Alpage: Transition-based Semantic Graph Parsing with Syntactic Features

Corentin Ribeyre[⋆ ◦] **Eric Villemonte de la Clergerie**[⋆] **Djamé Seddah**[⋆ ⋄]
[⋆]Alpage, INRIA
[◦]Univ Paris Diderot, Sorbonne Paris Cité
[⋄] Université Paris Sorbonne
firstname.lastname@inria.fr

Abstract

This paper describes the systems deployed by the ALPAGE team to participate to the *SemEval-2014 Task on Broad-Coverage Semantic Dependency Parsing*. We developed two transition-based dependency parsers with extended sets of actions to handle non-planar acyclic graphs. For the open track, we worked over two orthogonal axes – lexical and syntactic – in order to provide our models with lexical and syntactic features such as word clusters, lemmas and tree fragments of different types.

1 Introduction

In recent years, we have seen the emergence of semantic parsing, relying on various techniques ranging from graph grammars (Chiang et al., 2013) to transitions-based dependency parsers (Sagae and Tsujii, 2008). Assuming that obtaining predicate argument structures is a necessary goal to move from syntax to accurate surface semantics, the question of the representation of such structures arises. Regardless of the annotation scheme that should be used, one of the main issues of semantic representation is the construction of graph structures, that are inherently harder to generate than the classical tree structures.

In that aspect, the shared task's proposal (Oepen et al., 2014), to evaluate different syntactic-semantic schemes (Ivanova et al., 2012; Hajic et al., 2006; Miyao and Tsujii, 2004) could not arrive at a more timely moment when state-of-the-art surface syntactic parsers regularly reach, or cross, a 90% labeled dependency recovery plateau for a wide range of languages (Nivre et al., 2007a; Seddah et al., 2013).

The two systems we present both extend transition-based parsers in order to be able to generate acyclic dependency graphs. The first one follows the standard greedy search mechanism of (Nivre et al., 2007b), while the second one follows a slightly more global search strategy (Huang and Sagae, 2010; Goldberg et al., 2013) by relying on dynamic programming techniques. In addition to building graphs directly, the main originality of our work lies in the use of different kinds of syntactic features, showing that using syntax for pure deep semantic parsing improves global performance by more than two points.

Although not state-of-the-art, our systems perform very honorably compared with other single systems in this shared task and pave quite an interesting way for further work. In the remainder of this paper, we present the parsers and their extensions for building graphs; we then present our syntactic features and discuss our results.

2 Systems Description

Shift-reduce transition-based parsers essentially rely on *configurations* formed of a stack and a buffer, with stack transitions used to go from a configuration to the next one, until reaching a final configuration. Following Kübler et al. (2009), we define a configuration by $c = (\sigma, \beta, A)$ where σ denotes a stack of words w_i, β a buffer of words, and A a set of dependency arcs of the form (w_i, r, w_j), with w_i the head, w_j the dependent, and r a label in some set R.

However, despite their overall similarities, transition-based systems may differ on many aspects, such as the exact definition of the configurations, the set of transitions extracted from the configurations, the way the search space is explored (at parsing and training time), the set of features, the way the transition weights are learned and ap-

$(\sigma, w_i\|\beta, A)$	$\vdash$	$(\sigma\|w_i, \beta, A)$	(shift)	BOTH
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_i, \beta, A \cup (w_i, r, w_j))$	(left-reduce)	S&T PARSER
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_j, \beta, A \cup (w_j, r, w_i))$	(right-reduce)	S&T PARSER
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_j\|w_i, \beta, A \cup (w_i, r, w_j))$	(left-attach)	BOTH
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_j, w_i\|\beta, A \cup (w_j, r, w_i))$	(right-attach)	BOTH
$(\sigma\|w_i, \beta, A)$	$\vdash$	(σ, β, A)	(pop0)	BOTH
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_i, \beta, A)$	(pop1)	DYALOG-SR
$(\sigma\|w_j\|w_i, \beta, A)$	$\vdash$	$(\sigma\|w_i\|w_j, \beta, A)$	(swap)	DYALOG-SR

Figure 1: An extended set of transitions for building dependency graphs.

plied, etc.

For various reasons, we started our experiments with two rather different transition-based parsers, which have finally converged on several aspects. In particular, the main convergence concerns the set of transitions needed to parse the three proposed annotation schemes. To be able to attach zero, one, or more heads to a word, it is necessary to clearly dissociate the addition of a dependency from the reduction of a word (i.e. its removal from the stack). Following Sagae and Tsujii (2008), as shown in Figure 1, beside the usual shift and reduce transitions of the *arc-standard* strategy, we introduced the new left and right attach actions for adding new dependencies (while keeping the dependent on the stack) and two reduce `pop0` and `pop1` actions to remove a word from the stack after attachement of its dependents. All transitions adding an edge should also satisfy the condition that the new edge does not create a cycle or multiple edges between the same pair of nodes. It is worth noting that the pop actions may also be used to remove words with no heads.

2.1 Sagae & Tsujii's DAG Parser

Our first parsing system is a partial rewrite, with several extensions, of the Sagae and Tsujii (2008) DAG parser (henceforth S&T PARSER). We modified it to handle dependency graphs, in particular non-governed words using `pop0` transitions. This new transition removes the topmost stack element when all its dependents have been attached (through attach or reduce transitions). Thus, we can handle partially connected graphs, since a word can be discarded when it has no incoming arc.

We used two different learning algorithms: (i) the averaged perceptron because of its good balance between training time and performance (Daume, 2006), (ii) the logistic regression model (maximum entropy (Ratnaparkhi, 1997)). For the latter, we used the truncated gradient optimization (Langford et al., 2009), implemented in Classias (Okazaki, 2009), in order to estimate the parameters. These algorithms have been used interchangeably to test their performance in terms of F-score. But the difference was negligeable in general.

2.2 DYALOG-SR

Our second parsing system is DYALOG-SR (Villemonte De La Clergerie, 2013), which has been developed to participate to the SPMRL'13 shared task. Coded on top of tabular logic programming system DYALOG, it implements a transition-based parser relying on dynamic programming techniques, beams, and an averaged structured perceptron, following ideas from (Huang and Sagae, 2010; Goldberg et al., 2013).

It was initially designed to follow an *arc-standard* parsing strategy, relying on shift and left/right reduce transitions. To deal with dependency graphs and non governed words, we first added the two `attach` transitions and the `pop0` transition. But because there exist some overlap between the reduce and attach transitions leading to some spurious ambiguities, we finally decided to remove the left/right reduce transitions and to complete with the `pop1` transition. In order to handle some cases of non-projectivty with minimal modifications of the system, we also added a `swap` transition. The parsing strategy is now closer to the *arc-eager* one, with an oracle suggesting to attach as soon as possible.

2.3 Tree Approximations

In order to stack several dependency parsers, we needed to transform our graphs into trees. We report here the algorithms we used.

The first one uses a simple strategy. For nodes with multiple incoming edges, we keep the longest incoming edge. Singleton nodes (with no head) are attached with a `_void_`-labeled edge (by decreasing priority) to the immediately adjacent

Word_{σ_1}	Lemma_{σ_1}	POS_{σ_1}
$\text{leftPOS}_{\sigma_1}$	$\text{rightPOS}_{\sigma_1}$	$\text{leftLabel}_{\sigma_1}$
$\text{rightLabel}_{\sigma_1}$	Word_{σ_2}	Lemma_{σ_2}
POS_{σ_2}	$\text{leftPOS}_{\sigma_2}$	$\text{rightPOS}_{\sigma_2}$
$\text{leftLabel}_{\sigma_2}$	$\text{rightLabel}_{\sigma_2}$	Word_{σ_3}
POS_{σ_3}	Word_{β_1}	Lemma_{β_1}
POS_{β_1}	Word_{β_2}	Lemma_{β_2}
POS_{β_2}	POS_{β_3}	$a\ d_{12}\ d'_{11}$

Table 1: Baseline features for S&T PARSER.

node N, or the virtual root node (token 0). This strategy already improves over the baseline, provided by the task organisers, on the PCEDT by 5 points.

The second algorithm tries to preserve more edges: when it is possible, the deletion of a reentrant edge is replaced by reversing its direction and changing its label l into $<l$. We do this for nodes with no incoming edges by reversing the longest edge only if this action does not create cycles. The number of labels increases, but many more edges are kept, leading to better results on DM and PAS corpora.

3 Feature Engineering

3.1 Closed Track

For S&T PARSER we define Word_{β_i} (resp. Lemma_{β_i} and POS_{β_i}) as the word (resp. lemma and part-of-speech) at position i in the queue. The same goes for σ_i, which is the position i in the stack. Let $d_{i,j}$ be the distance between Word_{σ_i} and Word_{σ_j}. We also define $d'_{i,j}$, the distance between Word_{β_i} and Word_{σ_j}. In addition, we define $\text{leftPOS}_{\sigma_i}$ (resp. $\text{leftLabel}_{\sigma_i}$) the part-of-speech (resp. the label if any) of the word immediately at the left handside of σ_i, and the same goes for $\text{rightPOS}_{\sigma_i}$ (resp. $\text{rightLabel}_{\sigma_i}$). Finally, a is the previous predicted action by the parser. Table 1 reports our baseline features.

For DYALOG-SR we have the following *lexical features* `lex`, `lemma`, `cat`, and morphosyntactic `mstag`. They apply to next unread word (`*I`, say `lemmaI`), the three next lookahead words (`*I2` to `*I4`), and (when present) to the 3 stack elements (`*0` to `*2`), their two leftmost and rightmost children (*before* `b[01]*[012]` and *after* `a[01]*[012]`). We have *dependency features* such as the labels of the two leftmost and rightmost edges (`[ab][01]label[012]`), the left and right *valency* (number of dependency, `[ab]v[012]`) and *domains* (set of dependency labels, `[ab]d[012]`). Finally, we have 3 (discretized) *distance features* between the next word and the stack elements (`delta[01]`) and between the two topmost stack elements (`delta01`). Most feature values are atomic (either numerical or symbolic), but they can also be (recursively) a list of values, for instance for the `mstag` and *domain* features. For dealing with graphs, features were added about the incoming edges to the 3 topmost stack elements, similar to valency (ngov[012]) and domain (`gov[012]`). For the PCEDT scheme, because of the high number of dependency labels, the 30 most unfrequent ones were replaced by a generic label when used as feature value.

Besides, for the PCEDT and DM corpora, static and dynamic guiding features have been tried for DYALOG-SR, provided by MATE (Bohnet, 2010) (trained on versions of these corpora projected to trees, using a 10-fold cross validation). The two static features `mate_label` and `mate_distance` are attached to each token h, indicating the label and the relative distance to its governor d (if any). At runtime, dynamic features are also added relative to the current configuration: if a semantic dependency (h, l, d) has been predicted by MATE, and the topmost 2 stack elements are either (h, d) or (d, h), a feature suggesting a left or right attachment for l is added.

We did the same for S&T PARSER, except that we used a simple but efficient hack: instead of keeping the labels predicted by our parser, we replaced them by MATE predictions whenever it was possible.

3.2 Open Track

For this track, we combined the previously described features (but the MATE-related ones) with various lexical and syntactic features, our intuition being that syntax and semantic are interdependent, and that syntactic features should therefore help semantic parsing. In particular, we have considered the following bits of information.

Unsupervized Brown clusters To reduce lexical sparsity, we extracted 1,000 clusters from the BNC (Leech, 1992) preprocessed following Wagner et al. (2007). We extended them with capitalization, digit features and 3 letters suffix signatures, leading to a vocabulary size reduced by half.

Constituent tree fragments They were part of the companion data provided by the organizers.

They consist of fragments of the syntactic trees and can be used either as enhanced parts of speech or as features.

Spinal elementary trees A full set of parses was reconstructed from the tree fragments. Then we extracted a *spine grammar* (Seddah, 2010), using the head percolation table of the Bikel (2002) parser, slightly modified to avoid determiners to be marked as head in some configurations.

Predicted MATE dependencies Also provided in the companion data, they consist in the parses built by the MATE parsers, trained on the Stanford dependency version of the PTB. We combined the labels with a distance $\delta = t - h$ where t is the token number and h the head number.

Constituent head paths Inspired by Björkelund et al. (2013), we used the MATE dependencies to extract the shortest path between a token and its lexical head and included the path length (in terms of traversed nodes) as feature.

	Tree frag.	**MATE labels+δ**	**Spines trees**	**Head Paths**
Train	648	1305	637	27,670
Dev	272	742	265	3,320
Test	273	731	268	2,389

Table 2: Syntactic features statistics.

4 Results and Discussion

We present here the results on section 21 (test set)[1] for both systems. We report in Table 3, the different runs we submitted for the final evaluation of the shared task. We also report improvements between the two tracks.

Both systems show relatively close F-measures, with correct results on every corpus. If we compare the results more precisely, we observe that in general, DYALOG-SR tends to behave better for the unlabeled metrics. Its main weakness is on MRS scheme, for both tracks.[2]

On the other hand, it is worth noting that syntactic features greatly improve semantic parsing. In fact, we report in Figure 2(a) the improvement of the five most frequent labels and, in Figure 2(b), the five best improved labels with a frequency over 0.5% in the training set, which represent 95% of the edges in the DM Corpus. As we can see, syntactic information allow the systems to perform better on coordination structures and to reduce ambiguity between modifiers and verbal arguments (such as the *ARG3* label).

We observed the same behaviour on the PAS corpus, which contains also predicate-argument structures. For PCEDT, the results show that syntactic features give only small improvements, but the corpus is harder because of a large set of labels and is closer to syntactic structures than the two others.

Of course, we only scratched the surface with our experiments and we plan to further investigate the impact of syntactic information during semantic parsing. We especially plan to explore the deep parsing of French, thanks to the recent release of the Deep Sequoia Treebank (Candito et al., 2014).

5 Conclusion

In this paper, we presented our results on the task 8 of the *SemEval-2014 Task on Broad-Coverage Semantic Dependency Parsing*. Even though the results do not reach state-of-the-art, they compare favorably with other single systems and show that syntactic features can be efficiently used for semantic parsing.

In future work, we will continue to investigate this idea, by combining with more complex systems and more efficient machine learning techniques, we are convinced that we can come closer to state of the art results. and that syntax is the key for better semantic parsing.

Acknowledgments

We warmly thank Kenji Sagae for making his parser's code available and kindly answering our questions.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a Treebank for French. In *Treebanks*

[1]Dev set results are available online at http://goo.gl/w3XcpW.

[2]The main and still unexplained problem of DYALOG-SR was that using larger beams has no impact, and often a negative one, when using the attach and pop transitions. Except for PAS and PCEDT where a beam of size 4 worked best for the open track, all other results were obtained for beams of size 1. This situation is in total contradiction with the large impact of beam previously observed for the arc standard strategy during the SPMRL'13 shared task and during experiments led on the French TreeBank (Abeillé et al., 2003) (FTB). Late experiments on the FTB using the attach and pop actions (but delaying attachments as long as possible) has confirmed a problem with beams, even if less visible. We are still investigating why the use of the attach transitions and/or of the pop transitions seems to be incompatible with beams.

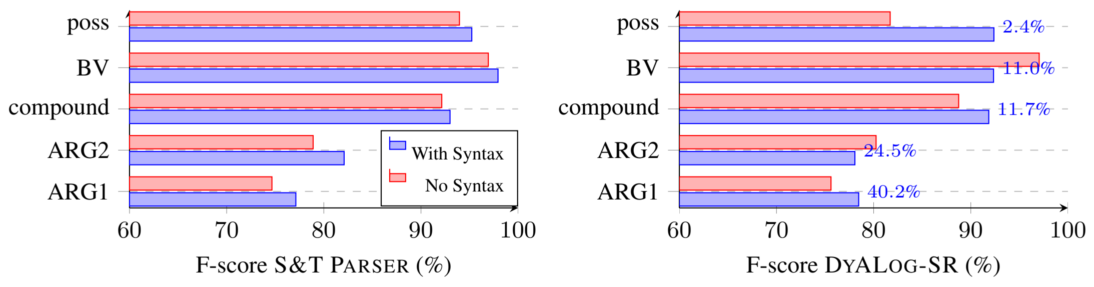
Closed track

PCEDT	LF	UF
Peking - Best	76.28	89.19
S&T Parser b5	67.83	80.86
DyALog-SR b1	67.81	81.23
DM (MRS)		
Peking - Best	89.40	90.82
S&T Parser b5	78.44	80.88
DyALog-SR b1	78.32	81.85
PAS (ENJU)		
Peking - Best	92.04	93.13
S&T Parser b5	82.44	84.41
DyALog-SR b1	84.16	86.09

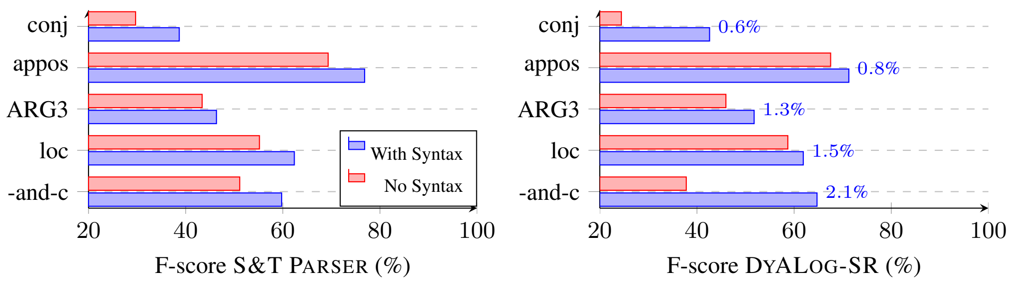
Open track

PCEDT	LF		UF	
Priberam - Best	77.90		89.03	
S&T Parser b5	69.20	+1.37	82.68	+1.86
DyALog-SR b4	69.58	+1.77	84.80	+3.77
DM (MRS)				
Priberam - Best	89.16		90.32	
S&T Parser b5	81.46	+3.02	83.68	+2.80
DyALog-SR b1	79.71	+1.39	81.97	+0.12
PAS (ENJU)				
Priberam - Best	91.76		92.81	
S&T Parser b5	84.97	+2.53	86.64	+2.23
DyALog-SR b4	85.58	+1.42	86.98	+0.87

Table 3: Results on section 21 (test) of the PTB for closed and open track.



(a) the 5 most frequent labels



(b) the 5 best improved labels (edges frequency above 0.5 % in the training set)

Figure 2: Improvement with syntactic features for DM (test) corpus.
(numbers indicate edge frequency in training set)

: Building and Using Parsed Corpora, pages 165–188. Springer.

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the second international conference on Human Language Technology Research*, pages 178–182. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Anders Björkelund, Ozlem Cetinoglu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA.

Marie Candito, Guy Perrier, Bruno Guillaume, Corentin Ribeyre, Karën Fort, Djamé Seddah, and Éric De La Clergerie. 2014. Deep Syntax Annotation of the Sequoia French Treebank. In *International Conference on Language Resources and Evaluation (LREC)*, Reykjavik, Islande, May.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Meeting of the ACL*.

Harold Charles Daume. 2006. *Practical structured learning techniques for natural language processing*. Ph.D. thesis, University of Southern California.

Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation of beam-search incremental parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sophia, Bulgaria, August.

Jan Hajic, Jarmila Panevová, Eva Hajicová, Petr Sgall, Petr Pajas, Jan Štepánek, Jiří Havelka, Marie Mikulová, Zdenek Zabokrtskỳ, and Magda Ševcıková Razımová. 2006. Prague dependency treebank 2.0. *CD-ROM, Linguistic Data Consortium, LDC Catalog No.: LDC2006T01, Philadelphia*, 98.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086. Association for Computational Linguistics.

Angelina Ivanova, Stephan Oepen, Lilja Øvrelid, and Dan Flickinger. 2012. Who did what to whom?: A contrastive study of syntacto-semantic dependencies. In *Proceedings of the sixth linguistic annotation workshop*, pages 2–11.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool Publishers.

John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(777-801):65.

Geoffrey Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep Linguistic Analysis for the Accurate Identification of Predicate-Argument Relations. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2004)*, pages 1392–1397, Geneva, Switzerland.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic, June.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, Dublin, Ireland.

Naoaki Okazaki. 2009. Classias: A collection of machine learning algorithms for classification.

Adwait Ratnaparkhi. 1997. A simple introduction to maximum entropy models for natural language processing. *IRCS Technical Reports Series*, page 81.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760, Manchester, UK, August. Coling 2008 Organizing Committee.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Éric Villemonte De La Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October.

Djamé Seddah. 2010. Exploring the spinal-stig model for parsing french. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may. European Language Resources Association (ELRA).

Éric Villemonte De La Clergerie. 2013. Exploring beam-based shift-reduce dependency parsing with DyALog: Results from the SPMRL 2013 shared task. In *4th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL'2013)*, Seattle, États-Unis.

Joachim Wagner, Djamé Seddah, Jennifer Foster, and Josef Van Genabith. 2007. C-structures and F-structures for the British National Corpus. In *Proceedings of the Twelfth International Lexical Functional Grammar Conference*. Citeseer.