Un outil pour la manipulation de ressources arborées

Yannick Parmentier[1]
(1) LIFO, Université d'Orléans, 45067 Orléans, France
yannick.parmentier@univ-orleans.fr, https://gitlab.com/parmenti/pytreeview

RÉSUMÉ

Dans cet article, nous présentons brièvement pytreeview, un outil pour la manipulation de ressources arborées (corpus annotés, grammaires électroniques). Initialement conçu pour assiter les utilisateurs linguistes dans leur tâche de développement de grammaires arborescentes, pytreeview a été étendu pour permettre de manipuler des ressources arborées variées (grammaires mais aussi corpus aux formats FTB, PTB, CoNLL, Tiger), afin d'en extraire des informations utiles (par exemple la distribution des cadres de sous-catégorisation). pytreeview est actuellement utilisé dans le cadre d'un projet visant l'extraction semi-automatique de grammaires abstraites (méta-grammaires) à partir de corpus arborés.

ABSTRACT

A tool for handling tree-based linguistic resources

In this paper, we briefly introduce pytreeview, a tool for handling tree-based linguistic resources (treebanks, electronic grammars). Initially designed for supporting linguist users in their grammar development task, pytreeview has been extended to handle various tree-based resources (grammars but also corpuses in the FTB, PTB, CoNLL, Tiger formats), in order to extract useful pieces of information from these (e.g. distributions of sub-category frames). pytreeview is currently used in a project aiming at semi-automatic abstract grammar (metagrammar) extraction from treebanks.

MOTS-CLÉS : visualisateur, grammaire d'arbres, corpus arboré.

KEYWORDS: viewer, tree grammar, treebank.

1 Introduction

De nombreuses applications de traitement automatique des langues reposent (directement ou indirectement) sur l'utilisation de ressources linguistiques sous forme arborescente. La plus courante de ces applications est probablement l'analyse syntaxique (c'est-à-dire le calcul automatique des relations entre les mots d'un énoncé, pour en construire une représentation syntaxique). Dans le cas de l'analyse syntaxique symbolique, on peut citer le projet XTAG (2001) qui a permi la création d'une grammaire noyau décrivant la syntaxe de l'anglais, comptant plus de 1000 schèmes d'arbres (c'est-à-dire, d'arbres avant lexicalisation), et utilisable en analyse. Dans le cas de l'analyse statistique, les approches courantes utilisent des ressources arborés (*treebanks*) pour la phase d'apprentissage (voir par exemple (Candito *et al.*, 2010)). Dans ce contexte, il est important de disposer d'outils permettant (au moins) l'exploration de ces ressources et leur vérification par un Humain. Les différents projets autour des grammaires et corpus arborés ont souvent été accompagnés du développement d'outils d'exploration spécifiques (voir § 3). Nous présentons ici un outil *libre*, *général* et *léger* pour la visualisation et la conversion de ressources arborescentes (voir § 2).

2 Description

Partant du constat que de nombreux formats (et outils associés) existent pour la représentation de ressources arborées (par exemple, des formats de type XML pour le French TreeBank ou le corpus Tiger, des formats textuels pour le Penn TreeBank ou le corpus CoNLL), `pytreeview` a été créé pour offrir un accès *unifié* à ces ressources, dans un but de visualisation et d'extraction d'information. `pytreeview` est donc au départ une application permettant d'*explorer* une ressource arborée (tous les formats listés ci-dessus sont supportés), et d'en afficher les arbres (plusieurs arbres peuvent être affichés simultanément, et une fonction de comparaison automatique permet de colorer les arcs et nœuds divergents entre paires d'arbres). `pytreeview` offre trois modes d'utilisation : via une interface graphique (GUI, pour une utilisation interactive, voir Figure 1 en Annexe), via une interface en ligne de commande (CLI, pour appel depuis un script par exemple), et via une interface web (pour visualiser la ressource dans un navigateur, dans le cadre d'une démo en ligne par exemple, voir Figure 2 en Annexe). En mode interactif, `pytreeview` permet également de rechercher dans la ressource arborée, l'ensemble des structures satisfaisant certains critères. Ces critères sont de 2 types : contraintes de traits et contraintes de structures (dominance ou précédence entre nœuds). Le langage utilisé pour écrire les requêtes de recherche est un sous-ensemble du langage utilisé par l'outil TigerSearch (König & Lezius, 2000, §2.2 et 2.3). En plus de pouvoir visualiser des arbres à l'écran avec zoom, `pytreeview` permet leur export sous 3 formes : image au format `png`, texte aux formats `json` (format léger permettant une interopérabilité entre ressources) et `tikz` (format permettant de documenter une ressource au moyen de l'outil d'édition LaTeX). `pytreeview` permet également d'extraire un ensemble d'information d'un corpus arborés (par exemple, le jeu d'étiquettes utilisé, la distribution des étiquettes et les différents cadres de sous-catégorisation).

`pytreeview` est développé au moyen du langage `python` (3685 lignes de codes réparties en 14 fichiers) et distribué librement sous licence libre GPLv3. La conception de `pytreeview` a été guidée par des objectifs de *légèreté* (installation et utilisation simples) et d'*extensibilité* (architecture modulaire permettant d'ajouter de nouveaux formats d'entrée / sortie, et de nouveaux traitements des arbres). `pytreeview` requiert trois librairies `python` pour la manipulation des formats d'export (`svg2tikz`, `svgwrite` et `cairosvg`), et huit autres librairies optionnelles pour l'interface graphique (par ex. `pyforms` ou `wand`), toutes sont disponibles librement et pré-compilées pour les environnements linux/windows/MacOS.

`pytreeview` est en cours de développement, et est utilisé dans le cadre d 'un projet d'extraction de (méta) grammaire. Une attention particulière a été donnée à la rapidité de traitement (utilisation de techniques de programmation parallèle pour l'export web par exemple), cependant la recherche de motif peut être longue (dizaines de sec. sur un corpus de 100000 arbres) et nécessite des optimisations.

3 Travaux liés

Le développement d'outils pour la manipulation de ressources arborées est un domaine toujours actif, comme en témoignent par exemple Gerdes (2013) pour l'annotation en dépendances et Wang *et al.* (2015) pour l'exploration de la grammaire XTAG. Les outils les plus proches de notre travail correspondent à *TigerSearch* (Brants *et al.*, 2002) et *TrEd* (Pajas & Štěpánek, 2009). Ces outils rassemblent une grande communauté d'utilisateurs et permettent non seulement la visualisation mais aussi l'édition d'arbres. La différence majeure avec notre approche réside dans le fait que ces outils ont leur propre format de représentation des données et reposent sur une architecture complexe.

Références

BRANTS S., DIPPER S., HANSEN S., LEZIUS W. & SMITH G. (2002). Tiger treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories, September 20-21 (TLT02)*, Sozopol, Bulgaria.

CANDITO M., CRABBÉ B. & DENIS P. (2010). Statistical french dependency parsing : Treebank conversion and first results. In N. C. C. CHAIR), K. CHOUKRI, B. MAEGAARD, J. MARIANI, J. ODIJK, S. PIPERIDIS, M. ROSNER & D. TAPIAS, Eds., *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, p. 1840–1847, Valletta, Malta : European Language Resources Association (ELRA).

GERDES K. (2013). Collaborative dependency annotation. In *Proceedings of the Second International Conference on Dependency Linguistics, DepLing 2013, August 27-30, 2013, Prague, Czech Republic*, p. 88–97.

KÖNIG E. & LEZIUS W. (2000). A description language for syntactically annotated corpora. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, p. 1056–1060, Stroudsburg, PA, USA : Association for Computational Linguistics.

PAJAS P. & ŠTĚPÁNEK J. (2009). System for querying syntactically annotated corpora. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, p. 33–36, Suntec, Singapore : Association for Computational Linguistics.

WANG Z., ZHANG H. & SARKAR A. (2015). A python-based interface for wide coverage lexicalized tree-adjoining grammars. *The Prague Bulletin of Mathematical Linguistics*, **103**(1), 139–159.

XTAG (2001). *A Lexicalized Tree Adjoining Grammar for English*. Rapport interne IRCS-01-03, IRCS, University of Pennsylvania.

Annexes

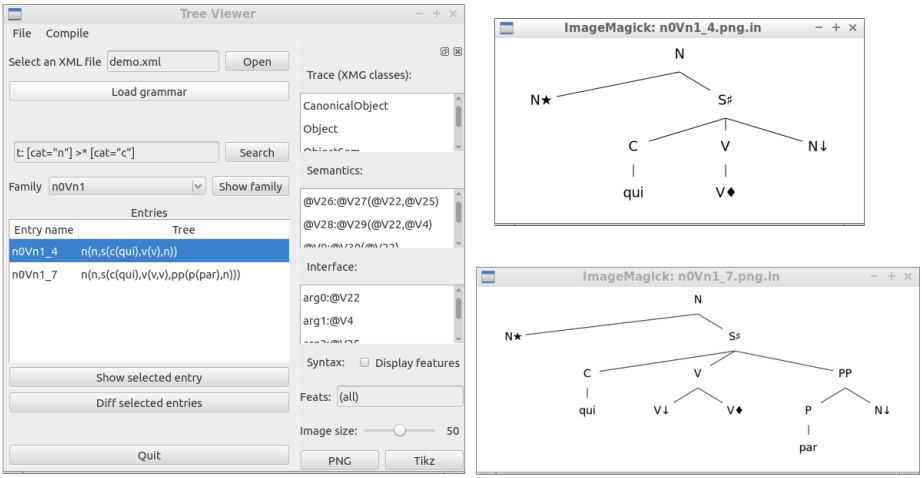

FIGURE 1 – Interface de `pytreeview` en mode *bureau*

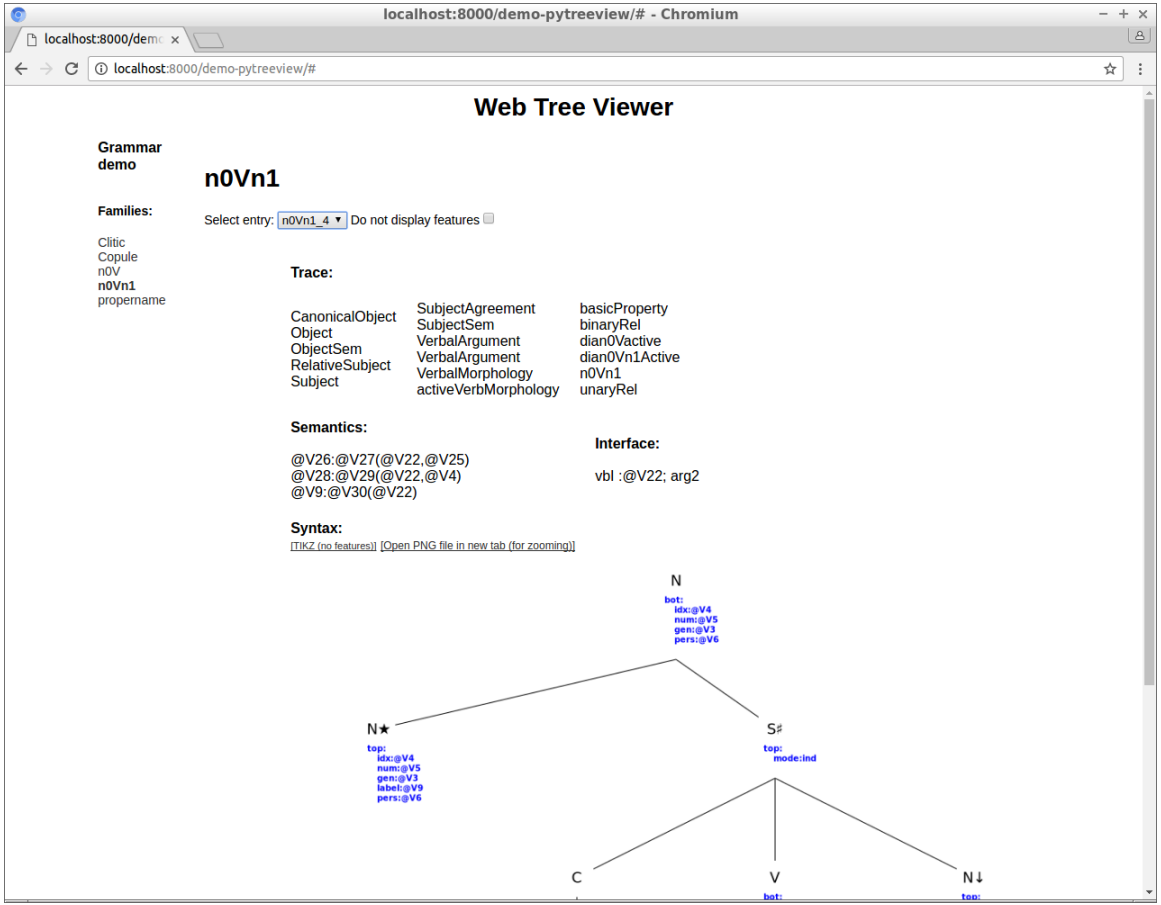

FIGURE 2 – Interface de `pytreeview` en mode *web*