Chaînes de traitement syntaxique

Pierre Boullier, Lionel Clément, Benoît Sagot, Éric Villemonte de La Clergerie
INRIA - Projet Atoll
Domaine de Voluceau, Rocquencourt, B.P. 105,78153 Le Chesnay (France)
{Benoit.Sagot,Eric.De_La_Clergerie}@inria.fr
Lionel.Clement@lefff.net

Mots-clefs : Analyse syntaxique, évaluation

Keywords: Parsing, Evaluation

Résumé Cet article expose l'ensemble des outils que nous avons mis en œuvre pour la campagne EASy d'évaluation d'analyse syntaxique. Nous commençons par un aperçu du lexique morphologique et syntaxique utilisé. Puis nous décrivons brièvement les propriétés de notre chaîne de traitement pré-syntaxique qui permet de gérer des corpus tout-venant. Nous présentons alors les deux systèmes d'analyse que nous avons utilisés, un analyseur TAG issu d'une méta-grammaire et un analyseur LFG. Nous comparons ces deux systèmes en indiquant leurs points communs, comme l'utilisation intensive du partage de calcul et des représentations compactes de l'information, mais également leurs différences, au niveau des formalismes, des grammaires et des analyseurs. Nous décrivons ensuite le processus de post-traitement, qui nous a permis d'extraire de nos analyses les informations demandées par la campagne EASy. Nous terminons par une évaluation quantitative de nos architectures.

Abstract This paper presents the set of tools we used for the EASy parsing evaluation campaign. We begin with an overview of the morphologic and syntactic lexicon we used. Then we briefly describe the properties of our pre-syntactic processing that allows us to deal with real-life corpus. Afterwards, we introduce the two parsers we used, namely a TAG parser based on a meta-grammar and an LFG parser. We compare these parsers, showing their common points, e.g., the extensive use of tabulation and compact representation techniques, but also their differences, concerning formalisms, grammars and parsers. We then describe the post-processing that allowed us to extract from our analyses the data required by the EASy campaign. We conclude with a quantitative evaluation of our architectures.

1 Introduction

L'objectif pour les participants de la campagne nationale EASy pour l'Évaluation des Analyseurs Syntaxiques était d'analyser, automatiquement et en moins d'une semaine, environ 35000 phrases. Les analyses devaient être rendues dans le format défini dans le Guide d'annotation (Gendner & Vilnat, 2004). Ce format regroupe une annotation (obligatoire) en constituants et une annotation (faculative) en dépendances syntaxiques, que l'on pouvait rendre sous une forme ambiguë ou désambiguïsée. Bien que nos analyseurs soient non-deterministes, nous avons choisi de fournir à la fois des constituants et des dépendances désambiguïsées.

Les corpus à analyser étaient des corpus réels, non retravaillés, mais segmentés en tokens et en phrases, principalement à des fins d'alignement des résultats des participants. Ils couvraient différents styles, avec environ 6 000 phrases de corpus généraux (journalistiques, législatifs), 8 000 phrases de corpus littéraires, près de 8 000 phrases de corpus de courrier électronique (avec tout le bruit que l'on peut imaginer dans un tel corpus), plus de 2 000 phrases de corpus médicaux, 7 000 phrases de corpus de transcription d'oral (avec les marques spécifiques à de tels corpus, comme les hésitations, les reprises, les répétitions, etc.), et 3 500 phrases de corpus de questions (issus de concours de questions-réponses).

Il nous a donc fallu développer un certain nombre d'outils permettant de transformer ces corpus en entrées acceptables par nos analyseurs. Par ailleurs, nous avons développé un lexique morphologique et syntaxique à large couverture, une méta-grammaire TAG et une grammaire LFG, et des mécanismes permettant de désambiguïser nos analyses et d'en extraire les constituants et dépendances définis par le guide d'annotation. Ces composants ont dû être articulés harmonieusement, construisant ainsi deux chaînes complètes d'analyse syntaxique.

2 Lexique

Le lexique que nous avons utilisé est en cours de développement au sein de l'équipe (Sagot *et al.*, 2005). Il s'agit d'un lexique morphologique et syntaxique à large couverture, dont l'architecture repose sur une structure hiérarchique avec héritage. En effet, le lexique morphologique et syntaxique est construit en deux phases à partir d'informations élémentaires factorisées. La première phase, morphologique, construit un fichier de formes fléchies associées à leur lemme et leur étiquette morphologique à partir d'un fichier de lemmes, d'un fichier décrivant les différentes flexions, et d'un fichier d'exceptions. La seconde phase, syntaxique, construit le lexique final à partir du fichier de formes fléchies, d'un fichier associant les lemmes à des patrons syntaxiques et d'un fichier décrivant ces patrons au sein d'une structure d'héritage.

Le lexique comporte aujourd'hui 404 366 formes fléchies distinctes représentant 600 909 entrées dont certaines sont factorisées. Le développement de ce lexique met en œuvre différentes technique d'acquisition, de complétion et de correction. Outre la récupération de ressources libres de droits, des techniques d'apprentissage automatique de lexiques morphologiques ont été utilisées. Elles ont donné naissance à la première version du *Lefff* (Clément *et al.*, 2004; Clément & Sagot, 2004), qui est un lexique des verbes français présents dans un gros corpus journalistique. Par ailleurs, un des points faibles des lexiques est souvent le manque de couverture pour les multi-mots (tels que *pomme de terre* ou *un peu*). Nous avons donc expérimenté des techniques d'acquisition de multi-mots (cf. (Sagot *et al.*, 2005)).

Notre lexique est encore récent et comporte un certain nombre d'erreurs et de manques. Pour le compléter et le corriger, d'autres techniques ont été employées (cf. (Sagot *et al.*, 2005)). Notre module de correction orthographique permet de détecter automatiquement les mots pour lesquels il n'existe pas de correction à faible coût. Il s'agit le plus souvent de mots manquants à rajouter manuellement. Nous avons également appliqué des méthodes de détection automatique des entrées syntaxiquement incorrectes. L'idée est qu'un mot apparaissant principalement dans des phrases non-analysables a des chances d'être syntaxiquement incomplet ou erroné dans le lexique. Enfin, certaines informations spécifiques (associations verbe-préposition, verbes supports et leurs noms prédicatifs, ...) peuvent être acquises semi-automatiquement moyennant des techniques statistiques simples sur gros corpus. D'autres méthodes sont aujourd'hui envisageables, par exemple des méthodes stochastiques sur des sorties d'analyse syntaxique de corpus avec des grammaires robustes sur-génératrices (cadres de sous-catégorisation très souples, etc.).

3 Traitements pré-syntaxiques

3.1 Description

Nous avons eu à traiter des corpus bruts et donc bruités, bien loin des phrases de linguistes ou des jeux de tests, impliquant le traitement de divers types d'entités nommées[1] (Maynard *et al.*, 2001), des adresses aux « smileys », la correction de fautes d'orthographe, la délimitation des phrases et des mots, et la gestion des particularités de certains corpus oraux ou de transcriptions de sites internet. La segmentation des corpus en phrases et tokens fournie par les organisateurs était parfois soit partielle soit incompatible avec nos outils. Cette segmentation devant être celle des résultats rendus, notre chaîne de traitement pré-syntaxique (décrite plus en détail dans (Sagot & Boullier, 2005)) a été adaptée pour garder en permanence un lien entre une unité morphosyntaxique manipulée par nos outils (unité que nous appelerons *mot*) et le ou les tokens d'entrée (issus de la segmentation fournie) qui lui correspondent. Ainsi, pendant tout le processus, les tokens d'entrée sont conservés dans des *commentaires* (entre accolades et complétés par leur position dans la chaîne d'entrée) qui sont immédiatement suivis du mot associé[2]. Par exemple[3],

contactez-moi␣au␣1␣av.␣Foch,␣75016␣Paris,␣ou␣par␣e-mail␣à␣my.name@my-email.com.

deviendra, si on laisse de côté les ambiguïtés[4]

{contactez$_{0..1}$*} contactez {-moi*$_{1..2}$*} moi {au*$_{2..3}$*} à {au*$_{2..3}$*} le {1 av. Foch, 75016 Paris*$_{3..9}$*} _ADDRESS {,*$_{9..10}$*} , {ou*$_{10..11}$*} ou {par*$_{11..12}$*} par {e-mail*$_{12..13}$*} e-mail {à*$_{13..14}$*} à {my.name@my-email.com*$_{14..15}$*} _EMAIL {.*$_{15..16}$*} . {.*$_{15..16}$*} _SENT_BOUND*.

[1] Nous utilisons ce terme dans un sens légèrement plus large, en y incluant toutes les séquences de tokens de ce type, y compris celles qui ne sont généralement pas considérées comme des entités nommées (p.ex. les nombres).

[2] Nous utilisons les conventions suivantes : un mot artificiel (par exemple un identifiant d'entité nommée) commence par un « _ » ; dans le corpus, les caractères « _ », « *{* » et « *}* » sont remplacés par les mots artificiels *_UNDERSCORE*, *_O_BRACE* et *_C_BRACE*, qui sont donc des mots du lexique. Ainsi, ces trois caractères sont disponibles comme méta-caractères.

[3] Dans cet article, le symbole « ␣ » représente de manière plus visible un espace, et donc une frontière de tokens ou de mots.

[4] On notera que le même token peut être utilisé plusieurs fois de suite, pour gérer les agglutinées (ainsi *au*$_{2..3}$). Par ailleurs, le token spécial *_SENT_BOUND* indique une frontière de phrase.

Par ailleurs, pour pouvoir prendre en compte certaines ambiguïtés, le résultat de notre chaîne de traitement pré-syntaxique, et donc l'entrée de nos analyseurs n'est pas une séquence de mots mais un treillis (DAG) de mots.

L'architecture de notre chaîne de traitement pré-syntaxique est la suivante :

Grammaires locales sur texte brut : reconnaissance d'un certain nombre d'entités nommées (et autres expressions apparentées) avant la phase de correction orthographique (adresses électroniques, URL, dates, numéros de téléphone, horaires, adresses, nombres en chiffres, smileys, mots entre guillemets, ponctuations et artefacts de transcription de l'oral),

Segmentation en phrases et identification des tokens inconnus : regroupement de deux phrases (au sens de la segmentation EASy) en une seule phrase, ou à l'inverse découpage d'une phrase en plusieurs (nous avons adapté pour cela notre segmenteur, qui étend les idées simples proposées p. ex. par (Grefenstette & Tapanainen, 1994)) ; puis identification des tokens non analysables comme mots du lexique ou combinaison de mots du lexique[5],

Grammaires locales concernant les tokens inconnus : reconnaissance d'entités nommées mettant en jeu des tokens inconnus à l'aide des résultats de la phase précédente : acronymes avec leur expansion, noms propres avec titres, séquences en langues étrangères[6],

Correction orthographique et segmentation : transformation de tout token inconnu (c.-à-d. ne faisant pas partie d'une entité nommée reconnue) en un ou plusieurs mots du lexique par correction orthographique[7], segmentation des tokens et regroupement de tokens adjacents, à l'aide du correcteur orthographique SXSPELL (Sagot & Boullier, 2005),

Grammaires locales sur mots connus : entités nommées composées de mots du lexique (nombres, y compris les ordinaux, et dates écrits en toutes lettres),

Traitement non-déterministe : cette phase, qui produit un treillis de mots du lexique, permet de reconnaître les multi-mots (comme *pomme de terre*) et les agglutinées (comme *au*) tout en préservant toutes les ambiguïtés possibles, mais aussi de représenter différentes alternatives pour gérer les erreurs d'accentuation ou de majuscule initiale[8].

À titre d'illustration, la figure 1 montre la sortie de cette chaîne pour la phrase unique *Jean␣abite␣en␣outre␣au␣1␣,␣rue␣de␣la␣Pompe*, où une espace correspond à une frontière de tokens au sens de la segmentation fournie par EASy. Les notations y sont allégées, et seuls les cas où il n'y a pas correspondance exacte entre un token et un mot sont indiqués : le ou les tokens

[5] Par *combinaison de mots du lexique* nous entendons des tokens tels que *parle-m'en* ou *anti-Bush-né*.

[6] Ces grammaires reposent sur la méthode suivante. Soit $w_1 \dots w_n$ une phrase dont les mots sont les w_i. Nous définissons une fonction d'étiquetage t qui associe (grâce à des expressions régulières) une étiquette $t_i = t(w_i)$ à chaque mot w_i, où les t_i sont pris dans un petit ensemble fini d'étiquettes possibles (respectivement 9 et 12 pour les deux grammaires locales concernées). Ainsi, une séquence d'étiquettes $t_1 \dots t_n$ est associée à $w_1 \dots w_n$. Ensuite, un (gros) ensemble de transducteurs finis transforme $t_1 \dots t_n$ en une nouvelle séquence d'étiquettes $t'_1 \dots t'_n$. Si dans cette dernière la sous-séquence $t'_i \dots t'_j$ correspond à un certain patron, la séquence de mots correspondante $w_i \dots w_j$ est considérée comme reconnue par la grammaire locale.

Soit par exemple l'énoncé *Peu␣après␣,␣le␣Center␣for␣irish␣Studies␣publiait …*, où *Center*, *irish* et *Studies* ont été identifiés comme mots inconnus. On associe à cet énoncé les étiquettes suivantes : `cnpNEEucn…`(`c` correspond à *initiale en majuscule*, `n` à *probablement français* (cas par défaut), `p` à *ponctuation*, `N` à *connu comme français*, `E` à *connu comme étranger* et `u` à *inconnu*). Ces étiquettes sont transformées en la nouvelle séquence `cnpNeeeen…`, où *e* correspond à *étranger* : *Center␣for␣irish␣Studies* est reconnu comme une séquence en langue étrangère.

[7] Si la correction orthographique est impossible ou trop coûteuse, deux mots du lexique représentant les mots inconnus sont utilisés, l'un correspondant aux mots à initiale majuscule, l'autre à ceux à initiale minuscule.

[8] Nous essayons aussi de corriger les composants de multi-mots qui n'existent pas isolément mais qui ne prennent pas part à leur multi-mot. Par exemple, *brac* n'existe que comme composant du multi-mot *bric␣à␣brac*. Ainsi, *un␣brac* n'a pas été corrigé précédemment, mais est corrigé en *un␣bras*.

sont alors entre accolades, le mot associé étant indiqué derrière. On notera que *Jean*, en tant que premier mot, peut aussi désigner une catégorie de pantalon, que la faute d'orthographe sur *abite* est corrigée, la reconnaissance de l'adresse et le traitement du multi-mot et de l'agglutinée.

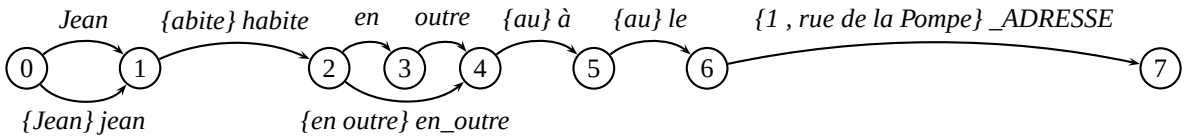

FIG. 1 – DAG associé à *Jean abite en outre au 1 , rue de la Pompe*.

Nos expériences montrent l'importance cruciale pour l'analyse syntaxique d'une telle chaîne de traitement pré-syntaxique, en particulier pour ceux des corpus d'EASy qui sont les plus éloignés du français écrit standard : les corpus de courrier électronique et de transcriptions d'oral.

3.2 Évaluation

L'évaluation d'une telle chaîne est difficile car nous ne disposons pas d'un corpus de référence approprié. Cependant, on peut en avoir un aperçu grâce à des tests préalablement menés sur un corpus journalistique de 1,1 million de mots. Tout le processus prend 13 minutes 01 seconde, soit environ 1400 tokens/sec[9]. Le tableau 1 indique les taux de détection de quelques catégories d'entités nommées manuellement validées.

Classe d'entités nommées	Occurrences	Précision	Rappel
URL	174	100%	100%
adresses (physiques)	35	100%	100%
Expressions en langue étrangère[10]	42	83%	88%

TAB. 1 – Évaluation partielle de la reconnaissance d'entités nommées.

L'évaluation de la segmentation en phrases nécessite une annotation manuelle. Nous l'avons effectuée sur les 400 premières phrases du corpus, ce qui donne un taux de précision de 100% et un taux de rappel de 100%. C'est très satisfaisant, compte tenu du fait que ce corpus journalistique est rempli de citations, de notes de bas de page, de références bibliographiques et de méta-informations qui rendent la détection des frontières de phrases assez difficile.

L'évaluation du correcteur orthographique est délicate. La phase de correction orthographique et de segmentation en mots étant réalisée par un composant qui fait appel au correcteur SXSPELL tout en gérant les phénomènes de segmentation et de majuscules, il y a deux sous-composants à évaluer : le correcteur SXSPELL et le segmenteur-correcteur qui l'utilise. De plus, il faut isoler leurs performances des qualités du lexique et du corpus considérés. Pour ce faire, nous avons identifié automatiquement parmi les 1,1 million de tokens tous ceux qui ne sont pas reconnus par le correcteur-segmenteur comme mots connus ou combinaisons valides de mots connus. Nous avons alors identifié parmi ces tokens inconnus ceux qui devraient être corrigés en des

[9]Le test a été réalisé sur une architecture AMD Athlon ? XP 2100+ (1.7 GHz) et les résultats peuvent paraître lents, comparé, par exemple, aux quelques milliers de mots par seconde que l'on peut obtenir en faisant de l'analyse syntaxique de surface. Mais la phase de correction orthographique est algorithmiquement très coûteuse (impliquant, pour chaque mot, des intersections dynamiques d'automates à plusieurs millions d'états). Les performances que nous obtenons sont donc excellentes.

[10]Test réalisé seulement sur 2000 phrases, car une annotation manuelle est nécessaire.

mots ou combinaisons de mots présents dans le lexique, et nous les avons corrigés manuellement (en tenant compte de leur contexte). Puis nous avons comparé cette correction manuelle à celle fournie par notre système. 91% des 150 tokens concernés sont corrigés (et éventuellement segmentés) correctement. Quelques exemples sont indiqués dans le tableau 2.

Token d'entrée	*arisienne*	*barriére*	*l'intervent␣ionnisme*	*n'aspire-til*	*plrrase*
Correction	*parisienne*	*barrière*	*l'␣interventionnisme*	*n'␣aspire␣-t-il*	*phrase*

TAB. 2 – Exemples de corrections réussies effectuées par le correcteur-segmenteur.

Par ailleurs, 1846 tokens sont analysés comme combinaison de mots du lexique avec (au moins) un préfixe (1712 cas) ou un suffixe (54 cas, seuls *-né*, *-clef* et leurs variantes étant concernés) connu. Ainsi, *quasi-parti␣unique␣chrétien-libéral-conservateur* est transformée en *quasi-_␣parti␣unique␣chrétien-_␣libéral-_␣conservateur*, où « -_ » est, par convention, la marque des préfixes. Il nous faut préciser à ce stade deux faits. Tout d'abord, le corpus considéré est de très bonne qualité (150 mots du français standard mal orthographiés parmi 1,1 million de mots). D'autre part, cette évaluation du correcteur-segmenteur nous a permis de réaliser l'incomplétude du lexique, en particulier en ce qui concerne les mots d'emprunt à des langues étrangères.

4 Analyseurs syntaxiques

Nous avons développé deux analyseurs utilisant des formalismes, des architectures et des grammaires différents. Le premier, SXLFG, est un analyseur LFG à deux passes. Le second, FRMG, est un analyseur TAG à une passe utilisant une grammaire qui est la représentation compacte d'une TAG avec structures de traits et qui est obtenue par compilation d'une méta-grammaire.

4.1 Analyseur SXLFG

Le système SXLFG (Boullier *et al.*, 2005) permet de construire des analyseurs à partir de grammaires écrites dans une variante du formalisme LFG (Lexical-Functional Grammars). Les grammaires sont donc des grammaires non-contextuelles (CFG) dites *grammaires support* dont les règles dont décorées par des *équations fonctionnelles* dont la résolution repose sur l'unification. Lors d'une analyse, les équations fonctionnelles sont calculées sur une représentation compacte des arbres d'analyse provenant de la grammaire support appelée *forêt partagée*.En cas d'ambiguïté, elle partage les sous-structures communes entre plusieurs analyses.

Pour obtenir un analyseur efficace, nous effectuons les calculs d'équations fonctionnelles directement sur la forêt partagée, et non sur chaque arbre d'analyse CFG. Ceci induit la spécificité de notre variante de LFG : toute information calculée dans les structures fonctionnelles ne peut l'être que de manière *bottom-up*. En effet, puisque l'on effectue ces calculs sur la forêt d'analyse sans la modifier, la structure fonctionnelle associée à la racine d'un sous-arbre ne peut dépendre que des structures associées à ses fils. Dans le cas général, le résultat de ces calculs est un ensemble de structures fonctionnelles associées à la racine de la forêt. Si cet ensemble contient plus d'un élément, on peut par la suite appliquer des heuristiques de désambiguïsation.

Notre analyseur est un analyseur robuste, et ce à plusieurs titres. Tout d'abord, l'analyseur CFG dispose de mécanismes de rattrapage d'erreurs, permettant de traiter les cas où la phrase d'entrée est agrammaticale pour la grammaire support (on parle de phrases *non-valides pour la CFG*

support). Ensuite, en cas d'échec du calcul des équations fonctionnelles, ces équations peuvent être assouplies et donner lieu à des résultats ayant divers degrés d'imperfection. Par exemple, on peut obtenir une structure pour toute la phrase d'entrée mais qui ne respecte pas nécessairement certaines contraintes comme les cadres de sous-catégorisation (on parle d'analyse *sans vérification de cohérence*, par opposition à une analyse qui se déroule correctement jusqu'au bout, dite *avec vérification de cohérence*). En cas d'échec de cet essai, des structures fonctionnelles couvrant des portions disjointes de la phrase sont produites, qui sont appelées *structures partielles*. Au pire, la phrase d'entrée peut être *sur-segmentée*, c'est-à-dire découpée en sous-phrases (avec 5 niveaux de découpage possibles) pour essayer d'en analyser des portions correctes.

Pour la campagne d'évaluation EASy, nous sommes partis d'une grammaire LFG du français développée pour le système XLFG (Clément & Kinyon, 2001), que nous avons modifiée et complétée. Sa couverture et le degré d'ambiguïté de sa grammaire support sont encore améliorables, mais elle traite correctement un nombre respectable de phénomènes syntaxiques complexes.

4.2 Analyseur FRMG

L'analyseur FRMG s'appuie sur une grammaire d'arbres adjoints (TAG) avec décorations engendrée à partir d'un niveau plus abstrait de description, une *méta-grammaire* (MG) (Candito, 1999; Thomasset & de la Clergerie, 2005). La grammaire obtenue est très compacte avec seulement 133 arbres, car elle s'appuie sur des *arbres factorisés* utilisant des disjonctions entre nœuds, des répétitions de nœuds et, surtout, des nœuds optionnels contrôlés par des gardes. L'ancrage des arbres par les entrées lexicales se fait par unification de structures de traits appelées *hypertags*.

Un analyseur syntaxique hybride TAG/TIG[11] a été compilé à partir de la grammaire. Il peut prendre en entrée les treillis produits par la chaîne d'entrée (section 3) modulo quelques conversions pour construire les hypertags. Au démarrage de l'analyse, les arbres sont filtrés par rapport aux mots du treillis d'entrée, pour ne garder que ceux dont les nœuds d'ancrages et les nœuds lexicaux sont compatibles avec ces mots. L'analyseur utilise une stratégie d'analyse tabulaire descendante gauche-droite en une seule passe : le traitement des décorations des nœuds n'est pas repoussé dans une seconde passe, contrairement à la stratégie SXLFG. Néanmoins, les décorations ne sont pas prises en compte pour les prédictions descendantes mais seulement dans les propagations de réponses. Le parcours des arbres factorisés se fait sans expansion de ceux-ci assurant une bonne efficacité. L'analyseur retourne soit une analyse complète du treillis d'entrée, soit, en mode robuste, un ensemble d'analyses partielles couvrant au mieux ce treillis. Les analyses sont émises sous formes de forêts partagées de dérivations TAG indiquant les diverses opérations effectuées (substitution, adjonction, ancrage,…) et ensuite converties en forêts partagées de dépendances (figure 2) servant de base pour les traitements post-syntaxiques.

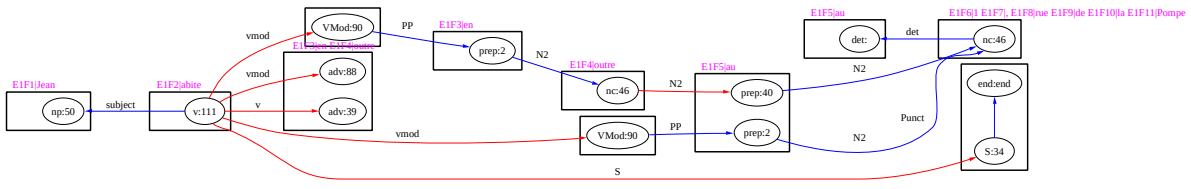

FIG. 2 – Forêt de dépendances (FRMG)

[11] Les TIG (*Tree Insertion Grammars*) sont une variantes des TAG faiblement équivalentes aux CFG.

5 Traitement post-syntaxique

Le format et la nature des informations attendus par les organisateurs de la campagne EASy (Gendner & Vilnat, 2004) ne correspondent pas nécessairement à nos propres formats et choix linguistiques (cf. figure 3). D'autre part, les techniques tabulaires de partage de calculs mises en œuvre dans nos analyseurs sont en partie motivées par le souci d'obtenir l'ensemble des analyses pour une phrase, alors que la piste d'évaluation de base pour EASy concerne des analyses syntaxiques non ambiguës. Il a donc été nécessaire de mettre en place des algorithmes de désambiguïsation et de conversion travaillant sur les structures partagées produites par nos analyseurs. Ces travaux ont été l'occasion d'explorer ce type d'algorithmes avec des approches assez différentes dans les cas de SXLFG et de FRMG. Nous avons également dû explorer diverses règles heuristiques de désambiguïsation et comprendre comment les exprimer.

GN 1	NV 2	GR 3		GP 4						
Jean	abite	en	outre	au	1	,	rue	de	la	Pompe
F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11

sujet	**verbe**
GN1	NV2

complément	**verbe**
GP4	NV2

modifieur	**verbe**
GR3	NV2

FIG. 3 – Sortie EASy fournie par SXLFG et FRMG pour la même phrase que précédemment

Dans le cas de FRMG, la désambiguïsation et la conversion s'appuient sur les forêts partagées de dépendances (section 4.2). Les arcs de dépendance se prêtent bien à l'expression d'heuristiques de désambiguïsation : chaque arc se voit attribuer un poids donné par la somme des poids élémentaires associés aux contraintes satisfaites par l'arc, avec, par exemple, un poids élevé pour une dépendance entre un verbe et un argument et moindre entre un verbe et un modifieur. Au niveau global, l'algorithme retient un ensemble d'arcs maximisant la somme de leurs poids et tels que tout nœud soit accessible par un et un seul chemin. Néanmoins, pour des raisons d'efficacité, l'algorithme a été (tardivement) complété par une notion de coût *régional* associé à un sous-ensemble d'arcs atteignables à partir d'un nœud. Une sélection bornée des meilleurs coûts régionaux est effectuée pour progressivement calculer un coût global qui n'est plus nécessairement optimal. Quoique bien plus efficace, l'algorithme reste encore trop lent dans certains cas. Une analyse plus poussée du problème (en partie aidée par l'approche suivie pour SXLFG) suggère que trop d'informations sont perdues lors de la conversion des dérivations en dépendances[12]. En particulier, le format actuel n'indique pas si deux dépendances issues d'un même mot appartiennent ou non à une même analyse, ce qui nécessite l'ajout de règles coûteuses favorisant les bonnes configurations. Nous prévoyons donc de faire évoluer notre notion de forêt partagée de dépendances. Malgré ces problèmes, nous avons pu constater l'adéquation des arcs de dépendance pour exprimer des règles de désambiguïsation ou de conversion.

Dans le système SXLFG, la phase de désambiguïsation se fait par l'application successive d'un certain nombre de règles sur les structures fonctionnelles associées à la racine de la forêt d'analyse produite par la grammaire support. Chaque règle met en œuvre un critère pour éliminer les structures fonctionnelles non optimales au sens de ce critère. La dernière règle choisit au hasard une analyse parmi celles qui restent. La forêt d'analyse est alors élaguée pour n'y laisser que l'arbre[13] support correspondant à la structure fonctionnelle choisie. L'extraction des constituants

[12] Ceci est dû au fait que nos forêts de dépendances ont initialement été conçues pour une visualisation simplifiée d'un ensemble important d'analyses.

[13] En toute rigueur, plusieurs arbres peuvent subsister s'ils correspondent à une structure fonctionnelle identique.

et des dépendances demandés par EASy se fait alors en parcourant la structure fonctionnelle et son arbre associé, à la recherche de motifs correspondant aux spécifications de la campagne. Cette phase est facilitée par le fait que l'analyse unique issue de la phase de désambiguïsation a été préalablement extraite, à l'inverse de ce qui se passe dans le système FRMG.

6 Mise en œuvre et résultats expérimentaux

Le volume de données à analyser pour EASy, le nombre d'essais que nous voulions effectuer et la complexité de la tâche étaient suffisamment conséquents pour que nous décidions de ventiler les analyses sur plusieurs machines, formant ainsi un cluster pour chaque système.

Les tableaux 3 à 5 présentent divers résultats concernant EASy mais aussi les corpus EUROTRA et TSNLP. Les nombres de phrases diffèrent selon le système, en raison d'heuristiques différentes de segmentation en phrases. Par ailleurs, le *taux d'ambiguïté moyen par mot* n'est disponible que pour FRMG, car dans SXLFG les heuristiques de désambiguïsation sont incorporées dans l'analyseur. Ce taux est défini comme le nombre moyen d'arcs de dépendance atteignant un mot moins un[14].

Corpus	#phrases	% couv.	temps d'analyse				amb.
			moy.	méd.	$\geq$ 1s	$\geq$ 10s	
EUROTRA	334	95.80%	1.81s	1.27s	61.68%	1.55%	0.7
TSNLP	1661	93.38%	0.72s	0.56s	22.03%	0.00%	0.4
EASy	34438	42.45%	5.55s	1.61s	64.41%	9.32%	0.6

TAB. 3 – Résultats pour FRMG, avec un *timeout* de 100 secondes[15]

Corpus	#phrases	couverture (sans vérif. de coh.[16])	couverture (avec vérif. de coh.)	temps d'analyse			
				moy.	méd.	$\geq$ 0.1s	$\geq$ 1s
EUROTRA	334	94.61%	84.43%	0.33s	0.02s	22.2%	6.0%
TSNLP	1661	98.50%	79.12%	0.03s	0.00s	2.8%	0.6%
EASy	40859	66.62%	41.95%	n.d.[17]			

TAB. 4 – Résultats pour SXLFG, avec un *timeout* de 15 secondes[15].

7 Conclusion

La campagne d'évaluation EASy nous a permis de mettre en évidence la différence considérable qu'il y a entre le développement d'un analyseur syntaxique et le développement d'une chaîne complète d'analyse syntaxique. En effet, outre l'importance de la qualité de la grammaire et de l'analyseur, cette campagne a montré le rôle non moins déterminant de la couverture et de la richesse du lexique, de la qualité de la chaîne de traitement, de la précision des méthodes d'exploitation des sorties des analyseurs, ainsi que la très forte interaction entre les différents composants, et en particulier entre le lexique et la grammaire.

[14] Pour une phrase non-ambiguë, chaque mot (sauf la « tête » de la phrase) est atteint par un seul arc, d'où un taux d'ambiguïté nul. Le nombre maximal d'analyses pour un taux α et une phrase de longueur n est en $O((1+\alpha)^n)$.

[16] On notera qu'un *timeout* plus élevé aurait augmenté les taux de couverture mais également les temps d'analyse.

[17] Nous n'avons pas conservé les informations permettant de donner les temps sur le corpus EASy. Toutefois, (Boullier *et al.*, 2005) donne les temps d'analyse pour les 87.51% de phrases reconnues par la CFG support.

		Corpus complet	Phrases valides pour la CFG support	
		Analyse CFG	Analyse CFG	Analyse complète
	#phrases	40859	35756	
	n_{moy} - n_{max}	20.95 - 541	19.06 - 173	
	UW_{moy} - UW_{max}	0.79 - 97	0.75 - 65	
Nombre d'analyses	med - max	$32\,028$ - 3.10^{73}	$29\,582$ - 5.10^{52}	1 - 1
	$\geq 10^{12}$	8.86%	7.84%	0%

TAB. 5 – Données sur les corpus[18] et nombres d'analyses pour SXLFG, avant application de l'heuristique de sur-segmentation.

Cette forte complémentarité entre les différentes phases des chaînes d'analyse syntaxique a inévitablement élargi le champ de la campagne EASy. Ce n'est pas seulement l'analyse syntaxique elle-même qui a été évaluée lors de cette campagne, mais la capacité à mettre en place des chaînes d'analyse syntaxique complètes[19]. Nous comptons exploiter le fait que nous avons déployé deux chaînes de traitement que tout sépare sauf le lexique et la chaîne pré-syntaxique. Ceci nous permettra d'effectuer des comparaisons et d'améliorer ainsi grammaires et analyseurs (en étudiant les différences entre nos résultats), mais aussi le lexique et la chaîne de traitement pré-syntaxique (en étudiant les erreurs communes).

Références

BOULLIER P., SAGOT B. & CLÉMENT L. (2005). Un analyseur LFG efficace : SXLFG. In *Actes de TALN'05*, Dourdan, France.

CANDITO M.-H. (1999). *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées*. PhD thesis, Université Paris 7.

CLÉMENT L. & KINYON A. (2001). XLFG-an LFG parsing scheme for French. In *Proc. of LFG'01*.

CLÉMENT L. & SAGOT B. (2004). Site internet du Lefff (Lexique des Formes Fléchies du Français). `www.lefff.net`.

CLÉMENT L., SAGOT B. & LANG B. (2004). Morphology Based Automatic Acquisition of Large-coverage Lexica. In *Proceedings of LREC'04*, p. 1841–1844.

GENDNER V. & VILNAT A. (2004). Les annotations syntaxiques de référence PEAS. En ligne sur `www.limsi.fr/Recherche/CORVAL/easy/PEAS_reference_annotations_v1.6.html`.

GREFENSTETTE G. & TAPANAINEN P. (1994). What is a word, what is a sentence ? Problems of tokenization. In *Proceedings of the 3rd CCLTR*, Budapest, Hungary.

MAYNARD D., TABLAN V., URSU C., CUNNINGHAM H. & WILKS Y. (2001). Named entity recognition from diverse text types. In *Proceedings of RANLP 2001*, Tzigov Chark, Bulgaria.

SAGOT B. & BOULLIER P. (2005). From raw corpus to word lattices : robust pre-parsing processing. In *Actes de L&TC 2005*, Poznań, Pologne.

SAGOT B., CLÉMENT L., ÉRIC VILLEMONTE DE LA CLERGERIE & BOULLIER P. (2005). Vers un méta-lexique pour le français : architecture, acquisition, utilisation. In *Journée ATALA sur l'interface lexique-grammaire*. `http ://www.atala.org/article.php3?id_article=240`.

THOMASSET F. & DE LA CLERGERIE E. V. (2005). Comment obtenir plus des méta-grammaires. In *Actes de TALN'05*, Dourdan, France.

[18] Pour les données sur les corpus, n désigne un nombre de mots, et UW un nombre de mots inconnus.

[19] En outre, l'harmonisation des résultats des différents participants passe par une segmentation commune en phrases et en mots, différente de celle produite et utilisée par nos outils, qui a dû être conservée en permanence.