogeneous platform. Our direct contribution deals with components for semantic analysis, and the formalization of a unified text analysis model. This paper introduces the platform and compares it with state-of-the-art projects. It offers to the NLP community a feedback from a software company, by underlining the architectural measures that should be taken to ensure that such complex software remains maintainable.*

MOTS-CLÉS : *traitement de corpus, analyse syntaxique et sémantique, lexique sémantique, désambiguïsation lexicale, résolution d'anaphores et de coréférences, Théorie Sens-Texte.*

KEYWORDS: *corpora analysis, parsing, semantic role labeling, semantic lexicon, word sense disambiguation, anaphora and coreference resolution, Meaning-Text Theory.*

1. Introduction

1.1. *Objectifs de la plate-forme*

La plate-forme de traitement linguistique Antelope (*Advanced Natural Language Object-oriented Processing Environment*), développée par la société Proxem depuis 2006, couvre une chaîne complète de traitement du langage. Conçue initialement pour l'anglais, pour des raisons de disponibilité de ressources dans cette langue, elle est en cours d'adaptation au français. Antelope vise à être simple à mettre en œuvre, pour en permettre l'utilisation par un informaticien n'ayant pas de connaissances particulières en linguistique. Pour cela, les principaux composants disposent de paramétrages par défaut, privilégiant un mode de traitement (rapide ou précis). Un utilisateur expert aura en revanche la possibilité de jouer sur des paramètres plus fins, ou d'inclure ses propres algorithmes dédiés à une tâche donnée. Le niveau de traitement le plus complet vise à extraire d'un texte des connaissances sémantiques représentées à l'aide d'un formalisme voisin des graphes conceptuels.

1.2. *Plan de l'article*

La section 2 présente les principes sous-jacents à la plate-forme, inspirés par la Théorie Sens-Texte, qui postule des niveaux de représentation morphologique, syntaxique et sémantique. Elle présente ensuite brièvement les catégories de composants de traitement linguistique permettant d'effectuer les transitions entre niveaux, ainsi que la conception des échanges au sein de la plate-forme. Enfin, elle détaille le modèle informatique que nous avons défini pour unifier les données des différents niveaux de représentation linguistique, et qui autorise à préserver autant que possible les ambiguïtés au niveau lexical, syntaxique et sémantique.

L'architecture technique est abordée dans la section 1, qui présente la conception informatique de la plate-forme, en insistant sur des « bonnes pratiques » de génie logiciel destinées à faciliter la modularité du logiciel et la réutilisation de composants. Cette section expose aussi quelques caractéristiques d'Antelope : capacité de passage à l'échelle, présence d'un mécanisme d'extensibilité au niveau des principaux objets, intégration de composants externes écrits en divers langages.

La section 1 présente le lexique sémantique, composant central de la plate-forme, créé sur la base de WordNet, et enrichi par intégration d'autres ressources à large couverture. Les autres composants créés spécifiquement pour Antelope, notamment pour implémenter l'interface syntaxe-sémantique, sont détaillés en section 5.

La section 6 présente brièvement des plates-formes de référence de traitement du langage, en positionnant Antelope par rapport à celles-ci. Avant de conclure, la section 7 décrit des exemples de réalisations mettant en œuvre Antelope.

2. Principes sous-jacents à la plate-forme

2.1. *Un modèle théorique : la Théorie Sens-Texte*

Notre objectif est de permettre une analyse de textes qui puisse aller jusqu'à une représentation sémantique, sans que ce soit systématiquement une obligation. On peut imaginer, par exemple, qu'un article d'encyclopédie soit analysé finement sur les premiers paragraphes, et plus superficiellement sur la suite. Nous souhaitons donc représenter différents niveaux d'information au sein d'un même document.

Le cadre théorique sur lequel nous nous sommes appuyés est la *Meaning-Text Theory*, ou Théorie Sens-Texte (Mel'čuk, 1988)[1]. Plusieurs implémentations en ont déjà été effectuées[2]. La TST suit la partition classique de la modélisation d'un énoncé en niveaux de représentation phonologique/phonétique, morphologique, syntaxique et sémantique. La spécificité de l'approche Sens-Texte consiste en une subdivision (profond *vs* de surface) des trois premiers niveaux. Sans forcément viser une implémentation complète et orthodoxe de ce modèle[3], nous mettons en œuvre les niveaux de représentation suivants :

– morphologie de surface (nous utiliserons dans la suite l'abréviation RMorphS) et morphologie profonde (RMorphP) : ces niveaux permettent de représenter une information linéaire résultant d'un étiquetage morphosyntaxique ou d'un *chunking* ;

– syntaxe de surface (RSyntS) : cette représentation consiste en un arbre de dépendances syntaxiques, dont les nœuds représentent des lexèmes (pleins ou vides) et les arcs des dépendances syntaxiques de surface, spécifiques à une langue donnée (voire à un analyseur syntaxique particulier) ; pour des raisons de commodité, Antelope permet aussi de la représenter sous forme d'un arbre de constituants ;

– syntaxe profonde (RSyntP) : c'est un arbre de dépendances non linéairement ordonné, dont les nœuds sont des unités lexicales et les arcs des dépendances syntaxiques profondes (universelles) ; les unités lexicales sont désambiguïsées et peuvent être des locutions (des expressions multimots) ; un lexème vide, comme une préposition régime, n'apparaît pas dans la RSyntP ;

– sémantique (RSém) : c'est un graphe dont les nœuds représentent les sens désambiguïsés des unités lexicales et grammaticales et les arcs des relations prédicat-argument (une représentation équivalente peut être donnée dans le formalisme du calcul des prédicats).

La TST propose un modèle bidirectionnel, utilisable en analyse ou en génération. Notre objectif vise, pour l'instant, à extraire des connaissances d'un texte ; nous mettons donc en œuvre des interfaces unidirectionnelles réalisant les transitions

---

[1] Pour une modélisation formelle de la TST, on pourra consulter (Kahane & Mel'čuk, 1998), ainsi que (Kahane, 2002) pour une grammaire d'unification basée sur la TST.

[2] Notamment en milieu industriel, parfois en simplifiant le modèle ; citons par exemple la Cogentex (Lavoie et al., 2000), Lexiquest (Coch, 1998) et VirtuOz.

[3] Nous n'avons, par exemple, pas de niveau phonologique, ni de distinction thème/rhème.

Texte ⇒ RMorphS ⇒ RMorphP ⇒ RSyntS ⇒ RSyntP ⇒ RSém. La contribution spécifique de notre plate-forme porte sur l'interface syntaxe-sémantique, qui effectue les transitions RSyntS ⇒ RSyntP et RSyntP ⇒ RSém, le passage d'un niveau au suivant étant effectué par une interface clairement définie.

2.2. *Composants effectuant les transitions entre niveaux de représentation*

Atteindre une représentation sémantique nécessite la mise en œuvre de plusieurs types de traitements complémentaires. L'implémentation des *transitions* entre niveaux de représentation se fait donc grâce à différents types de composants, comme précisé tableau 1. On y remarquera que certains d'entre eux (résolution d'anaphores, désambiguïsation lexicale) peuvent se retrouver dans plusieurs transitions ; en effet, leur traitement sait s'adapter en fonction du niveau où ils s'appliquent, en donnant un résultat plus ou moins précis.

La section 5 précise le fonctionnement exact des composants utilisés dans Antelope, et la section 3.3 indique la façon dont ils sont intégrés. Certains ont été réalisés spécifiquement pour la plate-forme, d'autres ont une origine extérieure.

Transition	Type de composant de traitement mis en œuvre
Texte ⇒ RMorphS	Segmentation (texte brut ou document HTML) Étiquetage morphosyntaxique Identification des expressions multimots Désambiguïsation lexicale (basique) Résolution d'anaphores et de coréférences (basique)
RMorphS ⇒ RMorphP	*Chunking*
RMorphP ⇒ RSyntS	Analyse syntaxique par dépendances ou en constituants Désambiguïsation syntaxique Désambiguïsation lexicale (intermédiaire)
RSyntS ⇒ RSyntP	Analyse syntaxique profonde Résolution d'anaphores et de coréférences (intermédiaire)
RSyntP ⇒ RSém	Étiquetage des rôles sémantiques Désambiguïsation lexicale (avancée) Résolution d'anaphores et de coréférences (avancée)

Tableau 1. *Composants typiquement utilisés pour implémenter une transition.*

2.3. *Conception des échanges dans la plate-forme*

Une approche répandue dans la fabrication des chaînes de traitement en TAL consiste à exécuter séquentiellement plusieurs programmes, chacun d'entre eux se focalisant sur une tâche particulière. Les données échangées sont spécifiées *via* un format d'entrée et un format de sortie attendus par chaque programme. L'avantage d'un tel pipeline de traitement est la modularité. Ce modèle nous semble toutefois

comporter deux faiblesses. En premier, le lancement d'un nouvel exécutable a un coût fixe en ressources, ce qui rendrait pénalisant la multiplication de tâches courtes (portant sur un mot isolé, par exemple). En second, un programme donné va se spécialiser sur une tâche particulière ; il est toujours possible d'élargir le champ de son traitement, mais cela se fait en complexifiant les formats d'entrée et de sortie.

Pour faciliter l'intégration dans Antelope de différents types de composants de traitement, nous avons choisi une approche alternative à celle du pipeline, de façon à favoriser la coopération entre composants de traitement linguistique. Un modèle de données unifié regroupe les différents niveaux de représentation de la TST. Les composants de traitement peuvent alors interagir facilement avec ces données.

2.4. *Modèle informatique unifié des niveaux de représentation linguistique*

Tous les objets linguistiques résultant de l'application d'un traitement (phrase, mot, dépendance, sens de mot, rôle syntaxique, rôle sémantique, anaphore…) sont définis dans un modèle de données unifié, décrit à la figure 1, qui correspond à l'union de l'ensemble des formats d'entrée et de sortie des composants utilisés.

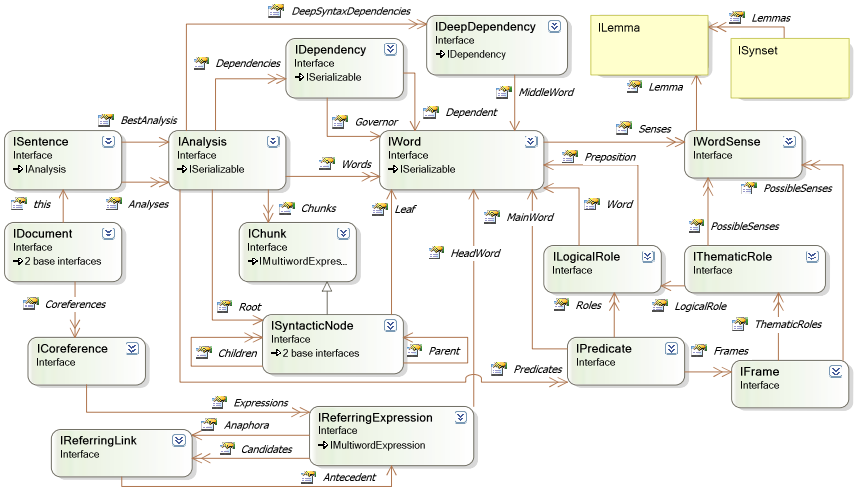

Figure 1. *Le modèle informatique unifié défini pour Antelope*

Le formalisme graphique utilisé ici est proche d'UML. Un rectangle arrondi représente une classe[4]. Les relations entre classes sont matérialisées par des flèches ; une flèche simple représente une référence vers un seul objet (ou une référence nulle) ; une flèche double symbolise un lien vers une liste d'objets. Pour des raisons

[4] En fait, une interface de programmation, l'équivalent d'une classe abstraite (voir 3.2.1).

d'espace, les attributs et méthodes des classes ne sont pas affichés ici. Les classes Lemma et Synset sont affichées différemment pour marquer leur appartenance au lexique sémantique (présenté en section 4.1) et non aux niveaux de représentation.

Nous allons à présent décrire succinctement les classes appartenant à chaque niveau de représentation. En partant du niveau texte, un Document est segmenté en plusieurs phrases (classe Sentence). Antelope cherche à préserver les ambiguïtés aussi longtemps que possible : chaque phrase peut donc être associée à une ou plusieurs analyses (classe Analysis) de niveau morphologique, syntaxique ou sémantique ; l'une de ces analyses (indiquée par la relation *BestAnalysis*) est la meilleure, au sens d'un système de vote qui sera décrit à la section 2.5.

Au niveau RMorphS, une analyse est constituée *a minima* d'une liste de mots (Word) dont la forme de base et la partie du discours sont connues. Le niveau RMorphP se compose de syntagmes (classe Chunk) qui regroupent des mots.

Le niveau RSyntS est constitué de dépendances entre un mot gouverneur et un mot dépendant (classe Dependency) et/ou du nœud racine d'un arbre syntaxique (la classe SyntacticNode hérite de Chunk et lui ajoute une relation récursive).

Les dépendances syntaxiques profondes (classe DeepDependency) du niveau RSyntP sont regroupées par gouverneur pour former des prédicats (classe Predicate) dont les arguments sont les fonctions syntaxiques profondes (classe LogicalRole).

Le niveau RSém est constitué de trois catégories d'informations :

– chaque prédicat du niveau RSyntP est associé à une ou (éventuellement) plusieurs « interprétations sémantiques » (classe Frame) qui précisent les rôles thématiques (classe ThematicRole) des arguments du prédicat ;

– chaque mot ou expression multimot est associé à une liste de sens possibles (classe WordSense). Un système de score permet de conserver les ambiguïtés : un sens possible est donc un lien vers un lemme du lexique sémantique (donc vers un Synset de WordNet), pondéré par ce score ;

– enfin, le document contient une liste de chaînes de coréférences composées d'un ensemble d'expressions référentielles (classe ReferringExpression) ; la classe ReferringLink permet de conserver la liste d'antécédents possibles d'une anaphore.

2.5. *Capacité à préserver les ambiguïtés*

Antelope permet de préserver, autant que possible, l'ambiguïté des différentes unités linguistiques. Ce processus se déroule en trois phases. Au début, un élément

potentiellement polysémique est créé avec ses différents « sens », « analyses » ou « interprétations ». Cela se produit notamment lors des opérations suivantes :

– analyse syntaxique. Elle associe à une phrase ses différents arbres syntaxiques, chacun d'eux étant pondéré par un coût ou une probabilité initiale. Pour éviter une explosion combinatoire, on peut fixer un nombre maximal d'arbres (en pratique, nous avons constaté que l'analyse correcte est souvent dans les dix premiers) ;

– désambiguïsation lexicale. La liste des sens possibles d'un mot est d'abord initialisée à partir des données du lexique sémantique, avec un score à zéro ;

– calcul des anaphores. Chaque pronom (non pléonastique) et groupe nominal déterminé a une liste d'antécédents possibles (avec un accord en genre et nombre) ;

– étiquetage du rôle sémantique des actants d'un prédicat. Par exemple, dans « *Brutus killed Caesar* », Brutus peut être *agent* ou *instrument* de l'événement.

À ce stade, tout élément porteur d'ambiguïté a une liste de « candidats » possibles possédant chacun un score initial ; ce score évolue ensuite, dans le cadre d'un vote effectué par les composants de traitements, qui appliquent successivement différentes heuristiques[6]. Précisons que la plate-forme est livrée en standard avec plusieurs heuristiques. L'utilisateur peut les étendre ou coder ses propres heuristiques ; il peut aussi choisir celles qui seront utilisées pour un traitement donné, en paramétrant leur importance relative et leur ordre d'exécution.

En fin de processus, les candidats ayant le meilleur score sont retenus. Cette approche permet donc de repousser un choix aussi tard que possible. Notons toutefois que pour éviter une multiplication des cas de figure, un mécanisme de vote n'est pas utilisé lors de la segmentation d'un document en phrases et de l'identification d'expressions multimots[7] ; nous formulons comme hypothèse simplificatrice que, dans ces cas, une décision locale suffit le plus souvent à lever l'ambiguïté. Un événement est alors déclenché pour prévenir le code de l'utilisateur qu'un choix est imminent ; le code de l'utilisateur peut alors valider le choix proposé, ou définir une heuristique adaptée à son contexte.

3. Architecture technique

3.1. *Environnement de développement .NET*

Antelope est développée nativement en C#, un langage objet créé par Microsoft dans le cadre de son architecture .NET. L'ensemble constitué par C#, le *framework*

[6] (Carré *et al.*, 1991) définit une heuristique comme « une règle qu'on a intérêt à utiliser en général, parce qu'on sait qu'elle conduit souvent à la solution, bien qu'on n'ait aucune certitude sur sa validité dans tous les cas ».

[7] Cela permet, par exemple, de choisir de relier ou non un verbe à sa particule (« *go up* »).

de classes et la machine virtuelle .NET, est proche de ce qui est proposé par Java[8]. Antelope fonctionne sous Windows avec .NET, et aussi sous Linux avec MONO[9]. .NET semble moins utilisé que Java, C/C++, PERL ou Python dans la communauté du TAL. On peut toutefois noter que NooJ ou SharpNLP ont aussi fait ce choix.

3.2. *Bonnes pratiques issues du génie logiciel*

Le développement de grands systèmes d'informations, qui nécessite des milliers de jours d'analyse et de codage sous la pression de délais courts, a forcé à structurer des méthodes de travail. La notion de génie logiciel s'est progressivement imposée dans l'industrie informatique, visant à fournir des « bonnes pratiques » sous forme de méthodes de conception (Gamma *et al.*, 1993). Elles sont aujourd'hui adoptées par les organisations ayant besoin de créer des applications de grande taille. Nous présentons ici les bonnes pratiques retenues pour la conception d'Antelope.

3.2.1. *Modèle de programmation par interfaces*

La programmation orientée objet est un paradigme qui a fait ses preuves. Les langages à objets récents (Java, C#...) ont introduit, en plus de la notion de classe[10], une notion explicite d'*interface*, un regroupement logique de propriétés et de méthodes. Une classe peut implémenter une ou plusieurs interfaces ; une interface peut être implémentée par plusieurs classes. Ce modèle de programmation systématise une séparation formelle entre interface et implémentation, favorisant un couplage faible entre composants, et donc une meilleure réutilisation.

Illustrons cette démarche sur un cas pratique. Une opération d'étiquetage morphosyntaxique prend en entrée une chaîne de caractères (éventuellement déjà découpée en mots) et associe en sortie une étiquette à chaque mot. On peut donc définir une interface `ITagger`[11] avec une méthode C# :

```
List<IEtiquette> Tag(string texte);
```

Plusieurs implémentations sont évidemment possibles. On codera alors autant de composants que d'implémentations souhaitées (`HmmTagger`, `SvmTagger`…), chacun de ces composants supportant l'interface `ITagger`. Le reste de l'application ne manipulera ces composants qu'en tant que `ITagger`, sans connaître leur implémentation particulière (sauf lors de l'instanciation). L'application cliente du composant peut alors très facilement substituer une implémentation à une autre.

---

[8] Langage objet moderne (permettant le développement par classes et interfaces ainsi que la gestion des erreurs par exceptions) tournant sur une machine virtuelle avec ramasse-miettes. La dernière version introduit des éléments de programmation fonctionnelle (λ-expressions).
[9] MONO (http://www.mono-project.com) est un portage libre de .NET sur Linux et Mac OS.
[10] En C++, une interface peut se définir comme une classe abstraite virtuelle pure.
[11] Par convention, le nom d'une interface commence par la lettre `I` en majuscule.

3.2.2. *Définition de tests unitaires et de tests de non-régression*

Un test unitaire est destiné à s'assurer du fonctionnement correct d'un composant d'un logiciel indépendamment du reste du programme, afin de vérifier qu'il répond aux spécifications prévues. Après une modification substantielle du code, ou un changement de version d'un composant externe, l'ensemble des tests unitaires peut être rejoué afin de rechercher d'éventuelles régressions.

3.2.3. *Conception technique permettant le passage à l'échelle*

La capacité à « monter en charge » est un point essentiel pour traiter des corpus importants. L'environnement .NET dispose de nombreux protocoles de communication intermachines[12], qui permettent de distribuer facilement des traitements, et d'infrastructures de grilles de calcul massivement parallèle[13].

La course à la puissance des processeurs passe aujourd'hui davantage par la multiplication des « cœurs » sur un processeur que par l'augmentation de leur fréquence. Pour en tirer parti, et ne pas sous-exploiter les architectures matérielles actuelles, il faut distribuer les calculs sur ces différents cœurs, en lançant des tâches multiples (*multi-threading*) au sein d'un même processus. Ce type de développement est au centre des systèmes d'exploitation et des serveurs de bases de données. La conception de tels systèmes est connue depuis longtemps pour être complexe (Dijkstra, 1965). De plus, les outils qui aident à détecter un risque d'interblocage sont rares ; une erreur nouvelle risque toujours d'apparaître à l'exécution dans une configuration qui n'aura jamais été rencontrée lors des tests.

Les composants spécifiquement écrits pour Antelope ont été conçus pour s'exécuter dans un environnement multitâche. Les composants externes sont encapsulés avec un mécanisme qui garantit l'invocation séquentielle des méthodes. Nous avons pu, par exemple, effectuer une analyse syntaxique complète de la *Simple Wikipedia*[14] en trois jours. Pour cela, nous avons distribué Antelope sur cinq clients bicœurs ; ils communiquaient (*via* des *Web services*) avec un serveur chargé de leur affecter des analyses de phrases, et de consolider et stocker les résultats.

3.2.4. *Sauvegarde et restauration du résultat d'une analyse*

De même qu'un logiciel bureautique permet d'enregistrer un document sur disque puis de le rouvrir, Antelope peut sauvegarder le résultat de l'analyse d'un texte dans un fichier (typiquement à un format XML), puis le recharger en mémoire. Ce mécanisme autorise le transport d'un résultat d'analyse entre machines, donc la répartition des traitements, ainsi que leur interruption et reprise.

---

[12] Web services, XML sur TCP, *.NET remoting* (flux binaire sur HTTP ou TCP).
[13] Par exemple www.alchemi.net ou www.digipede.net (Antelope ne les exploite pas encore).
[14] http://simple.wikipedia.org (comporte ~28 000 articles écrits en « anglais simplifié »).

3.2.5. *Présence d'un mécanisme d'extensibilité par annotations*

Les annotations sont un modèle très fréquemment utilisé, offrant un mécanisme d'enrichissement de l'information. Dans Antelope, ce modèle est utilisé pour le stockage (par exemple, dans des fichiers XML) et aussi, si nécessaire, pendant des opérations de calcul en mémoire. Il permet de greffer dynamiquement de nouvelles informations à des instances d'objets, sans imposer une modification du code de leur classe ou une spécialisation. Les principaux objets du modèle unifié d'Antelope (document, phrase, analyse syntaxique, mot, coréférence) ainsi que ceux du lexique sémantique (lemme, synset) disposent d'annotations[15]. Elles permettent de stocker :

– des résultats intermédiaires pendant un calcul : par exemple, lors du calcul des anaphores, une annotation précise si un pronom « *it* » est pléonastique ou non ;

– des données complémentaires optionnelles : un synset peut ainsi être relié à l'URL de l'article correspondant dans la Wikipédia ou à des données affectives.

3.3. *Intégration de composants externes*

Antelope intègre plusieurs composants externes, codés en différents langages :

– l'étiqueteur morphosyntaxique *SSTagger*[16] (composant C++) ;

– un étiqueteur morphosyntaxique « à la Brill » (réécrit nativement en C#) ;

– deux analyseurs syntaxiques robustes pour la langue anglaise, qui traitent avec succès des phrases complexes, et tolèrent la présence de mots inconnus :

- le *Stanford Parser* (Manning et Klein, 2002) est un analyseur probabiliste écrit en Java, fourni avec plusieurs grammaires (allemande, chinoise, arabe). Il produit une forêt d'arbres de constituants, et sait les traduire en arbres de dépendances orientées,

- le *Link Grammar Parser* (Sleator et Temperley, 1991), codé en C, repose sur des règles. Il produit un ou plusieurs graphes orientés acycliques de dépendances (liens typés reliant des paires de mots), puis les transforme en arbres de constituants ;

– un analyseur syntaxique du français : le *TagParser* (Francopoulo, 2008), codé en Java, qui produit comme analyse un arbre de dépendances entre constituants.

Antelope utilisant systématiquement le modèle de programmation par interface, ces composants externes sont encapsulés par une interface `ITagger` ou `IParser`.

---

[15] Une annotation est implémentée sous forme d'une structure de dictionnaire, c'est-à-dire une liste de paires (nom, valeur), avec la syntaxe `objet.Annotations["nom"] = valeur`.

[16] Un étiqueteur qui utilise une extension des modèles de Markov avec Maximum d'Entropie. Voir (Tsuruoka, Tsujii, 2005).

3.3.1. *Mécanismes techniques d'intégration*

L'intégration des composants externes se fait de différentes façons :

– si le code source du composant est disponible et écrit dans un langage pour lequel il existe un compilateur pour .NET[17], une recompilation suffit ;

– si le composant est écrit en Java, nous utilisons le transcodeur IKVM pour traduire un `.class` ou `.jar` binaire dans le *bytecode* équivalent de .NET ;

– sinon, pour du code en C ou C++, nous encapsulons le composant en exposant ses services à travers une bibliothèque dynamique (DLL).

3.3.2. *Adaptation à un format commun*

Il est relativement aisé d'intégrer un nouvel étiqueteur morphosyntaxique pour l'anglais. En effet, le jeu d'étiquette du *Penn TreeBank* est un standard de fait, généralement bien suivi (à quelques détails près parfois, comme des étiquettes particulières pour les verbes *to be* et *to have*).

L'effort à fournir est plus important pour les analyseurs syntaxiques. En effet, même s'ils produisent des structures identiques (constituants et dépendances) avec des étiquettes de constituants normalisées (NP, VP, PP…), les étiquettes et l'organisation des dépendances diffèrent radicalement entre différents analyseurs. La figure 2 permet de le constater, en comparant la RSyntS produite par le Link Grammar Parser avec celle du Stanford Parser, lors de l'analyse syntaxique d'une même phrase. Nous verrons en section 5.4 que, lors du passage en RSyntP, nous obtenons un format commun de dépendances syntaxiques profondes.

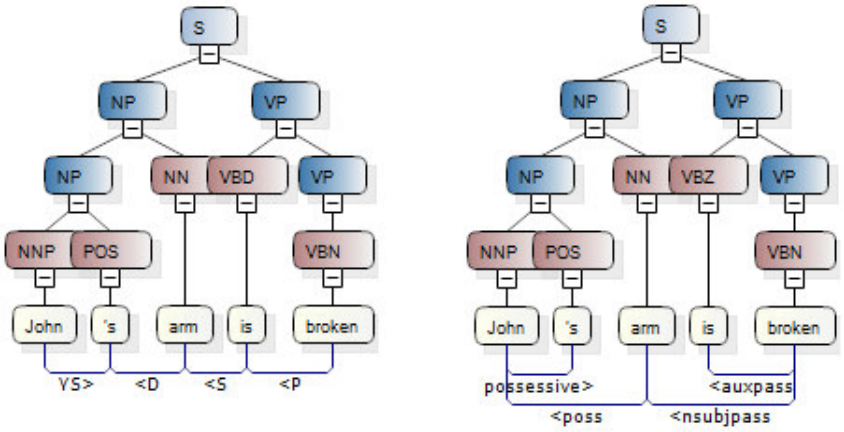

Figure 2. *Sorties du Link Grammar (à gauche) et du Stanford Parser (à droite). L'arbre de constituants est au-dessus des mots, l'arbre de dépendances en-dessous*

[17] VB, Pascal, Python, Eiffel, COBOL, etc. étendus pour offrir des fonctionnalités identiques.

4. Lexique sémantique

4.1. *Intégration de ressources linguistiques à large couverture*

4.1.1. *WordNet*

WordNet (Miller, 1995) version 3.0 constitue la base du lexique sémantique d'Antelope. Ce projet, mené depuis 1985 à Princeton, offre un réseau sémantique très complet de la langue anglaise. WordNet est utilisable librement, y compris pour un usage commercial, ce qui en a favorisé une diffusion très large. S'il n'est pas exempt de critiques (granularité très fine, absence de certaines relations…), il n'en reste pas moins l'une des ressources de TAL les plus populaires. Les nœuds sont constitués par des ensembles de synonymes (ou *synsets*), correspondant au sens d'un ou plusieurs lemmes. Un synset est défini par une glose et surtout d'une façon différentielle par les relations qu'il entretient avec les sens voisins. Par exemple, des relations d'hyperonymie et d'hyponymie relient les « ancêtres » des noms et des verbes avec leurs « spécialisations »[18]. WordNet associe à chaque lemme une fréquence (sur la base de son nombre d'occurrences dans le corpus Brown), et classe les différents sens d'un même mot dans l'ordre décroissant de fréquence.

La figure 3 présente la modélisation de ce lexique sémantique. Un synset est associé à un ou plusieurs lemmes. Chaque lemme est associé à un seul synset. Un synset est en relation sémantique avec d'autres synsets ; de même, un lemme est en relation lexicale avec d'autres lemmes. Comme une opération fréquemment utilisée dans les algorithmes de parcours du graphe est l'énumération des hyperonymes d'un synset, il les présente sous forme de liste ordonnée.

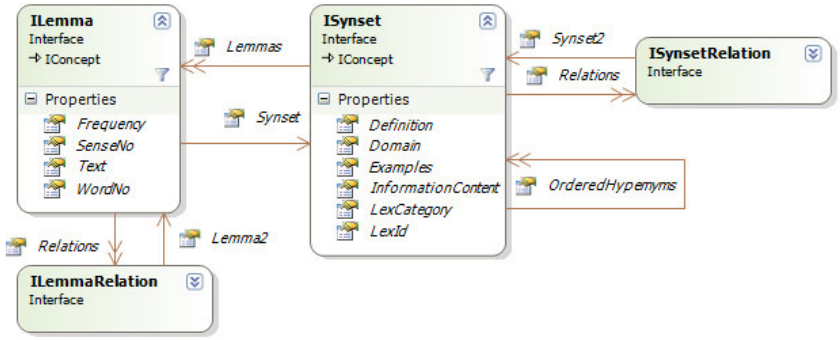

Figure 3. *Modélisation du lexique sémantique*

[18] La version 2.1 a introduit la notion d'*instance hyponyme*, qui distingue une instance (typiquement une entité nommée) d'une sous-classe d'un *synset*. Par exemple, des hyponymes de TOUR sont MINARET et PHARE, et TOUR_EIFFEL en est une instance hyponyme.

4.1.2. *Écosystème de WordNet*

Plusieurs autres ressources linguistiques à large couverture, constituées manuellement ou automatiquement, se rattachent à WordNet. L'ensemble constitue un écosystème complet couvrant des aspects lexicaux, syntaxiques et sémantiques. La figure 4 présente quelques-unes de ces ressources.

Figure 4. *Quelques-unes des ressources disposant d'un lien vers WordNet*

Antelope combine WordNet avec VerbNet, eXtended WordNet, WordNet Domains, WordNet Affect, SentiWordNet et l'ontologie SUMO. L'ensemble constitue un lexique sémantique homogène, utilisé pour la désambiguïsation, la résolution d'anaphores et le passage en forme sémantique.

Notons que l'interopérabilité entre ces six ressources lexico-sémantiques et WordNet est permise par la présence d'un identifiant unique pour chaque synset[19]. Ces ressources fournissent explicitement l'information de traçabilité vers un synset et sont donc homogènes avec WordNet. Leur intégration nécessite un travail d'ingénierie, mais ne soulève pas de difficulté conceptuelle.

En l'absence d'une telle information, l'intégration de la ressource à WordNet se heurte à un problème d'hétérogénéité conceptuelle. Il faut alors établir une correspondance entre chaque entrée d'une telle ressource et un synset ; c'est par exemple ce que nous avons établi dans le cas de la Wikipédia (voir section 7.2).

4.1.3. *Cadres de sous-catégorisation des verbes (VerbNet)*

L'interface syntaxe-sémantique d'Antelope est basée sur VerbNet, une ressource développée par Martha Palmer (Université de Boulder), qui décrit les cadres de sous-catégorisation des verbes anglais. C'est un prolongement des travaux de (Levin, 1993) décrit dans (Kipper-Schuler, 2003). La ressource la plus proche pour les verbes français nous semble être le lexique-grammaire du LADL (Gross, 1994). Les verbes partageant les mêmes comportements syntaxiques et sémantiques sont regroupés en classes décrivant leurs constructions typiques (« *frames* »). Une classe de verbes identifie des rôles thématiques avec d'éventuelles contraintes de sélection. La sémantique de l'action ou de l'événement est également précisée. Des sous-classes permettent de décrire d'éventuelles spécialisations d'une classe. VerbNet version 2.1 distingue 237 classes de verbes qui regroupent 4 991 sens.

Par exemple, la classe de verbe « *murder* » décrit trois constructions typiques :

– Agent élimine Patient (« Brutus tua Jules César ») ;

[19] Cet identifiant change hélas à chaque version de WordNet. Pour importer une ressource donnée, il faut donc connaître la version de référence, et utiliser une table de correspondance. L'Université de Catalogne (www.lsi.upc.es/~nlp) propose des telles tables de correspondance.

– Agent élimine Patient avec Instrument (« Brutus tua César avec un couteau ») ;

– Instrument élimine Patient (« le pesticide tua les insectes »).

Chaque description de classe de verbes déclare des contraintes de sélection sur les rôles thématiques. Dans « *murder* », l'Agent et le Patient doivent avoir un trait *animé* (en pratique, *humain* ou *organisation*) et l'Instrument doit être *concret*. La sémantique fine de l'événement est également précisée :

– au démarrage de l'événement, Patient est vivant : alive(start(E), Patient) ;

– à la fin de l'événement, Patient n'est plus vivant : not(alive(result(E), Patient)).

4.1.4. *Gloses désambiguïsées (eXtended WordNet - WordNet Gloss Corpus)*

L'ensemble des gloses de WordNet (définitions des synsets) constitue un corpus particulier. Disposer d'une analyse syntaxique des gloses et d'une désambiguïsation de leurs termes est une facilité appréciable. Ces informations sont utilisées :

– par des applications qui fabriquent automatiquement des ressources lexicales : citons par exemple la mesure de similarité basée sur les vecteurs conceptuels décrite dans (Schwab, 2006) et l'application de détection de patrons de polysémie évoquée en section 7.4 ;

– par l'interface syntaxe-sémantique pour la vérification des contraintes de sélection : par exemple, comme il n'existe pas de synset hyperonyme des « *objets pointus* », la façon la plus simple de déterminer si un nom désigne un tel objet consiste à examiner sa définition.

Le lexique sémantique d'Antelope propose une telle analyse au format décrit en section 2.4. La première version utilisait eXtended WordNet ; ce projet, mené en 2003 à l'Université de Dallas, enrichit chaque synset de WordNet 2.0 avec une analyse syntaxique de sa glose et une forme logique (Mihalcea *et al.*, 2001). Les mots des définitions ont fait l'objet d'une désambiguïsation lexicale automatique, mais seule une faible partie d'entre eux (1,3 %) a été validée manuelle (qualité *gold*). Le lexique utilise à présent les données du *WordNet Gloss Corpus*. Édité en avril 2008 par l'Université de Princeton, ce projet offre le double avantage de reposer sur WordNet version 3.0 et de disposer d'une validation beaucoup plus couvrante (29 % des mots ont été désambiguïsés manuellement).

4.1.5. *Affectation d'un synset à un ou plusieurs domaines (WordNet Domains)*

WordNet Domains (Magnini et Cavaglià, 2000) est une extension multilingue de WordNet 2.0, développée à l'Instituto Trentino di Cultura (ITC-irst). Elle associe à chaque synset au moins une étiquette de domaine (par exemple *Politique*, *Médecine*, *Économie*…) choisie dans un ensemble de deux cents étiquettes organisées hiérarchiquement. Le domaine *Médecine* regroupe par exemple des noms tels que DOCTOR#1 (le premier sens du nom docteur) et HOSPITAL#1, et des verbes comme OPERATE#7. L'utilisation de WordNet Domains permet d'améliorer l'efficacité d'algorithmes de désambiguïsation et d'expansion de requêtes.

4.1.6. *Ressources pour l'analyse de sentiments (WordNet-Affect - SentiWordNet)*

Basée sur WordNet Domains, WordNet-Affect (Strapparava, Valitutti, 2004) est une ressource linguistique pour la représentation lexicale de connaissances sur les affects ; elle enrichit les synsets en leur associant une étiquette affective. SentiWordNet (Esuli et Sebastiani, 2006) assigne à chaque synset trois valeurs reflétant leur degré d'objectivité et leur connotation positive ou négative.

Ces deux ressources permettent la détection d'affects dans les textes. Elles sont par exemple utilisées par l'application décrite en section 7.3. De tels traitements ont un intérêt économique grandissant : par exemple, une société peut chercher à détecter les critiques positives ou négatives sur ses produits, en analysant la blogosphère ou les dépêches d'agences de presse.

4.1.7. *Ontologie liée à WordNet (SUMO: Suggested Upper Merged Ontology)*

SUMO (Niles et Pease, 2003) est une proposition de standard soumise à l'IEEE pour représenter un « haut » générique d'ontologie, répertoriant d'une façon réutilisable et générique de grandes catégories de la pensée humaine. MILO est un ensemble d'ontologies multidomaines, de niveau intermédiaire, basées sur SUMO. L'ensemble, écrit en une version simplifiée du *Knowledge Interchange Format* (langage logique du premier ordre), compte 20 000 termes et 60 000 axiomes.

4.2. *Mesures de similarité*

Les algorithmes effectuant des traitements sémantiques ont souvent besoin de comparer deux synsets[20]. De nombreux auteurs ont proposé des définitions de mesures de similarité, et plusieurs implémentations basées sur WordNet sont disponibles[21]. Antelope implémente plusieurs mesures de similarité, pour offrir un choix de « points de vue » en fonction des objectifs d'un algorithme. Pour être comparables, les résultats de toutes ces mesures sont dans l'intervalle [0 ; 1]. La mesure est un nombre réel, valant 1 quand les deux synsets sont identiques, et d'autant plus proche de 0 que les synsets sont différents.

[20] Citons par exemple le cas de la résolution d'anaphores nominales.
[21] Par exemple, WordNet::Similarity (http://www.d.umn.edu/~tpederse/similarity.html).

4.2.1. *Mesure de similarité « structurelle » (parcours du graphe d'hyperonymes)*

(Lin, 1998) définit comme mesure de similarité entre deux synsets *s1* et *s2* :

$$sim(s1, s2) = \frac{2.\log P(s)}{\log P(s1) + \log P(s2)} \quad [1]$$

Dans la formule [1], *s* est le synset le plus spécifique subsumant *s1* et *s2* dans la hiérarchie de WordNet, et *P(s)* représente la fréquence du synset *s*. Nous ramenons cette valeur dans l'intervalle [0 ; 1] en prenant la racine carrée de son inverse.

Notre implémentation introduit deux niveaux supplémentaires dans la hiérarchie des verbes. En effet, s'il existe un nom-racine unique, ce n'est pas le cas pour les verbes ; or, la qualité de la mesure de similarité est fonction de la finesse de la hiérarchie. De façon à rendre tous les verbes comparables, nous avons créé un pseudo-synset qui sert de racine commune à tous les verbes, ainsi que des pseudo-synsets regroupant les catégories lexicales (verbes de mouvement, d'état…).

4.2.2. *Mesure de similarité « conceptuelle » (recouvrement des gloses)*

Cette mesure vectorielle est basée sur le recouvrement des mots entre gloses, et utilise une pondération de type TF-IDF[22]. Considérons les deux sens du nom « *samurai* » ; on remarque que les deux définitions ont trois mots en commun :

– SAMURAI#1 : *a **Japanese** warrior member of the feudal **military aristocracy***

– SAMURAI#2 : *feudal **Japanese military aristocracy***

Le premier synset a pour hyperonyme « personne » et le second « groupe » : ils sont donc très distants (similarité égale à 0,04) du point de vue de la mesure de similarité structurelle présentée en sous-section 4.2.1. En revanche, ils sont proches (similarité valant 0,56) du point de vue plus « conceptuel » de cette seconde mesure.

4.3. *Lexiques définis par l'utilisateur*

Antelope permet à l'utilisateur d'enrichir le lexique sémantique de base (correspondant aux données de WordNet) en créant des lexiques spécialisés. Ce mécanisme permet de décrire en XML de nouveaux synsets, lemmes et relations. Deux lexiques de ce type sont livrés avec Antelope, contenant :

– la traduction française de 44 200 lemmes, provenant de WOLF, un WordNet libre du français décrit dans (Sagot et Fišer, 2008) ;

[22] TF-IDF (*term frequency-inverse document frequency*) est une méthode de pondération souvent utilisée dans la fouille de textes. Cette mesure statistique nous permet ici d'évaluer l'importance d'un mot au sein d'une définition. Le poids augmente proportionnellement en fonction du nombre d'occurrences du mot dans la définition. Il varie également en fonction de la fréquence du mot dans le corpus formé par l'ensemble des définitions.

– 300 000 nouveaux synsets, des entités nommées (marques, produits, personnes, lieux…) correspondant à un sous-ensemble de la Wikipédia anglaise.

5. Composants de traitement linguistique

5.1. *Segmentation et traitement de l'enrichissement typographique*

Antelope traite des documents au format texte brut ou au format HTML. Dans ce dernier cas, un traitement préalable sépare le texte de son enrichissement typographique ; l'analyse des balises HTML permet un découpage du document en paragraphes. Ces paragraphes sont ensuite découpés en phrases en utilisant un ensemble de règles. L'information permettant de relier les mots, phrases et paragraphes aux balises est par la suite toujours disponible, ce qui permet :

– de déterminer si un paragraphe est un titre ou un élément d'une énumération[23] ;

– d'identifier les références quand le document contient des liens hypertextes[24].

5.2. *Étiquetage morphosyntaxique,* **chunking** *ou analyse syntaxique*

En fonction de ses besoins de vitesse ou de précision, l'utilisateur peut choisir entre un étiquetage morphosyntaxique, un *chunking* ou une analyse syntaxique. Antelope utilise pour cela les composants externes présentés en section 3.3. Dans les trois cas, le modèle unifié décrit en section 2.4 est alimenté, mais seules les représentations concernées (RMorphS, RMorphP ou RSyntS) sont renseignées.

5.3. *Identification des expressions multimots*

Antelope utilise le lexique sémantique pour identifier les expressions multimots. La forme de base de chaque mot est d'abord calculée par le module morphologique du lexique ; le composant teste la présence de *n*-uplets dans le lexique (*n* variant de cinq jusqu'à deux). Des règles supplémentaires permettent d'identifier aussi des expressions multimots non contiguës, comme dans « *Pierre and Marie Curie* », « *Alabama and Mississippi Rivers* » ou « *the canon and civil law* ».

Lors d'une analyse syntaxique (par opposition à un simple étiquetage morphosyntaxique), une contrainte supplémentaire de rattachement est appliquée. Les mots ne sont regroupés que s'ils appartiennent à un même sous-arbre. Comme le montre la figure 5, l'expression « *Battle of Gettysburg* » est reconnue dans l'analyse syntaxique de gauche, mais pas dans celle de droite (absence de tête commune).

---

[23] Auquel cas, il nécessite peut-être un traitement particulier : « décapitalisation » des initiales pour un titre, analyse syntaxique de phrase averbale pour un élément d'énumération.

[24] Les liens hypertextes donnent un indice important de désambiguïsation lexicale (pour les entités nommées dans les articles d'encyclopédie, par exemple).

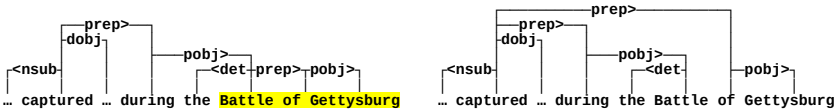

Figure 5. *Identification de l'expression «* Battle of Gettysburg *» dans l'analyse syntaxique de gauche mais pas dans celle de droite (absence de racine commune)*

On remarquera ici un cas concret d'application du mécanisme de préservation de l'ambiguïté décrit en section 2.5. Le composant qui identifie les expressions multimots contribue aussi à la désambiguïsation syntaxique ; en effet, il vote pour améliorer le score de l'analyse syntaxique où il a reconnu une expression, et l'augmentation du score est proportionnelle au nombre de mots regroupés[25].

5.4. *Analyse syntaxique profonde (transition RSyntS ⇒ RSyntP)*

Nous allons à présent illustrer les deux transitions consécutives réalisées par l'interface syntaxe-sémantique (RSyntS ⇒ RSyntP et RSyntP ⇒ RSém).

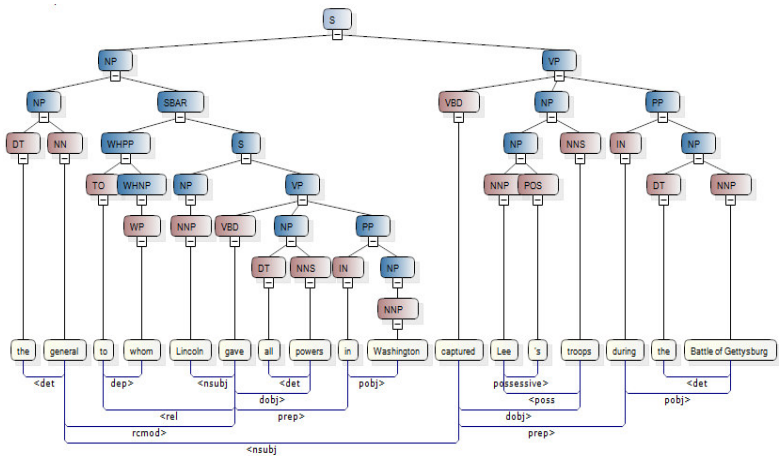

Figure 6. *Représentation syntaxique de surface de la phrase d'exemple*

La phrase que nous utiliserons, dont l'analyse syntaxique de surface est représentée figure 6, est « *the general to whom Lincoln gave all powers in Washington captured Lee's troops during the Battle of Gettysburg* » (« le général à

[25] L'hypothèse linguistique formulée ici est qu'il faut privilégier, toutes choses égales par ailleurs, les analyses syntaxiques contenant les expressions multimots les plus longues.

qui Lincoln a donné tous les pouvoirs à Washington a capturé les troupes de Lee pendant la bataille de Gettysburg »).

5.4.1. *Calcul des dépendances syntaxiques profondes*

Pour calculer les dépendances en syntaxe profonde, nous partons d'une copie de l'arbre de dépendances en syntaxe de surface. Nous cherchons d'abord à y reconnaître des sous-graphes particuliers (formes verbales passives, relatives, subordonnées…). Nous appliquons ensuite des règles de réécriture pour modifier, créer ou supprimer des dépendances. Les paires de dépendances introduisant des groupes prépositionnels (verbe vers préposition, préposition vers tête du groupe nominal) sont fusionnées pour que le verbe pointe directement vers la tête du groupe prépositionnel, en mémorisant la préposition régime. Ce module est actuellement implémenté en PROLOG ; nous pensons utiliser une boîte à outils spécialisée dans la réécriture de graphes (GrGen) pour optimiser les performances.

La figure 7 montre l'analyse de la phrase d'exemple avec les dépendances en syntaxe de surface (au-dessus des mots) et en syntaxe profonde (en-dessous des mots). On remarquera la dépendance syntaxique profonde `PropObject(to)` qui identifie « *general* » en tant que complément d'objet indirect du verbe « *give* ».

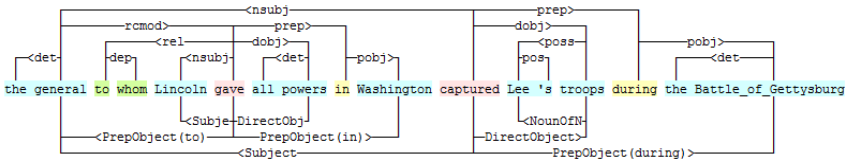

Figure 7. *Syntaxe de surface (au-dessus des mots) et syntaxe profonde (en dessous)*

Les règles de transformation RSyntS ⇒ RSyntP dépendent du jeu d'étiquettes des dépendances de surface, et sont donc fonction de l'analyseur syntaxique mis en œuvre. En revanche, le jeu d'étiquettes des dépendances en syntaxe profonde est universel. À ce stade, nous oublions les différences entre analyseurs syntaxiques, et partons toujours du même formalisme pour réaliser les traitements ultérieurs.

5.4.2. *Identification des compléments de temps et d'espace*

Antelope permet de détecter les dépendances syntaxiques profondes qui correspondent à des compléments de temps et d'espace. Un système de règles est utilisé. Par exemple, si une dépendance de type `PrepObject` utilise une préposition compatible avec un complément de temps, et que le groupe prépositionnel est une date ou un nom qui a pour hyperonyme le synset EVENT#1, alors la dépendance est affinée en `TimeComplement`, comme illustré figure 8. De plus, si dans le lexique sémantique la définition du nom contient une date (1863 pour la bataille de Gettysburg), cette information est notée dans la dépendance.

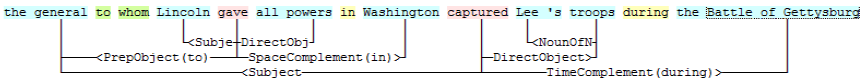

Figure 8. *Extraction des compléments de temps et de lieu*

5.4.3. *Extraction des prédicats*

À ce stade, il est aisé d'extraire des prédicats syntaxiques, en regroupant l'ensemble des dépendances en syntaxe profonde gouvernées par un même verbe ou un même nom prédicatif. La phrase d'exemple précédente donne deux prédications :

– gave(**Subject**: Lincoln, **DirectObject**: powers, **PrepObject**: *to* general, **SpaceComplement**: *in* Washington) ;

– captured(**Subject**: general, **DirectObject**: troops, **TimeComplement**: *during* Battle_of_Gettysburg).

5.5. *Analyse sémantique (transition RSyntP => RSém)*

5.5.1. *Étiquetage des rôles sémantiques*

(Chaumartin, 2006) décrit en détail l'implémentation de cette partie de l'interface syntaxe-sémantique. Les ressources WordNet, VerbNet et SUMO du lexique sémantique sont mises à contribution. VerbNet est utilisé en deux temps. Lors d'une étape préparatoire, la description de la syntaxe des classes de verbes est traduite en programmes PROLOG. Lors de l'étiquetage des rôles sémantiques d'un prédicat, ces programmes utilisent le mécanisme d'unification pour identifier des schémas, et reconnaître le cadre de sous-catégorisation du verbe fourni en entrée. Le graphe des noms de WordNet, les gloses désambiguïsées des synsets et l'ontologie SUMO servent ici à la vérification des contraintes de sélection.

Chaque prédicat calculé lors de l'analyse syntaxique profonde de l'exemple est alors étiqueté. On notera que certains rôles thématiques déclarés dans le cadre de sous-catégorisation du verbe peuvent ne pas être explicitement renseignés[26].

– E1: gave(**Agent** $_{|\text{Subject}]}$: Lincoln $_{[\text{animate}]}$, **Theme** $_{[\text{DirectObject}]}$: powers,
Recipient $_{[\text{PrepObject}]}$: *to* general $_{[\text{animate}]}$, Asset: ?, Source: ?,
Location: *in* Washington) ;

– E2: captured(**Agent** $_{[\text{Subject}]}$: general $_{[\text{animate}]}$, **Theme** $_{[\text{DirectObject}]}$: troops,
Source: ?, Beneficiary: ?, **Time**: *during* Battle_of_Gettysburg).

L'étiquetage des rôles sémantiques contribue directement à la désambiguïsation lexicale. En effet, dans le cas de l'exemple, VerbNet restreint les sens possibles du verbe ainsi que de ses actants (par application des contraintes de sélection) :

[26] *Asset* et *Source* dans le premier prédicat, *Source* et *Beneficiary* dans le second.

– seuls huit des quarante-neuf sens du verbe « *give* » sont possibles ;

– le nom « *general* » étant contraint par un trait *Animé*, ses deux sens possibles sont GENERAL#1 et #2 (le sens #3, général par opposition à particulier, est exclu) ;

– le seul sens possible du verbe « *capture* » dans le contexte est CAPTURE#5.

VerbNet décrit aussi la sémantique fine de chaque cadre de sous-catégorisation avec un « assembleur conceptuel » composé de deux cents prédicats de base. Par exemple, la traduction du premier prédicat dans ces concepts élémentaires donne :

– has_possession(start(E1), Agent=Lincoln, Theme=powers) ;

– has_possession(end(E1), Recipient=general, Theme=powers) ;

– transfer(during(E1), Theme=powers) ;

– cause(Agent=Lincoln, E1).

L'étiquetage des rôles sémantiques est une opération qui nécessite des développements complémentaires avant d'être réellement utilisable. Aujourd'hui, nous considérons qu'elle n'identifie correctement les rôles que sur un tiers des textes. Cette limitation nous semble liée à deux facteurs principaux : d'une part, la couverture perfectible de la ressource VerbNet, et d'autre part la complexité technique intrinsèque de la tâche.

5.5.2. *Désambiguïsation lexicale*

La désambiguïsation lexicale est une tâche complexe comme le rappelle (Ide et Véronis, 1998). Utiliser WordNet comme référence ne facilite pas forcément cette tâche ; disposer d'un lexique multipliant les nuances de sens a pour revers de la médaille de complexifier l'identification du « meilleur » sens[27] ; d'autre part, l'exploration d'un graphe riche et dense de grande taille doit être soumise à des conditions d'arrêt pour éviter une explosion combinatoire lors des recherches.

Notre approche actuelle de la désambiguïsation lexicale consiste à implémenter plusieurs heuristiques, pondérées par une importance relative, participant à un vote. On peut distinguer plusieurs typologies d'algorithmes. Certains sont spécialisés dans la reconnaissance d'un phénomène particulier, peu fréquent ; ils sont alors très précis, mais ne s'appliquent que dans de rares cas de figure. On peut classer dans cette catégorie l'heuristique qui reconnaît le sens de « *Paris* » dans des constructions comme « *Paris, Texas* » ou « *Paris, France* » ; simple à implémenter, et s'appliquant dès le niveau d'étiquetage morphosyntaxique, elle offre une précision de 80,0 %[28].

[27] (Véronis, 2001) souligne que la granularité de WordNet est souvent trop fine pour que même des humains s'accordent sur la bonne étiquette à donner à un mot.

[28] Ces évaluations ont été faites sur le corpus SemCor. Ce sous-ensemble du corpus Brown compte au total 676 546 mots (hors ponctuations) dont 234 135 noms, verbes, adjectifs et adverbes. Leur sens a été annoté manuellement par rapport à WordNet 1.6.

D'autres sont au contraire de portée très générale. Dans cette famille, l'algorithme consistant à prendre le premier sens d'un mot dans WordNet sert souvent de base de comparaison. Son implémentation est particulièrement simple ; il propose toujours un résultat pour un nom commun (sauf quand le nom est inconnu du lexique), ce qui lui donne un rappel voisin de 100 % ; sa précision est de 71,3 %.

Les autres heuristiques implémentées dans Antelope sont :

– un simple test de la capitalisation de l'initiale (précision = 88,5 %) ;

– l'algorithme de (Lesk, 1986) enrichi pour WordNet, décrit dans (Banerjee et Pedersen, 2003), qui consiste à compter le nombre de mots communs entre les définitions d'un mot et les définitions des mots de son contexte (ici, une fenêtre de quatre mots pleins à gauche et à droite du mot cible). Le sens retenu correspond à la définition pour laquelle on compte le plus grand nombre de mots communs avec le contexte. WordNet permet de généraliser cette approche en suivant les relations de synonymie et d'hyperonymie. La précision obtenue ici est de 45,6 % ;

– les restrictions de sens appliquées par l'interface syntaxe-sémantique lors de l'application des cadres de sous-catégorisation (précision = 42,7 %).

Ces différentes heuristiques peuvent être activées ou non, à la demande. Le résultat du vote est la combinaison linéaire des valeurs calculées par chaque heuristique, pondérées par son poids. Quand toutes les heuristiques sont combinées, la précision globale est de l'ordre de 55 % ; ce chiffre est décevant quand on le compare à l'heuristique qui consiste à simplement choisir le premier sens. Pour essayer d'améliorer ces résultats, nous avons en cours de conception un algorithme d'apprentissage des relations syntaxiques du corpus SemCor ; sur des tests préliminaires, il offre une précision de l'ordre de 60 %.

5.5.3. *Résolution d'anaphores et de coréférences*

Une anaphore est un mot ou un syntagme qui, dans un énoncé, assure une reprise sémantique d'un précédent segment appelé antécédent ; l'utilisation d'anaphores permet d'éviter une répétition lexicale. La résolution d'anaphores est un problème complexe en linguistique (pour la modélisation des phénomènes entrant en jeu) et en TAL (pour l'implémentation de ces modèles). Elle fait appel, selon le cas de figure, à des connaissances de nature lexicale, syntaxique, sémantique et pragmatique, ainsi qu'à une compréhension du contexte, pour lever les éventuelles ambiguïtés. Une vaste littérature existe autour de ce sujet[29], proposant de classifier ces phénomènes (anaphore pronominale, nominale…) ou d'en proposer des modélisations.

Antelope met en œuvre un système complet de résolution d'anaphores et d'identification de chaînes de coréférence (l'ensemble des anaphores portant sur une même entité). Nous utilisons simultanément des heuristiques classiques, pauvres en connaissances, qui s'appliquent dès le niveau d'étiquetage morphosyntaxique, et des techniques requérant une analyse syntaxique profonde et le lexique sémantique.

---

[29] Voir par exemple (Salmon-Alt, 2002) ou (Kleiber, 1994).

L'algorithme macroscopique de fonctionnement est classique :

– analyse du texte (étiquetage morphosyntaxique ou analyse syntaxique) ;

– parcours du document :

- détection des pronoms personnels et possessifs,

- détermination du caractère anaphorique du pronom, par élimination des « *it* » pléonastiques (« *It is possible that…* ») et impersonnels (« *It rains…* ») ;

– pour les pronoms anaphoriques :

- marquage des différents antécédents candidats,

- vérification des contraintes syntaxiques (c-commande),

- vérification de l'accord en genre et en nombre,

- application de différentes heuristiques qui augmentent ou diminuent le score de chaque candidat ; celui présentant au final le score le plus élevé est retenu ;

– extraction des chaînes de coréférences (calcul des composantes connexes du graphe des anaphores).

Les heuristiques sont le cœur de traitement de ce composant. Nous avons fondé notre première implémentation sur les heuristiques de (Mitkov, 1998). Disposant d'une plate-forme autorisant une analyse syntaxique, nous avons pu y ajouter les caractéristiques de l'algorithme de (Lappin et Leass, 1994). Nous avons également implémenté une heuristique de résolution des anaphores nominales, qui permet, par exemple, d'identifier correctement l'anaphore dans « *As* ***Lincoln*** *sat in the balcony, Booth crept up behind the* ***President****'s box.* » Cette dernière heuristique utilise les mesures de similarité décrites en section 4.2.

L'ensemble a été évalué dans le cadre d'articles d'encyclopédies. Les anaphores étant fortement présentes dans de tels articles, leur résolution est indispensable si on souhaite parvenir à une représentation sémantique correcte. Ces anaphores sont majoritairement pronominales, et portent souvent sur le titre (le sujet) de l'article. Sur quarante-sept anaphores relevées lors d'une annotation manuelle de onze articles, quarante-six ont été détectées correctement. L'antécédent a été correctement trouvé dans quarante-trois cas. Sur ce jeu de tests réduit, la précision est donc de 93 % et le rappel de 97 %.

6. Positionnement par rapport à d'autres plates-formes

Cette section présente brièvement des architectures et plates-formes de référence de traitement du langage, en positionnant Antelope par rapport à celles-ci. Au vu des caractéristiques de ces plates-formes, il nous semble que la principale originalité d'Antelope réside dans son modèle en niveaux de représentations clairement définis, et dans la présence d'une interface syntaxe-sémantique.

6.1. *UIMA (http://incubator.apache.org/uima - www.research.ibm.com/UIMA)*

UIMA (*Unstructured Information Management Architecture*) est une architecture logicielle[30] pour le développement et le déploiement d'outils d'analyse de l'information non structurée. Son objectif est de décrire les étapes de traitement d'un document de type texte, image ou vidéo afin d'en extraire de façon automatique des informations structurées. En revanche, UIMA ne décrit ni comment ces informations doivent être extraites, ni la façon de s'en servir. Elle a pour ambition de s'imposer en tant que norme et standard industriels.

Antelope a des objectifs plus modestes, et ne traite que l'analyse d'un texte et l'extraction des connaissances qu'il contient. Le modèle unifié d'Antelope est conçu sur mesure pour assurer cette tâche ; il ne s'agit donc pas d'un métamodèle, comme c'est le cas dans UIMA. On peut souligner comme similarités entre UIMA et Antelope que dans les deux cas, la conception est orientée composants et basée sur un modèle de programmation par interfaces, avec des structures extensibles.

6.2. *GATE (http://gate.ac.uk)*

GATE (*General Architecture for Text Engineering*) (Cunningham *et al.*, 1996) est une infrastructure permettant le développement et le déploiement de composants pour le traitement de la langue naturelle. Développée depuis 1995 à l'Université de Sheffield, elle est largement utilisée sur des tâches de fouille de textes et d'extraction d'informations. GATE propose une architecture, un *framework* en Java (incluant de nombreux modules) et un environnement de développement intégré.

En comparaison de GATE, Antelope n'offre pas d'environnement de développement intégré ; ce rôle est joué par Visual Studio, l'environnement standard pour l'architecture .NET. En ce qui concerne les composants linguistiques, le tronc commun inclut la segmentation, l'étiquetage morphosyntaxique, la détection de coréférences, l'identification d'entités nommées et l'analyse syntaxique. GATE intègre en standard plusieurs modules (annotation de documents, extraction d'informations…) qui n'existent pas, par défaut, dans Antelope.

6.3. *OpenNLP (http://opennlp.sourceforge.net - www.codeplex.com/sharpnlp)*

OpenNLP est une boîte à outil pour le TAL, codée en Java, contenant des modules de segmentation, étiquetage morphosyntaxique, *chunking*, analyse syntaxique en constituants, détection d'entités nommées, extraction des coréférences ; ces différents modules se basent sur la librairie Java d'apprentissage *OpenNLP.Maxent*, qui utilise un modèle de maximisation d'entropie.

---

[30] IBM a créé une implémentation de référence *open source* d'UIMA en Java et C++.

La conception d'ensemble d'OpenNLP et sa couverture nous paraissent proches de celles d'Antelope. Nous disposons toutefois d'une interface syntaxe-sémantique et d'analyseurs syntaxiques en dépendances absents d'OpenNLP.

6.4. *LinguaStream (http://www.linguastream.org)*

LinguaStream (Bilhaut et Widlöcher, 2006) est une plate-forme générique pour le TAL, développée en Java au GREYC depuis 2001. Son environnement de développement intégré permet de créer visuellement des chaînes de traitement linguistique complexes, en assemblant des modules de différents niveaux. Chaque maillon de la chaîne peut annoter le document. LinguaStream facilite la réalisation d'expériences sur corpus, en ne requérant que peu de compétences informatiques.

Le public visé par les deux plates-formes n'est pas exactement identique. Pour Antelope, il s'agit essentiellement de développeurs informaticiens ; la cible de LinguaStream est peut-être davantage constituée de linguistes informaticiens désireux de réaliser facilement des expérimentations sur corpus.

7. Applications de TAL utilisant Antelope

7.1. *Extraction d'informations*

Nous présentons ici des exemples d'applications utilisant Antelope en commençant par l'outil d'extraction d'informations livré à titre de démonstration avec la plate-forme. Il permet d'automatiser des requêtes sur le Web telles que « Quelle sont les sociétés que X a achetées ? ». L'utilisateur exprime d'abord ses requêtes sous forme de plusieurs paraphrases[31], qui sont envoyées à des moteurs de recherche du Web (actuellement Google, MSN ou Yahoo). Les pages Web obtenues sont dédoublonnées, puis chaque page est examinée ; le système filtre les phrases et retient celles qui contiennent les mots-clés de l'une des paraphrases de la requête, dans le même ordre ; ces phrases font ensuite l'objet d'une analyse syntaxique en dépendances. Finalement, le système vérifie (avec un mécanisme d'unification) que l'arbre syntaxique de la requête est contenu dans celui de la phrase analysée.

Antelope est utilisée ici pour la segmentation, l'analyse syntaxique, et le *pattern matching* entre les critères de requête et les phrases analysées. Sur un exemple utilisant dix-huit paraphrases de la requête « *Microsoft buys X* », nous collectons deux mille huit cents pages Web ; 10 % de ces documents sont identifiés comme contenant une réponse à la question posée. Un examen manuel de ces documents montre que le nom de la société achetée est correctement identifié dans deux cent

[31] Manuellement pour l'instant ; nous espérons pouvoir automatiser largement cette étape dans une version ultérieure, en demandant à l'utilisateur de fournir un exemple significatif, contenant deux entités nommées, permettant de trouver automatiquement des paraphrases.

dix cas ; cela correspond à une encourageante précision de 75 % ; le rappel est toutefois largement améliorable.

7.2. *Appariement entre WordNet et un sous-ensemble de la Wikipédia anglaise*

(Chaumartin, 2007) présente une méthode permettant d'établir automatiquement des liens entre WordNet et l'encyclopédie libre collaborative Wikipédia. Le lexique sémantique est enrichi de nouveaux synsets, avec une identification automatique du bon hyperonyme. La précision de l'appariement entre WordNet et un sous-ensemble de la Wikipédia anglaise (autour de quinze mille huit cents articles) est de 92 % ; en cas de création d'un nouveau synset, l'hyperonyme est correctement identifié dans 85 % des cas.

Antelope est utilisée ici, d'une part pour effectuer une analyse syntaxique de la première phrase d'un article, de façon à détecter son genre prochain, d'autre part pour calculer une distance entre définitions, de façon à proposer des appariements.

7.3. *Analyse de sentiments et d'émotions*

Antelope a été utilisée lors du *workshop* ACL SemEval-2007 (Chaumartin, 2007b) pour analyser automatiquement les sentiments et émotions (colère, dégoût, peur, joie, tristesse, surprise) suscités par des titres de journaux. Il fallait également déterminer s'il s'agissait d'une bonne ou d'une mauvaise nouvelle. Ce système a obtenu la meilleure précision (89,43 %) dans la détection des émotions.

7.4. *Détection de métaphores et de métonymies dans WordNet*

(Barque et Chaumartin, 2008) décrivent une analyse et une modélisation des relations de polysémie régulière. Cette étude exploite la hiérarchie des synsets (noms et verbes) du lexique sémantique, et la définition associée à chacun de ces synsets. Le résultat est constitué d'un ensemble de règles qui ont permis d'identifier d'une façon largement automatisée deux mille trois cent cinquante instances de relations de métaphore et de métonymie, avec une précision voisine de 91 %. La méthode utilisée permet aussi d'obtenir une désambiguïsation lexicale partielle de la définition associée aux synsets.

8. Perspectives et conclusion

Disponible sans contrainte pour la recherche et l'enseignement, Antelope a été téléchargée (sur www.proxem.com) par plus de huit cents internautes. Compatible avec les principaux systèmes d'exploitation du marché (Windows et Linux[32]), cette

[32] Antelope fonctionne en principe sous Mac OS avec MONO, mais le test reste à effectuer.

plate-forme de traitement linguistique est encore en cours de développement (version 0.8.5, en novembre 2008) mais d'ores et déjà utilisable. Nous pensons que la dimension « industrielle » de la plate-forme est justifiée parce qu'elle est :

– robuste : elle a fait ses preuves sur l'analyse syntaxique de larges corpus ;

– simple à mettre en œuvre : elle est livrée avec un programme d'installation et une documentation complète (tutorial, exemples de code, fichier d'aide). Un informaticien non linguiste peut programmer des traitements syntaxiques et sémantiques complexes au sein d'un système d'information d'entreprise ;

– extensible : un informaticien linguiste peut facilement implémenter des heuristiques spécifiques, et enrichir le lexique sémantique avec ses propres données.

Nous souhaitons continuer à faire progresser Antelope sur deux axes.

Le premier axe porte sur le multilinguisme ; nous avons récemment commencé la prise en compte du français dans Antelope, en intégrant[33] l'analyseur syntaxique TagParser et WOLF (WordNet libre du français). Nous pensons fournir, courant 2009, un niveau de traitement sémantique du français comparable à ce qui est déjà proposé aujourd'hui pour l'anglais.

Le second axe vise à améliorer les performances de l'interface syntaxe-sémantique sur ses différentes composantes :

– en exploitant les N-grams de Google[34] ;

– en introduisant de nouvelles heuristiques basées sur l'apprentissage ;

– en augmentant l'interaction et le partage de résultats entre composants de traitement, pour arriver à un système de coopération entre agents linguistiques.

Notre objectif est d'appliquer Antelope d'une façon opérationnelle à l'extraction de connaissances encyclopédiques. Nous espérons disposer, fin 2009, d'une indexation sémantique des vingt-huit mille articles de la *Simple Wikipedia* anglaise.

9. Bibliographie

Barque L., Chaumartin F., « La polysémie régulière dans WordNet », *Actes de TALN*, 2008, Avignon.

Banerjee S., Pedersen T., "Extended gloss overlaps as a measure of semantic relatedness", In *8th International Conference on Artificial Intelligence (IJCAI)*, 2003, Acapulco, Mexico.

Bilhaut F., Widlöcher A., "LinguaStream: An Integrated Environment for Computational Linguistics Experimentation", *Proc. of the 11th Conference of the European Chapter of the Association of Computational Linguistics (Companion Volume)*, 2006, Trento, Italy.

[33] À titre indicatif, l'intégration de ces deux ressources a nécessité une dizaine de jours.

[34] Nous espérons pouvoir mettre à profit cette ressource dans les tâches de désambiguïsation lexicale, désambiguïsation syntaxique et résolution d'anaphores.

Chaumartin F., « Construction automatique d'une interface syntaxe-sémantique utilisant des ressources de large couverture en langue anglaise », *Actes de RECITAL*, 2006, Leuven.

Chaumartin F., « Extraction de paraphrases désambiguïsées à partir d'un corpus d'articles encyclopédiques alignés automatiquement », *Actes de RECITAL*, 2007, Toulouse.

Chaumartin F., "A knowledge-based system for headline sentiment tagging", *Proc. of SemEval-2007 (ACL Workshop)*, 2007, Prague.

Coch J., "Interactive generation and knowledge administration in MultiMeteo", *Proc. 9th Int. Workshop on Natural Language Generation (INLG'98)*, 1998, Niagara-on-the-Lake.

Cunningham H., Wilks Y., Gaizauskas R., "GATE - a General Architecture for Text Engineering", *Proc. 16th Conference on Computational Linguistics*, 1996, Copenhagen.

Dijkstra E. W., "Solution of a problem in concurrent programming control", In *Communications of the ACM*, septembre 1965, volume 8, p. 569.

Esuli A., Sebastiani F., "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining". *Proc. of LREC 2006, fifth international conference on Language Resources and Evaluation*, 2006, pp. 417-422.

Francopoulo G., "TagParser: on the way to ISO-TC37 conformance", *Proc. of International Conference on Global Interoperability for Language Resources*, 2008, Hong Kong.

Gamma E., Helm R., Johnson R., Vlissides J., "Design patterns: Abstraction and reuse of object-oriented design", In *European Conference on Object-Oriented Programming Proceedings*, 1993, volume 707 of Lecture Notes in Computer Science, Springer-Verlag.

Gross M, "Constructing Lexicon-grammars", In *Computational Approaches to the Lexicon*, 1994, Atkins and Zampolli (eds.), Oxford Univ. Press, pp. 213-263.

Ide N., Véronis, J., "Introduction to the special issue on word sense disambiguation: the state of the art". *Computational Linguistics*, 1998, 24(1):1-40.

Kahane S., Mel'čuk I., « Synthèse de phrases à extraction en français contemporain (du réseau sémantique à l'arbre syntaxique », *TAL*, 1999, 40: 2, 25-85.

Kipper-Schuler K, "VerbNet: a broad coverage, comprehensive, verb lexicon", 2003, Thèse, University of Pennsylvania.

Kleiber G., *Anaphores et pronoms*, Duculot, 1994, Louvain-la-Neuve.

Lappin S., Leass H.J., "An algorithm for pronominal anaphora resolution", *Computational Linguistics*, 1994, p. 535-561.

Lavoie B., Kittredge R., Korelsky T., Rambow O., "A Framework for MT and Multilingual NLG Systems Based on Uniform Lexico-Structural Processing", *Proc. of the 6th Conference on Applied Natural Language Processing (ANLP)*, 2000, Seattle.

Lesk M., "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone", in *The Fifth International Conference on Systems Documentation*, 1986, ACM SIGDOC.

Lin D., "An information-theoretic definition of similarity", *Actes de 15th International Conf. on Machine Learning*, 1998, p. 296–304.

Magnini B., Cavaglià G., "Integrating Subject Field Codes into WordNet". *Actes de LREC-2000, Second International Conference on Language Resources and Evaluation*, 2000, Athènes, Grèce, pp. 1413-1418.

Manning C., Klein D., "Fast Exact Inference with a Factored Model for Natural Language Parsing", *Advances in Neural Information Processing Systems 15* (NIPS), 2002.

Mel'čuk I., *Dependency syntax: Theory and practice*. Albany, NY, 1988.

Mihalcea R., Moldovan D., "eXtended WordNet: Progress Report", In *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, Pittsburgh, PA, 2001.

Miller G., "WordNet: A lexical database", In *Communications of the ACM*, novembre 1995.

Mitkov R., "Robust pronoun resolution with limited knowledge", *COLING*, 1998, Montréal.

Niles I., Pease A., "Linking Lexicons and Ontologies: Mapping WordNet to the Suggested Upper Merged Ontology". *Proc. of International Conference on Information and Knowledge Engineering (IKE '03)*, 2003, Las Vegas, Nevada.

Sagot B., Fišer D., « Construction d'un WordNet libre du français à partir de ressources multilingues », *Actes de TALN*, 2008, Avignon.

Salmon-Alt S., « Le projet Ananas : l'annotation anaphorique pour l'analyse de corpus sémantiques », *Actes du Workshop CRAA – TALN*, 2002, Nancy.

Schwab D., « Approche hybride -lexicale et thématique- pour la modélisation, la détection et l'exploitation des fonctions lexicales en vue de l'analyse sémantique de texte », 2006, Thèse, Université Montpellier II.

Sleator D., Temperley D., "Parsing English with a Link Grammar", *Proc. of the Third International Workshop on Parsing Technologies*, 1991.

Strapparava C., Valitutti A., "WordNet-Affect: an Affective Extension of WordNet", *Proc. of LREC*, 2004, Lisbonne, pp. 1083-1086.

Tsuruoka Y., Tsujii J., "Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data", *Proc. of HLT/EMNLP*, 2005, pp. 467-474.

Véronis J., "Sense tagging: does it make sense?", In *The Corpus Linguistics Conference*, 2001, Lancaster, UK.