TALN 2010, Montréal, 19–23 juillet 2010

LEOPAR, un analyseur syntaxique pour les grammaires d'interaction

Bruno Guillaume, Guy Perrier
INRIA Nancy-Grand Est - LORIA - Nancy-Université

Résumé. Nous présentons ici l'analyseur syntaxique LEOPAR basé sur les grammaires d'interaction ainsi que d'autres outils utiles pour notre chaîne de traitement syntaxique.

Abstract. We present the parser LEOPAR which is based on the Interaction Grammars formalism. We present also other tools used in our framework for parsing.

Mots-clés : Analyse syntaxique, grammaires d'interaction, polarités.

Keywords: Parsing, Interaction Grammars, polarities.

Introduction

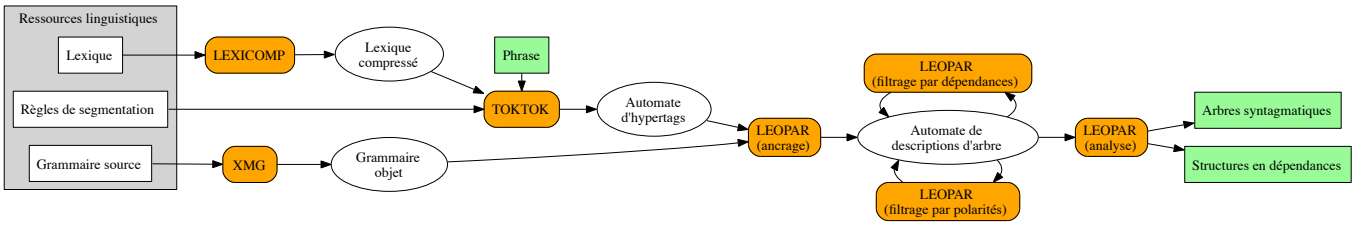
Les grammaires d'interaction (Guillaume & Perrier, 2010) sont un formalisme grammatical qui place la notion de *polarité* au cœur du mécanisme de composition syntaxique. Les objets de base d'une grammaire d'interaction sont des fragments d'arbres syntaxiques sous-spécifiés qui sont décorés par des polarités. Ces polarités expriment l'état de saturation du fragment concerné et sa capacité d'interaction avec d'autres fragments. La composition syntaxique consiste alors à superposer partiellement ces fragments d'arbres pour saturer leurs polarités et obtenir un arbre unique complètement spécifié où toutes les polarités auront été saturées.

L'opération de superposition d'arbres est plus générale que les opérations habituelles dans les grammaires d'arbres comme les TAG et elle permet donc de mieux abstraire et de factoriser un certain nombre de constructions syntaxiques. Cependant les grammaires à large couverture restent nécessairement très ambiguës. Avec les ressources actuelles pour le français, l'ambiguïté moyenne par mot est de l'ordre de 7, c'est-à-dire qu'il y a en moyenne 7 descriptions d'arbres différentes à considérer pour chaque mot d'une phrase et donc que le nombre de combinaison à considérer pour une phrase de N mots et de l'ordre de 7^N.

Les outils que nous avons développés ont pour but de valider le formalisme en permettant d'écrire des ressources à large échelle et de les tester. Ainsi l'analyseur LEOPAR dans sa version actuelle est conçu pour construire, à l'aide d'une grammaire et d'un lexique, toutes les analyses possibles d'une phrase en entrée. Compte tenu de cet objectif, nous n'avons pas développé de méthodes statistiques pour l'analyse syntaxique dans les grammaires d'interaction et nous utilisons peu d'heuristiques lors des analyses. Nous nous sommes essentiellement intéressés à la mise au point de méthodes exactes de désambiguïsation lexicale.

1 La chaîne de traitement

La chaîne de traitement de LEOPAR est décrite par la figure ci-dessous :



Le compilateur de grammaires XMG [1] est un outil de développement de grammaires à large couverture, pour lesquelles le maintien de la cohérence est une tâche particulièrement difficile. XMG est fondé sur la distinction entre *grammaire source* et *grammaire objet*. La première est écrite par un humain dans un langage de haut niveau sous forme de classes combinées par conjonction ou disjonction. Ensuite, XMG compile cette grammaire source en un grammaire objet directement utilisable par un système de TAL.

Le compilateur de lexiques LEXICOMP permet de compiler des lexiques extensionnels sous forme d'automates compacts et rapides d'accès. Chaque forme fléchie est décrite à l'aide d'hypertags qui regroupent les informations syntaxiques et morphologiques.

Le segmenteur en mots TOKTOK permet de segmenter en mots une phrase. Il utilise les informations du lexique et représente l'ambiguïté à l'aide d'automates acycliques.

L'ancrage de la grammaire dans le lexique utilise la notion d'hypertag. À chaque arbre produit par XMG est associé un hypertag qui décrit les contraintes d'ancrage qui se fait alors par unification avec les informations lexicales.

Le filtrage avec les polarités (Bonfante *et al.*, 2004) utilise la saturation des polarités comme principe contrôlant la composition syntaxique. Ce principe est utilisé pour la désambiguïsation lexicale : on peut éliminer les choix lexicaux qui ne sont pas globalement neutres.

Le filtrage avec les dépendances (Bonfante *et al.*, 2009) repose sur le fait que chaque arbre initial est une structure insaturée qui attend d'interagir avec un autre arbre pour créer une dépendance. On peut calculer statiquement sur la grammaire objet une matrice de contraintes ; celle-ci permet, pour une phrase donnée, de supprimer les choix lexicaux qui n'ont pas de possibilité de se saturer dans la phrase considérée.

L'analyse profonde recherche de façon exhaustive l'ensemble des modèles pour lesquels l'analyse de gauche à droite ne laisse pas un nombre trop grand de polarités non résolues. La recherche se fait incrémentalement par fusion successives de nœuds.

2 La mise en œuvre de LEOPAR

En plus d'une interface par ligne de commande, l'analyseur offre une interface graphique de dialogue avec l'utilisateur. Cette interface permet de piloter l'analyse mais aussi de visualiser les ressources lexicales et grammaticales, ce qui est très utile pour le débogage. Sous une forme ou sous une autre, l'interface

1. XMG (Duchier *et al.*, 2004) est librement disponible sous licence CeCILL (`http://sourcesup.cru.fr/xmg`)

utilisateur fournit différents paramètres qui permettent de personnaliser l'utilisation de l'analyseur.

L'analyse peut être effectuée de façon totalement automatique ou selon un mode manuel. Dans ce cas, c'est l'utilisateur qui effectue la sélection lexicale et qui ensuite choisit, à chaque étape de l'analyse, les nœuds à fusionner. Une fenêtre interactive permet de visualiser l'état d'analyse à chaque étape et de revenir éventuellement en arrière pour la reprendre différemment. Comme il est expliqué dans Marchand *et al.* (2009, 2010), les saturations utilisées lors de l'analyse permettent également de déduire une structure en dépendances.

Notre chaîne de traitement a été utilisée pour produire FRIGRAM, une GI du français à large couverture. La plupart des constructions grammaticales du français sont couvertes, et parmi elles, un certain nombre qui sont non triviales : coordination, extraction avec "pied piping" et barrières, négation . . . La grammaire objet, dans son état actuel, contient 2 670 arbres élémentaires sous-spécifiés non ancrés issus de 359 classes de la grammaire source.

La grammaire a été testée sur la TSNLP du français (Test Suite for Natural Language Processing). Le fait que notre grammaire soit fondée sur des connaissances linguistiques lui assure une bonne précision et limite la surgénération : 88% des 1 300 phrases grammaticales sont analysées correctement et 85% des 1 600 phrases non grammaticales sont rejetées par notre grammaire.

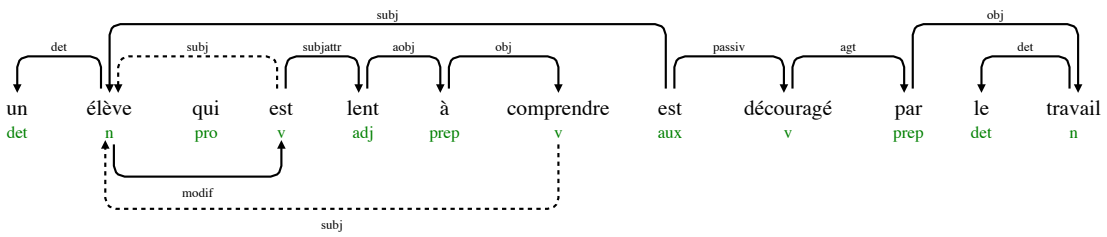
3 Exemple

Considérons la phrase : « *Un élève qui est lent à comprendre est découragé par le travail.* »

La figure ci-dessous montre les résultats obtenus par les différentes méthodes de filtrage. Avant tout désambiguïsation, chaque mot a en moyenne 7.34 structures possibles (soit $7.34^{12} = 24 \times 10^9$ sélections lexicales à considérer). Chacune des deux méthodes de filtrages permet de diminuer fortement cette ambiguïté moyenne (2.63 pour les polarités et 2.90 pour les dépendances) mais surtout, elles sont très complémentaires tant en temps de filtrage qu'en termes d'ambiguïté (au final une ambiguïté moyenne de 1.53, c'est-à-dire seulement 168 sélections lexicales à considérer).

Méthode de désambiguïsation	Nombre de sélections lexicales	Ambiguïté par mot	temps
avant désambiguïsation	24 671 606 400	7.34	
polarité	112 781	2.63	1.78s
dépendances	362 880	2.90	0.02s
polarité + dépendances	168	1.53	0.16s

Faute de place, seul le format en structure de dépendances est illustré ci-dessous pour notre exemple. Il est intéressant de noter qu'on obtient plus que des dépendances de surfaces : en traits discontinus, par exemple, on note que le nom *élève* est le sujet de l'infinitif *comprendre*.



4 Perspectives

De façon évidente, LEOPAR n'est pas actuellement opérationnel pour traiter de gros corpus comme ceux en jeu dans la dernière campagne d'évaluation PASSAGE [2]. LEOPAR n'est pas conçu pour faire de l'analyse robuste, il ne fournit donc aucune information partielle pour des phrases qui ne sont pas grammaticales au sens de notre grammaire. De plus, LEOPAR n'utilise pas de statistiques, il fait une recherche exhaustive parmi les nombreuses ambiguïtés possibles pour une phrase d'un corpus ; les méthodes de désambiguïsation développées permettent d'analyser des phrases d'environ 20 mots mais nous ne pouvons pas pour l'instant analyser les phrases plus longues.

Les deux points sur lesquels nous travaillons actuellement sont, d'une part, le développement de méthodes d'analyses syntaxiques robustes et, d'autre part, la développement d'analyses sémantiques à partir des structures produites par l'analyseur (Bonfante *et al.*, 2010).

Remerciements

Nous tenons à remercier Paul Masson qui a largement contribué au développement des outils présentés ici.

Références

BONFANTE G., GUILLAUME B. & MOREY M. (2009). Dependency constraints for lexical disambiguation. In *proceedings of IWPT 09*, Paris, France.

BONFANTE G., GUILLAUME B., MOREY M. & PERRIER G. (2010). Réécriture de graphes de dépendances pour l'interface syntaxe-sémantique. In *Actes de TALN 10*, Montréal, Canada.

BONFANTE G., GUILLAUME B. & PERRIER G. (2004). Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *CoLing'2004, 2004*, p. 303–309, Geneva, Switzerland.

DUCHIER D., LE ROUX J. & PARMENTIER Y. (2004). The metagrammar compiler : A NLP Application with a Multi-paradigm Architecture. In *Second International Mozart/Oz Conference - MOZ 2004, Charleroi, Belgium*.

GUILLAUME B. & PERRIER G. (2010). Interaction Grammars. *Research on Language and Computation (à paraître)*.

MARCHAND J., GUILLAUME B. & PERRIER G. (2009). Analyse en dépendances à l'aide des grammaires d'interaction. In *Actes de TALN 09*, Senlis, France. poster.

MARCHAND J., GUILLAUME B. & PERRIER G. (2010). Motifs de graphe pour le calcul de dépendances syntaxiques complètes. In *Actes de TALN 10*, Montréal, Canada.

2. `http://atoll.inria.fr/passage/eval2.fr.html`