Improving a symbolic parser through partially supervised learning

Éric Villemonte de la Clergerie
INRIA - Rocquencourt - B.P. 105
78153 Le Chesnay Cedex, FRANCE
Eric.De_La_Clergerie@inria.fr

Abstract

Recently, several statistical parsers have been trained and evaluated on the dependency version of the French TreeBank (FTB). However, older symbolic parsers still exist, including FRMG, a wide coverage TAG parser. It is interesting to compare these different parsers, based on very different approaches, and explore the possibilities of hybridization. In particular, we explore the use of partially supervised learning techniques to improve the performances of FRMG to the levels reached by the statistical parsers.

1 Introduction

Most stochastic parsers are trained and evaluated on the same source treebank (for instance the Penn TreeBank), which, by definition, avoid all problems related to differences between the structures returned by the parsers and those present in the treebank. Some symbolic or hybrid parsers are evaluated on a treebank specifically designed for their underlying formalism, possibly by converting and hand-correcting the treebank from some other annotation scheme (as done in (Hockenmaier and Steedman, 2007)). Besides the cost of the operation, an issue concerns the comparison with other parsers. By contrast, the most problematic case remains the evaluation of a parser on an unrelated treebank and scheme (Sagae et al., 2008).

This situation arose for French with the recent emergence of several statistical parsers trained and evaluated on the French TreeBank (FTB) (Abeillé et al., 2003), in particular under its dependency version (Candito et al., 2010b) represented in CONLL format (Nivre et al., 2007). On the other hand, older parsing systems still exist for French, most of them not based on statistical approaches and not related to FTB. For instance, FRMG is a wide coverage symbolic parser for French (de La Clergerie, 2005), based on Tree Adjoining Grammars (TAGs), that has already participated in several parsing campaigns for French. It was important to be able to compare it with statistical parsers on their native treebank, but also possibly to extend the comparison for other treebanks.

A first necessary step in this direction was a conversion from FRMG's native dependency scheme into FTB's dependency scheme, a tedious task highlighting the differences in design at all levels (segmentation, parts of speech, representation of the syntactic phenomena, etc.). A preliminary evaluation has shown that accuracy is good, but largely below the scores reached by the statistical parsers.

A challenge was then to explore if training on the FTB could be used to improve the accuracy of a symbolic parser like FRMG. However, the main difficulty arises from the fact that FTB's dependency scheme has little in common with FRMG's underlying grammar, and that no reverse conversion from FTB to FRMG structures is available. Such a conversion could be investigated but would surely be difficult to develop. Instead, we tried to exploit directly FTB data, using only very minimal assumptions, nevertheless leading to important gains and results close to those obtained by the statistical parsers. The interest is that the technique should be easily adaptable for training data with different annotation schemes. Furthermore, our motivation was not just to improve the performances on the FTB and for the annotation scheme of FTB, for instance by training a reranker (as often done for domain adaptation), but to exploit the FTB to achieve global improvement over all kinds of corpora and for FRMG native annotation scheme.

Section 2 provides some background about FRMG. We expose in Section 3 how partially supervised learning may be used to improve its performances. Section 4 briefly presents the French TreeBank and several other corpora used for training and evaluation. Evaluation results are presented and discussed in Section 5 with a preliminary analysis of the differences between FRMG and the other statistical parsers.

2 FRMG, a symbolic TAG grammar

FRMG (de La Clergerie, 2005) denotes (a) a French meta-grammar; (b) a TAG grammar (Joshi et al., 1975) generated from the meta-grammar; and (c) a chart-like parser compiled from the grammar. As a parser, FRMG parses DAGs of words, built with SXPIPE (Sagot and Boullier, 2008), keeping all potential segmentation ambiguities and with no prior tagging. The parser tries to get *full parses* covering the whole sentence, possibly relaxing some constraints (such as number agreement between a subject and its verb); if not possible, it switches to a *robust mode* looking for a sequence of *partial parses* to cover the sentence.

All answers are returned as shared TAG derivation forests, which are then converted into dependency shared forests, using the anchors of the elementary trees as sources and targets of the dependencies. Some elementary trees being not anchored, pseudo empty words are introduced to serve as source or target nodes. However, in most cases, by a simple transformation, it is possible to reroot all edges related to these pseudo anchors to one of their lexical child.

Finally, the dependency forests are disambiguated using heuristic rules to get a tree. The local *edge rules* assign a positive or negative weight to an edge e, given information provided by e (form/lemma/category/...for the source and target nodes, edge label and type, anchored trees, ...), by neighbouring edges, and, sometimes, by competing edges. A few other *regional rules* assign a weight to a governor node G, given a set of children edges forming a valid derivation from G. The disambiguation algorithm uses dynamic programming techniques to sum the weights and to return the best (possibly non-projective) dependency tree, with maximal weight.

Several conversion schemes may be applied on FRMG's native dependency trees. A recent one returns dependency structures following the annotation scheme used by the dependency version of the French TreeBank and represented using the column-based CONLL format (Nivre et al., 2007). The conversion process relies on a 2-stage transformation system, with constraints on edges used to handle non-local edge propagation, as formalized in (Ribeyre et al., 2012). Figure 1 illustrates the native FRMG's dependency structure (top) and, on the lower side, its conversion to FTB's dependency scheme (bottom). One may observe differences between the two dependency trees, in particular with a (non-local) displacement of the root node. It may be noted that FTB's scheme only considers projective trees, but that the conversion process is not perfect and may return non projective trees, as shown in Figure 1 for the `p_obj` edge.

Let's also mention an older conversion process from FRMG dependency scheme to the EASy/Passage scheme, an hybrid constituency/dependency annotation scheme used for the first French parsing evaluation campaigns (Paroubek et al., 2009). This scheme is based on a set of 6 kinds of chunks and 14 kinds of dependencies.

3 Partially supervised learning

The set of weights attached to the rules may be seen as a statistical model, initially tailored by hand, through trials. It is tempting to use training data, provided by a treebank, and machine learning techniques to improve this model. However, in our case, the "annotation schemes" for the training data (FTB) and for FRMG are distinct. In other words, the training dependency trees cannot be immediately used as oracles as done in most supervised learning approaches, including well-known perceptron ones. Still, even partial information extracted from the training data may help, using partially supervised learning techniques.

Figure 2 shows the resulting process flow. Learning is done, using the disambiguated dependency trees produced by FRMG on training sentences, with (partial) information about the discarded alternatives. The resulting statistical model may then be used to guide disambiguation, and be improved through iterations. Actually, this simple process may be completed with the construction and use of (imperfect) oracles adapted to FRMG. The learning component can produce such an oracle but can also exploit it. Even better, the oracle can be directly used to guide the disam-

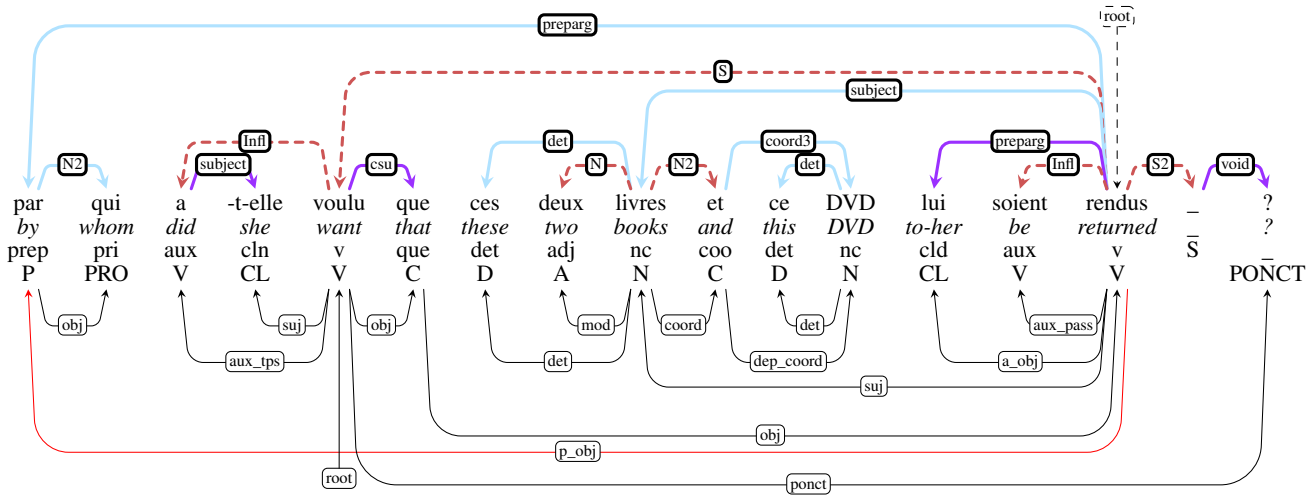


Figure 1: Sample of disambiguated FRMG output, without conversion (top) and with FTB conversion (bottom)

biguation process. Again, by iterating the process, one can hopefully get an excellent oracle for the learning component, useful to get better models.

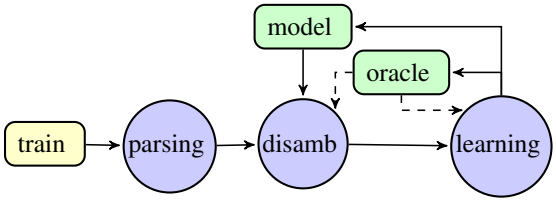


Figure 2: Parsing and partially supervised learning with imperfect oracles

The minimal information we have at the level of a word w is the knowledge that the its incoming dependency d proposed by conversion is correct or not, leading to 4 situations as summarized in Table 1.

	d is correct	d is not correct
selected D	favor r	penalize r'
competitor D'	penalize r'	??

Table 1: Handling weight updates for rules

For the FTB conversion, when d is correct, we can generally assume that the FRMG incoming dependency D for w is also correct and that disambiguation is correct in selecting D. We can then consider than any edge disambiguation rule r applicable on D should then be favored by (slightly) increasing its weight and rules applying on competitors of D should see their weight decrease (to reinforce the non-selection of D).

On the other hand, when d is not correct, we should penalize the rules r applying on D and try to favor some competitor D' of D (and favor the rules r' applying to D'). However, we do not know which competitor should be selected, except in cases where there is only one possible choice. By default, we assume that all competitors have equal chance to be the correct choice and favor/penalize in proportion their rules. If we have n competitors, we can say that it is a bad choice not to keep D' in $\frac{1}{n}$ cases (and should favor rules r') and it is a bad choice to keep D' in $\frac{n-1}{n}$ cases (and should penalize rules r'). So, practically, the dependency D' falling in the problematic case is distributed between the keep/bad case (with weight $\frac{1}{n}$) and the discard/bad case (with weight $\frac{n-1}{n}$). These proportions may be altered if we have more precise information about the competitors, provided by an oracle (as hinted in Figure 2), weights, ranks, or other elements. For instance, if we known that d is not correct but has the right dependency label or the right governor, we use this piece of information to discard some competitors and rerank the remaining ones.

Of course, the update strategy for the problematic case will fail in several occasions. For instance, maybe D is the right FRMG dependency to keep, but the conversion process is incorrect and produces a bad

FTB dependency d. Maybe FRMG is incomplete and has no correct source dependency D. Finally, maybe d (with target word w) derives from some source dependency $D_{w'}$ for some other target word w'. We assume that these cases remain limited and that improving edge selection for the easy cases will then guide edge selection for the more complex cases.

The learning algorithm could be used online, adjusting the weights when processing a sentence. However, we have only implemented an offline version where the weights are updated after considering all training sentences (but discarding some long sentences and sentences with low accuracy scores).

More formally, given the parses for the training sentences, for any edge disambiguation rule r and value tuple $\mathbf{v}$ for a feature template $\mathbf{f}$, we compute the number $n_{r,\mathbf{f}=\mathbf{v}}$ of occurrences of r in context $\mathbf{f} = \mathbf{v}$, and $\text{keep}^{\text{ok}}_{r,\mathbf{f}=\mathbf{v}}$ the number of occurrences where the edge was selected and it was a correct choice. Similarly, but taking into account the above-mentioned redistribution, we compute $\text{discard}^{\text{ok}}_{r,\mathbf{f}=\mathbf{v}}$, $\text{keep}^{\text{bad}}_{r,\mathbf{f}=\mathbf{v}}$, and $\text{discard}^{\text{bad}}_{r,\mathbf{f}=\mathbf{v}}$.

These figures are used to compute an adjustment $\delta_{r,\mathbf{f}=\mathbf{v}}$ added to the base weight w_r of r for context $\mathbf{f} = \mathbf{v}$, using Eq (1), where θ denotes a *temperature*:

$$(1)\quad \delta_{r,\mathbf{f}=\mathbf{v}} = \theta.a_{r,\mathbf{f}=\mathbf{v}}.\begin{cases} \text{discard}^{\text{bad}}_{r,\mathbf{f}=\mathbf{v}} & \text{if } a_{r,\mathbf{f}=\mathbf{v}} > 0 \\ \text{keep}^{\text{bad}}_{r,\mathbf{f}=\mathbf{v}} & \text{otherwise} \end{cases}$$

The $a_{r,\mathbf{f}=\mathbf{v}}$ factor is related to the direction and force of the expected change[1], being positive when selecting an edge thanks to r tends to be a good choice, and negative otherwise (when the edge should rather be discarded), as expressed in the following formula:

$$a_{r,\mathbf{f}=\mathbf{v}} = \frac{\text{keep}^{\text{ok}}}{\text{keep}^{\text{ok}} + \text{keep}^{\text{bad}}} - \frac{\text{discard}^{\text{ok}}}{\text{discard}^{\text{ok}} + \text{discard}^{\text{bad}}}$$

The last factor in Eq (1) is the number of edges whose status (selected or discarded) should ideally change.

The process is iterated, reducing the temperature at each step, and we keep the best run. At each iteration, the edges found to be correctly kept or discarded are added to an oracle for the next iteration.

[1]It may be noted that the interpretation of $a_{r,\mathbf{f}=\mathbf{v}}$ may sometimes be unclear, when both $\text{keep}^{\text{ok}}_{r,\mathbf{f}=\mathbf{v}}$ and $\text{discard}^{\text{ok}}_{r,\mathbf{f}=\mathbf{v}}$ are low (i.e., when neither keeping or discarding the corresponding edges is a good choice). We believe that these cases signal problems in the conversion process or the grammar.

We use standard features such as form, lemma, pos, suffixes, sub-categorization information, morphosyntactic features, anchored TAG trees for words (dependency heads and targets, plus adjacent words); and dependency distances, direction, type, label, and rank for the current dependency and possibly for its parent. For smoothing and out-of-domain adaptation, we add a *cluster* feature attached to forms and extracted from a large raw textual corpus using Brown clustering (Liang, 2005). It may noted that the name of a disambiguation rule may be considered as the value of a `rule` feature. Each feature template includes the label and type for the current FRMG dependency.

It seems possible to extend the proposed learning mechanism to adjust the weight of the regional rules by considering (second-order) features over pairs of adjacent sibling edges (for a same derivation). However, preliminary experiments have shown an explosion of the number of such pairs, and no real gain.

4 The corpora

The learning method was tried on the French TreeBank(Abeillé et al., 2003), a journalistic corpus of 12,351 sentences, annotated in morphology and constituency with the Penn TreeBank format, and then automatically converted into projective dependency trees, represented in the CONLL format (Candito et al., 2010a). For training and benchmarking, the treebank is split into three parts, as summarized in Table 2.

		Cover.	Time (s)	
Corpus	**#sent.**	**(%)**	**Avg**	**Median**
FTB train	9,881	95.9	1.04	0.26
FTB dev	1,235	96.1	0.88	0.30
FTB test	1,235	94.9	0.85	0.30
Sequoia	3,204	95.1	1.53	0.17
EASyDev	3,879	87.2	0.87	0.14

Table 2: General information on FTB and other corpora

To analyze the evolution of the performances, we also consider two other corpora. The **Sequoia** corpus (Candito and Seddah, 2012) also uses the FTB dependency scheme (at least for its version 3), but covers several styles of documents (medical, encyclopedic, journalistic, and transcription of political discourses). The **EASyDev** corpus also covers various styles (journalistic, literacy, medical, mail, speech, ...), but was

annotated following the EASy/Passage scheme for evaluation campaigns (Paroubek et al., 2006).

Table 2 shows that coverage (by full parses) is high for all corpora (slightly lower for **EASyDev** because of the mail and speech sub-corpora). Average time per sentence is relatively high but, as suggested by the much lower median times, this is largely due to a few long sentences and due to a large timeout.

5 Results and discussions

Table 3 shows evaluation results for different versions of FRMG on each corpus. On **FTB** and **Sequoia**, we use Labelled Attachment Scores (LAS) without taking into account punctuation, and, on **EASyDev**, F1-measure on the dependencies[2]. The *init* system corresponds to a baseline version of FRMG with a basic set of rules and hand-tailored weights. The *+restr* version of FRMG adds *restriction rules*, exploiting attachment preferences and word (semantic) similarities extracted from a very large corpus parsed with FRMG, using Harris distributional hypothesis[3]. This version shows that unsurpervised learning methods already improve significantly the performances of a symbolic parser like FRMG for all corpora. The *+tuning* version of FRMG keeps the restriction rules and adds the partially supervised learning method. We observe large improvements on the FTB dev and test parts (between 4 and 5 points), but also on **Sequoia** (almost 3 points) on different styles of documents. We also get similar gains on **EASyDev**, again for a large diversity of styles, and, more interestingly, for a different annotation scheme and evaluation metric.

The bottom part of Table 3 lists the accuracy of 3 statistical parsers on FTB as reported in (Candito et al., 2010b). The Berkeley parser (BKY) is a constituent-based parser whose parses are then converted into FTB dependencies (using the same tool used to convert the FTB). MALT parser is a greedy transition-based parser while MST (*maximum spanning tree*) globally extracts the best dependency tree from all possible ones. We see that FRMG (with tuning) is better than BKY on the test part (but not on the dev part), close to MALT, and still below MST. Clearly, tuning allows FRMG to be more competitive with statistical parsers even on their native treebank.

	FTB			other corpora	
system	train	dev	test	Sequoia	EASy
init	79.95	80.85	82.08	81.13	65.92
+restr	80.67	81.72	83.01	81.72	66.33
+tuning	86.60	85.98	87.17	84.56	69.23
BKY	–	86.50	86.80	–	–
MALT	–	86.90	87.30	–	–
MST	–	87.50	88.20	–	–

Table 3: Performances of various systems on French data

system	emea-test	ftb-test	loss
BKY (evalb)	80.80	86.00	5.20
FRMG+tuning (LAS)	84.13	87.17	3.04

Table 4: Evolution for an out-of-domain medical corpus

We do not have results for the 3 statistical parsers on the **Sequoia** corpus. However, (Candito and Seddah, 2012) reports some results for Berkeley parser on constituents for the medical part of **Sequoia**, listed in Table 4. The metrics differ, but we observe a loss of 5.2 for BKY and only of 3.04 for FRMG, which tends to confirm the stability of FRMG across domains, possibly due to the constraints of its underlying linguistically-motivated grammar (even if we observe some over-fitting on FTB).

Figure 3 shows the evolution of accuracy on the 3 components of FTB during the learning iterations. We observe that learning is fast with a very strong increase at the first iteration, and a peak generally reached at iterations 3 or 4. As mentioned in Section 3, rule names may be seen as feature values and it is possible to discard them, using only a single *dummy* rule. This *dummy* edge rule checks nothing on the edges but only acts as a default value for the `rule` feature. However, old experiments showed a LAS on FTB dev of 84.31% keeping only a dummy rule and of 85.00% with all rules, which seems to confirm the (global) pertinence of the hand-crafted disambiguation rules. Indeed, these rules are able to consult additional information (about adjacent edges and alternative edges)

[2] F1-measures on chunks are less informative.

[3] We used a 700Mwords corpus composed of AFP news, French Wikipedia, French Wikisource, etc.. The attachment weights are used for handling PP attachments over verbs, nouns, adjectives, but also for relatives over antecedents, or for filling some roles (subject, object, attribute). Similarities between words are used for handling coordination.

not available through the other features. [4]

As suggested in Figure 2, the oracle built by the learning component on FTB train may be used during disambiguation (on FTB train) by setting a very high weight for the edges in the oracle and a very low weight for the others. The disambiguation process is then strongly encouraged to select the edges of the oracle (when possible). Iterating the process, we reach an accuracy of 89.35% on FTB train, an interesting first step in direction of a FRMG version of the FTB[5].

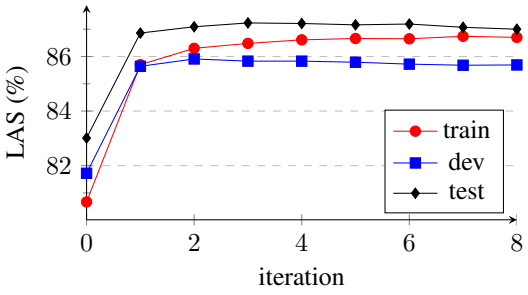


Figure 3: LAS evolution on FTB train per iteration

One reason still explaining the differences between FRMG and the statistical parsers arises from the conversion process to FTB annotation scheme being not perfect. For instance, FRMG and FTB do not use the same list of multi-word expressions, leading to problems of mapping between words and of dependency attachments, in particular for complex prepositions and conjunctions. The segmenter SXPIPE also recognizes named entities such as *Communauté européenne* (*European Community*), *5 millions*, or *Mardi prochain* (*next Tuesday*) as single terms whereas FTB adds internal dependencies for these expressions. During the conversion phase, most of the missing dependencies are added leading to an accuracy of 75.38% on the specific dependencies in FTB train (around 3.5% of all dependencies), still largely below the global accuracy (86.6%). There are also 1259 sentences in FTB train (12.7%) where FRMG produces non-projective trees when FTB expects projective ones.[6]

[4]However, it is clear that some disambiguation rules are redundant with the other features and could be discarded.

[5]The problem is that the treebank would have to be regenerated to follow the evolution of FRMG.

[6]It does not mean that so many FRMG trees are non-projective, just that the conversion builds non-projective trees, because of edge movement. A quick investigation has shown that many cases were related to punctuation attachment.

Then, following (McDonald and Nivre, 2007), we tried to compare the performance of FRMG, MST, and MALT with respect to several properties of the dependencies. Figure 4(a) compares the recall and precision of the systems w.r.t. the distance of the dependencies (with, in background, the number of gold dependencies). We observe that all systems have very close recall scores for small distances, then MST is slightly better, and, at long distance, both MST and MALT are better. On the other hand, FRMG has a much better precision than MALT for long distance dependencies. One may note the specific case of null distance dependencies actually corresponding to root nodes, with lower precision for FRMG. This drop corresponds to the extra root nodes added by FRMG in robust mode when covering a sentence with partial parses.

As shown in Figure 4(b), the recall curves w.r.t. dependency depths are relatively close, with FRMG slightly below for intermediate depths and slightly above for large depths. Again, we observe a precision drop for root nodes (depth=0) which disappears when discarding the sentences in robust mode.

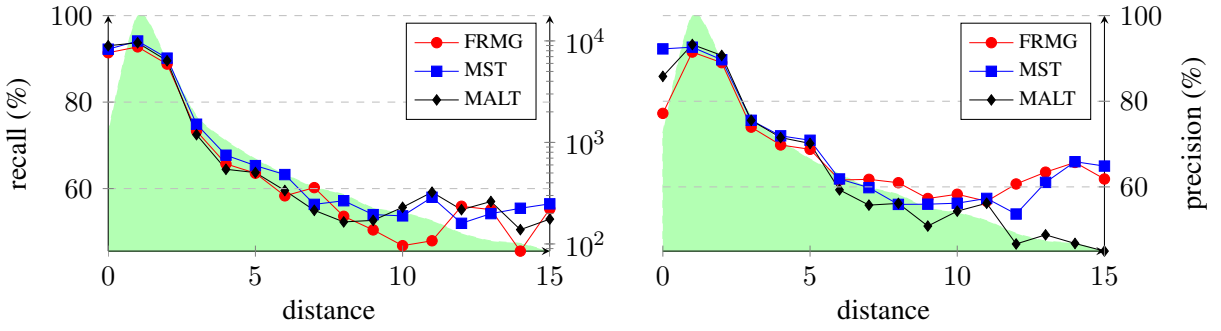
In Figure 4(c), we get again a lower recall for large numbers of sibling edges with, surprisingly, a much higher precision for the same values.

Figure 4(d) compares recall and precision w.r.t. dependency rank[7], with again the lower precision due to the extra root nodes (rank=0) and again a lower recall and higher precision for large absolute ranks.

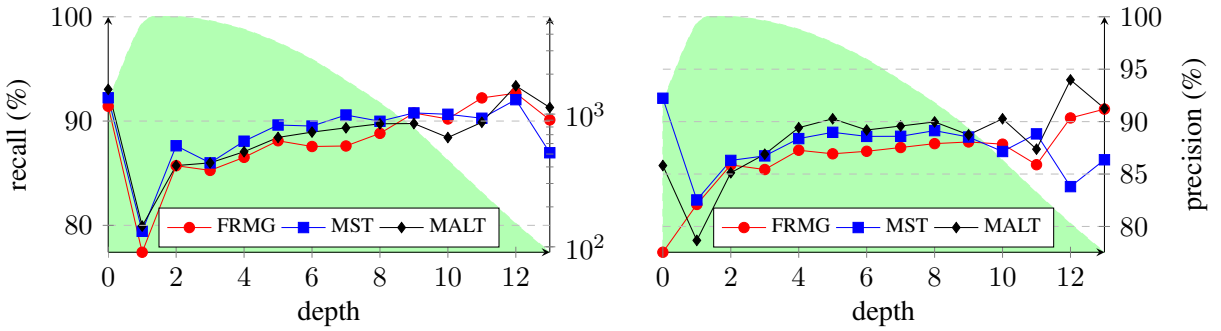
More generally, FRMG tends to behave like MST rather than like MALT. We hypothesize that it reflects than both systems share a more global view of the dependencies, in particular thanks to the domain locality provided by TAGs for FRMG.

Figure 5 shows recall wrt some of the dependency labels. The most striking point is the weak recall for coordination by all systems but, nevertheless, the better score of FRMG. We observe a lower recall of FRMG for some verbal prepositional arguments (`a_obj`, `de_obj`) that may be confused with verb modifiers or attached to a noun or some other verb. Verbal modifiers (`mod`), a category covering many different syntactic phenomena, seem also difficult, partly due to the handling of prepositional attachments. On

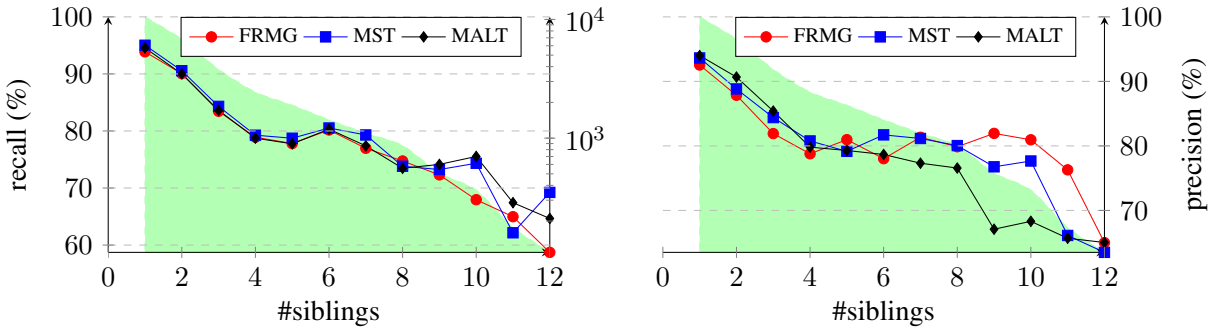
[7]defined as the number of siblings (plus 1) between a dependant and its head, counted positively rightward and negatively leftward.



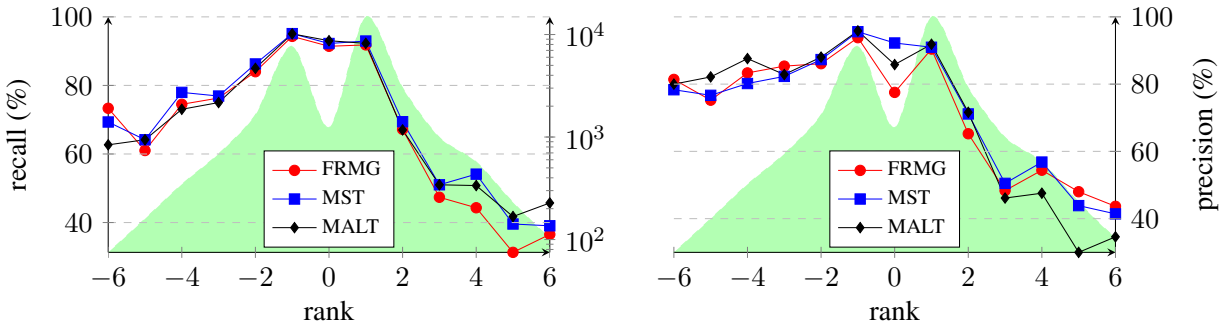
(a) w.r.t. dependency distance



(b) w.r.t. dependency depth



(c) w.r.t. number of siblings



(d) w.r.t. dependency rank

Figure 4: System comparison

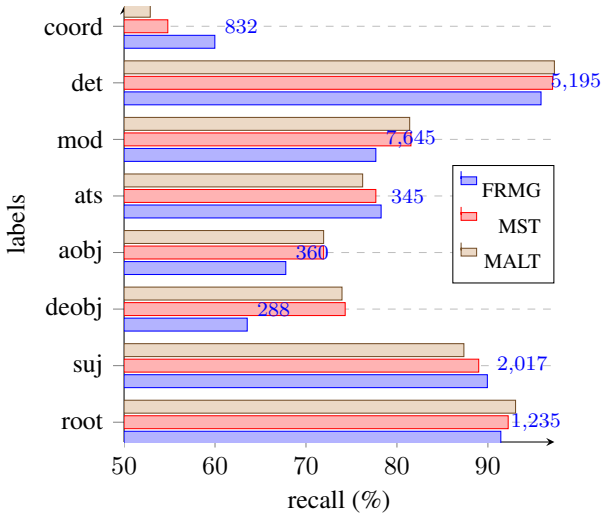


Figure 5: System comparison w.r.t. dependency labels

the other hand, FRMG has a better recall for subjects, possibly because the grammar accepts a large range of positions and realization for subjects.

6 Conclusion

We have presented a new partially supervised learning approach exploiting the information of a training treebank for tuning the disambiguation process of FRMG, a symbolic TAG-based parser. Even considering minimal assumptions for transferring oracle information from the training treebank, we strongly improve accuracy, allowing FRMG to be on par with statistical parsers on their native treebank, namely the French TreeBank. Even if the gains are important, several extensions of the learning algorithm have still to be explored, in particular to build and exploit better oracles, and to incorporate more higher order features, such as sibling features.

The approach explored in this paper, even if tried in the specific context of FRMG, is susceptible to be adapted for other similar contexts, in particular when some imperfect annotation conversion process takes place between a disambiguation process and a training treebank. However, some work remains be done to get a better characterization of the learning algorithm, for instance w.r.t. perceptrons.

We are aware that some of the data collected by the learning algorithm could be used to track problems either in the conversion process or in FRMG grammar (by exploring the cases where neither selecting or discarding an edge seems to be a good choice). We would like to fix these problems, even if most of them seem to have very low frequencies. The conversion process could also be improved by allowing some non-deterministic choices, again controlled by probabilistic features. However, it is not yet clear how we can couple learning for the disambiguation process and learning for the conversion process.

More investigations and comparisons are needed, but some hints suggest that an underlying linguistically-motivated grammar ensures a better robustness w.r.t. document styles and domains. On the other hand, the evaluation shows that the choices made in FRMG to handle lack of full coverage using partial parses should be improved, maybe by using some guiding information provided by a statistical parser to handle the problematic areas in a sentence.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Marie Candito and Djamé Seddah. 2012. Le corpus sequoia: annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012-19e conférence sur le Traitement Automatique des Langues Naturelles*.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010a. Statistical french dependency parsing: treebank conversion and first results. In *Proceedings of the 7th Language Resources and Evaluation Conference (LREC'10)*, La Valette, Malte.

Marie Candito, Joakim Nivre, Pascal Denis, and Enrique Henestroza Anguiano. 2010b. Benchmarking of statistical dependency parsers for french. In *Proceedings of COLING'2010 (poster session)*, Beijing, China.

Éric de La Clergerie. 2005. From metagrammars to factorized TAG/TIG parsers. In *Proceedings of IWPT'05 (poster)*, pages 190–191, Vancouver, Canada.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Aravind K. Joshi, Leon Levy, and Makoto Takahashi. 1975. Tree Adjunct Grammars. *Journal of Computer and System Science 10*, 10(1):136–163.

Percy Liang. 2005. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Ryan T. McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*, pages 122–131.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *The CoNLL 2007 shared task on dependency parsing*.

Patrick Paroubek, Isabelle Robba, Anne Vilnat, and Christelle Ayache. 2006. Data, Annotations and Measures in EASy, the Evaluation Campaign for Parsers of French. In *Proceedings of the 5th international conference on Language Resources and Evaluation (LREC'06)*, Gênes, Italie.

Patrick Paroubek, Éric Villemonte de la Clergerie, Sylvain Loiseau, Anne Vilnat, and Gil Francopoulo. 2009. The PASSAGE syntactic representation. In *7th International Workshop on Treebanks and Linguistic Theories (TLT7)*, Groningen, January.

Corentin Ribeyre, Djamé Seddah, and Éric Villemonte De La Clergerie. 2012. A Linguistically-motivated 2-stage Tree to Graph Transformation. In Chung-Hye Han and Giorgio Satta, editors, *TAG+11 - The 11th International Workshop on Tree Adjoining Grammars and Related Formalisms - 2012*, Paris, France. INRIA.

K. Sagae, Y. Miyao, T. Matsuzaki, and J. Tsujii. 2008. Challenges in mapping of syntactic representations for framework-independent parser evaluation. In *Proceedings of the Workshop on Automated Syntatic Annotations for Interoperable Language Resources at the First International Conference on Global Interoperability for Language Resources (ICGL'08)*, Hong-Kong, January.

Benoît Sagot and Pierre Boullier. 2008. SXPIPE 2 : architecture pour le traitement présyntaxique de corpus bruts. *Traitement Automatique des Langues (T.A.L.)*, 49(2):155–188.

On Different Approaches to Syntactic Analysis Into Bi-Lexical Dependencies
An Empirical Comparison of Direct, PCFG-Based, and HPSG-Based Parsers

Angelina Ivanova[♠], Stephan Oepen[♠♡], Rebecca Dridan[♠], Dan Flickinger[♣], and Lilja Øvrelid[♠]

[♠] University of Oslo, Department of Informatics

[♡] Potsdam University, Department of Linguistics

[♣] Stanford University, Center for the Study of Language and Information

{angelii|oe|rdridan|liljao}@ifi.uio.no, danf@stanford.edu

Abstract

We compare three different approaches to parsing into syntactic, bi-lexical dependencies for English: a 'direct' data-driven dependency parser, a statistical phrase structure parser, and a hybrid, 'deep' grammar-driven parser. The analyses from the latter two are post-converted to bi-lexical dependencies. Through this 'reduction' of all three approaches to syntactic dependency parsers, we determine empirically what performance can be obtained for a common set of dependency types for English, across a broad variety of domains. In doing so, we observe what trade-offs apply along three dimensions, accuracy, efficiency, and resilience to domain variation. Our results suggest that the hand-built grammar in one of our parsers helps in both accuracy and cross-domain performance.

1 Motivation

Bi-lexical dependencies, i.e. binary head–argument relations holding exclusively between lexical units, are widely considered an attractive target representation for syntactic analysis. At the same time, Cer et al. (2010) and Foster et al. (2011), inter alios, have demonstrated that higher dependency accuracies can be obtained by parsing into a phrase structure representation first, and then reducing parse trees into bi-lexical dependencies.[1] Thus, if one is willing to accept pure syntactic dependencies as a viable interface (and evaluation) representation, an experimental setup like the one of Cer et al. (2010) allows the exact experimental comparison of quite different parsing approaches.[2] Existing such studies to date are limited to purely data-driven (or statistical) parsers, i.e. systems where linguistic knowledge is exclusively acquired through supervised machine learning from annotated training data. For English, the venerable Wall Street Journal (WSJ) portion of the Penn Treebank (PTB; Marcus et al., 1993) has been the predominant source of training data, for phrase structure and dependency parsers alike.

Two recent developments make it possible to broaden the range of parsing approaches that can be assessed empirically on the task of deriving bi-lexical syntactic dependencies. Flickinger et al. (2012) make available another annotation layer over the same WSJ text, 'deep' syntacto-semantic analyses in the linguistic framework of Head-Driven Phrase Structure Grammar (HPSG; Pollard & Sag, 1994; Flickinger, 2000). This resource, dubbed DeepBank, is available since late 2012. For the type of HPSG analyses recorded in DeepBank, Zhang and Wang (2009) and Ivanova et al. (2012) define a reduction into bi-lexical syntactic dependencies, which they call Derivation Tree-Derived Dependencies (DT). Through application of the converter of Ivanova et al. (2012) to DeepBank, we can thus obtain a DT-annotated version of the standard WSJ text, to train and test a data-driven dependency and phrase structure parser, respectively, and to compare parsing results to a hybrid, grammar-driven HPSG parser. Furthermore, we can draw on a set of additional corpora annotated in the same HPSG format (and thus amenable to conversion for both phrase structure and dependency parsing), instantiating a comparatively diverse range of domains and genres (Oepen et al., 2004). Adding this data to our setup for additional cross-domain testing, we seek to document not only what trade-offs apply in terms of dependency accuracy vs. parser efficiency, but also how these trade-offs are affected by domain and genre variation, and, more generally, how resilient the different approaches are to variation in parser inputs.

[1] This conversion from one representation of syntax to another is lossy, in the sense of discarding constituency information, hence we consider it a reduction in linguistic detail.

[2] In contrast, much earlier work on cross-framework comparison involved post-processing parser outputs in form *and* content, into a target representation for which gold-standard annotations were available. In § 2 below, we argue that such conversion inevitably introduces blur into the comparison.

2 Related Work

Comparing between parsers from different frameworks has long been an area of active interest, ranging from the original PARSEVAL design (Black et al., 1991), to evaluation against 'formalism-independent' dependency banks (King et al., 2003; Briscoe & Carroll, 2006), to dedicated workshops (Bos et al., 2008). Grammatical Relations (GRs; Briscoe & Carroll, 2006) have been the target of a number of benchmarks, but they require a heuristic mapping from 'native' parser outputs to the target representations for evaluation, which makes results hard to interpret. Clark and Curran (2007) established an upper bound by running the mapping process on gold-standard data, to put into perspective the mapped results from their CCG parser proper. When Miyao et al. (2007) carried out the same experiment for a number of different parsers, they showed that the loss of accuracy due to the mapping process can swamp any actual parser differences. As long as heuristic conversion is required before evaluation, cross-framework comparison inevitably includes a level of fuzziness. An alternative approach is possible when there is enough data available in a particular representation, and conversion (if any) is deterministic. Cer et al. (2010) used Stanford Dependencies (de Marneffe & Manning, 2008) to evaluate a range of statistical parsers. Pre- or post-converting from PTB phrase structure trees to the Stanford dependency scheme, they were able to evaluate a large number of different parsers.

Fowler and Penn (2010) formally proved that a range of Combinatory Categorial Grammars (CCGs) are context-free. They trained the PCFG Berkeley parser on CCGBank, the CCG annotation of the PTB WSJ text (Hockenmaier & Steedman, 2007), advancing the state of the art in terms of supertagging accuracy, PARSEVAL measures, and CCG dependency accuracy. In other words, a specialized CCG parser is not necessarily more accurate than the general-purpose Berkeley parser; this study, however, fails to also take parser efficiency into account.

In related work for Dutch, Plank and van Noord (2010) suggest that, intuitively, one should expected that a grammar-driven system can be more resiliant to domain shifts than a purely data-driven parser. In a contrastive study on parsing into Dutch syntactic dependencies, they substantiated this expectation by showing that their HPSG-based Alpino system performed better and was more resilient to domain variation than data-driven direct dependency parsers.

3 Background: Experimental Setup

In the following, we summarize data and software resources used in our experiments. We also give a brief introduction to the DT syntactic dependency scheme and a comparison to 'mainstream' representations.

DeepBank HPSG analyses in DeepBank are manually selected from the set of parses licensed by the English Resource Grammar (ERG; Flickinger, 2000). Figure 1 shows an example ERG derivation tree, where labels of internal nodes name HPSG constructions (e.g. subject–head or head–complement: sb-hd_mc_c and hd-cmp_u_c, respectively; see below for more details on unary rules). Preterminals are labeled with fine-grained lexical categories, dubbed ERG lexical types, that augment common parts of speech with additional information, for example argument structure or the distinction between count, mass, and proper nouns. In total, the ERG distinguishes about 250 construction types and 1000 lexical types.

DeepBank annotations were created by combining the native ERG parser, dubbed PET (Callmeier, 2002), with a discriminant-based tree selection tool (Carter, 1997; Oepen et al., 2004), thus making it possible for annotators to navigate the large space of possible analyses efficiently, identify and validate the intended reading, and record its full HPSG analysis in the treebank. Owing to this setup, DeepBank in its current version 1.0 lacks analyses for some 15 percent of the WSJ sentences, for which either the ERG parser failed to suggest a set of candidates (within certain bounds on time and memory usage), or the annotators found none of the available parses acceptable.[3] Furthermore, DeepBank annotations to date only comprise the first 21 sections of the PTB WSJ corpus. Following the splits suggested by the DeepBank developers, we train on Sections 0–19, use Section 20 for tuning, and test against Section 21 (abbreviated as WSJ below).[4]

[3] Thus, limitations in the current ERG and PET effectively lead to the exclusion of a tangible percentage of sentences from our training and testing corpora. We discuss methodological ramifications of this setup to our study in § 9 below.

[4] To 'protect' Section 21 as unseen test data, also for the ERG parser, this final section in Version 1.0 of DeepBank was not ex-

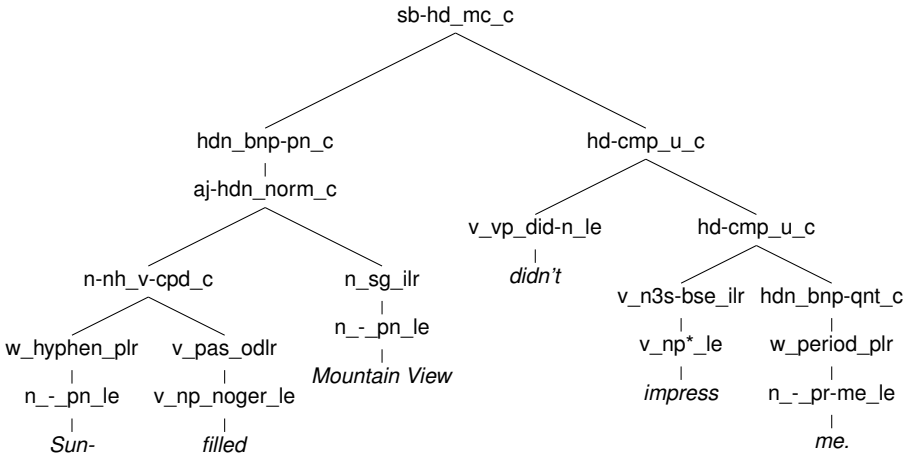


Figure 1: Sample HPSG derivation: construction identifiers label internal nodes, lexical types the preterminals.

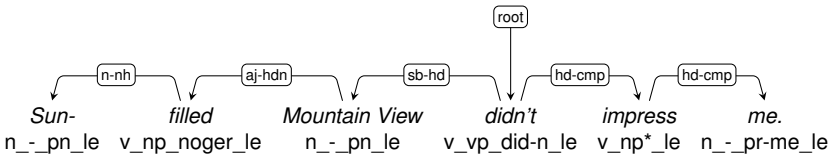


Figure 2: Sample DT bi-lexical dependencies: construction identifiers are generalized at the first underscore.

DT Dependencies As ERG derivations are grounded in a formal theory of grammar that explicitly marks heads, mapping these trees onto bi-lexical dependencies is straightforward (Zhang & Wang, 2009). Ivanova et al. (2012) coin the term DT for ERG Derivation Tree-Derived Dependencies, where they reduce the inventory of some 250 ERG syntactic rules to 48 broad HPSG constructions. The DT syntactic dependency tree for our running example is shown in Figure 2.

To better understand the nature of the DT scheme, Ivanova et al. (2012) offer a quantitative, structural comparison against two pre-existing dependency standards for English, viz. those from the CoNLL dependency parsing competitions (Nivre et al., 2007) and the 'basic' variant of Stanford Dependencies. They observe that the three dependency representations are broadly comparable in granularity and that there are substantial structural correspondences between the schemes. Measured as average Jaccard similarity over unlabeled dependencies, they observe the strongest correspondence between DT and CoNLL (at a Jaccard index of 0.49, compared to 0.32 for DT and Stanford, and 0.43 between CoNLL and Stanford).

posed to its developers until the grammar and disambiguation model were finalized and frozen for this release.

Ivanova et al. (2013) complement this comparison of dependency schemes through an empirical assesment in terms of 'parsability', i.e. accuracy levels available for the different target representations when training and testing a range of state-of-the-art parsers on the same data sets. In their study, the dependency parser of Bohnet and Nivre (2012), henceforth B&N, consistently performs best for all schemes and output configurations. Furthermore, parsability differences between the representations are generally very small.

Based on these observations, we conjecture that DT is as suitable a target representation for parser comparison as any of the others. Furthermore, two linguistic factors add to the attractiveness of DT for our study: it is defined in terms of a formal (and implemented) theory of grammar; and it makes available more fine-grained lexical categories, ERG lexical types, than is common in PTB-derived dependency banks.

Cross-Domain Test Data Another benefit of the DT target representation is the availability of comparatively large and diverse samples of additional test data. The ERG Redwoods Treebank (Oepen et al., 2004) is similar in genealogy and format to DeepBank, comprising corpora from various domains and genres. Although Redwoods counts a total of some 400,000 annotated tokens, we only draw on it for addi-

	Name	Sentences	Tokens	Types
DeepBank	**Train**	33,783	661,451	56,582
	Tune	1,721	34,063	8,964
	WSJ	1,414	27, 515	7,668
Redwoods	CB	608	11,653	3,588
	SC	864	13,696	4,925
	VM	993	7,281	1,007
	WS	520	8,701	2,974

Table 1: Sentence, token, and type counts for data sets.

tional *testing* data. In other words, we do not attempt parser re-training or adaptation against this additional data, but rather test our WSJ-trained parsers on out-of-domain samples from Redwoods. We report on four such test corpora, viz. (a) a software advocacy essay, *The Cathedral and the Bazaar* (CB); (b) a subset of the SemCor portion of the Brown Corpus (SC; Francis & Kucera, 1982); (c) a collection of transcribed, task-oriented spoken dialogues (VM; Wahlster, 2000); and (d) part of the Wikipedia-derived WeScience Corpus (WS; Ytrestøl et al., 2009). Table 1 provides exact sentence, token, and type counts for these data sets.

Tokenization Conventions A relevant peculiarity of the DeepBank and Redwoods annotations in this context is the ERG approach to tokenization. Three aspects in Figure 1 deviate from the widely used PTB conventions: (a) hyphens (and slashes) introduce token boundaries; (b) whitespace in multi-word lexical units (like *ad hoc*, *of course*, or *Mountain View*) does not force token boundaries; and (c) punctuation marks are attached as 'pseudo-affixes' to adjacent words, reflecting the rules of standard orthography. Adolphs et al. (2008) offer some linguistic arguments for this approach to tokenization, but for our purposes it suffices to note that these differences to PTB tokenization may in part counter-balance each other, but do increase the types-per-tokens ratio somewhat. This property of the DeepBank annotations, arguably, makes English look somewhat similar to languages with moderate inflectional morphology. To take advantage of the fine-grained ERG lexical categories, most of our experiments assume ERG tokenization. In two calibration experiments, however, we also investigate the effects of tokenization differences on our parser comparison.

PET: Native HPSG Parsing The parser most commonly used with the ERG is called PET (Callmeier, 2002), a highly engineered chart parser for unification grammars. PET constructs a complete parse forest, using subsumption-based ambiguity factoring (Oepen & Carroll, 2000), and then extracts from the forest n-best lists of complete analyses according to a discriminative parse ranking model (Zhang et al., 2007). For our experiments, we trained the parse ranker on Sections 00–19 of DeepBank and otherwise used the default configuration (which corresponds to the environment used by the DeepBank and Redwoods developers), which is optimized for accuracy. This parser, performing exact inference, we will call ERG$_a$.

In recent work, Dridan (2013) augments ERG parsing with lattice-based sequence labeling over lexical types and lexical rules. Pruning the parse chart prior to forest construction yields greatly improved efficiency at a moderate accuracy loss. Her lexical pruning model is trained on DeepBank 00–19 too, hence compatible with our setup. We include the best-performing configuration of Dridan (2013) in our experiments, a variant henceforth referred to as ERG$_e$. Unlike the other parsers in our study, PET internally operates over an ambiguous token lattice, and there is no easy interface to feed the parser pre-tokenized inputs. We approximate the effects of gold-standard tokenization by requesting from the parser a 2000-best list, which we filter for the top-ranked analysis whose leaves match the treebank tokenization. This approach is imperfect, as in some cases no token-compatible analysis may be on the n-best list, especially so in the ERG$_e$ setup (where lexical items may have been pruned by the sequence-labeling model). When this happens, we fall back to the top-ranked analysis and adjust our evaluation metrics to robustly deal with tokenization mismatches (see below).

B&N: Direct Dependency Parsing The parser of Bohnet and Nivre (2012), henceforth B&N, is a transition-based *dependency parser* with joint tagger that implements global learning and a beam search for non-projective labeled dependency parsing. This parser consistently outperforms pipeline systems (such as the Malt and MST parsers) both in terms of tagging and parsing accuracy for typologically diverse languages such as Chinese, English, and German. We apply B&N mostly 'out-of-the-box', training on the DT conversion of DeepBank Sections 00–19, and running the parser with an increased beam size of 80.

Berkeley: PCFG Parsing The Berkeley parser (Petrov et al., 2006; henceforth just Berkeley) is a gen-

Labels	Unary Rules Preserved						Unary Rules Removed			
	Long		Short		Mixed		Long		Short	
Cycles	**5**	**6**	**5**	**6**	**5**	**6**	**5**	**6**	**5**	**6**
Gaps	2	5	*0*	*0*	11	19	3	3	*0*	*0*
TA	90.96	90.62	91.11	*91.62*	90.93	90.94	88.46	87.65	89.16	88.46
F_1	76.39	75.66	79.81	*80.33*	76.70	76.74	74.53	73.72	75.15	73.56
LAS	86.26	85.90	82.50	83.15	*86.72*	86.16	83.96	83.20	80.49	79.56
UAS	89.34	88.92	89.80	*90.34*	89.42	88.84	87.12	86.54	87.95	87.15

Table 2: Tagging accuracy, PARSEVAL F_1, and dependency accuracy for Berkeley on WSJ development data.

erative, unlexicalized *phrase structure* parser that automatically derives a smoothed latent-variable PCFG from the treebank and refines the grammar by a split–merge procedure. The parser achieves state-of-the-art performance on various standard benchmarks. In § 4 below, we explain how we adapt ERG derivations for training and testing with Berkeley; for comparison to the other parsers in terms of DT dependency accuracy, we apply the converter of Ivanova et al. (2012) to Berkeley outputs. For technical reasons, however, the optional mapping from ERG to PTB tokenization is not applicable in this setup, and hence our experiments involving Berkeley are limited to ERG tokens and fine-grained lexical categories.

Evaluation Standard evaluation metrics in dependency parsing are labeled and unlabeled attachment scores (LAS, UAS; implemented by the CoNLL eval.pl scorer). These measure the percentage of tokens which are correctly attached to their head token and, for LAS, have the right dependency label. As assignment of lexical categories is a core part of syntactic analysis, we complement LAS and UAS with tagging accuracy scores (TA), where appropriate. However, in our work there are two complications to consider when using eval.pl. First, some of our parsers occasionally fail to return any analysis, notably Berkeley and ERG$_e$. For these inputs, our evaluation re-inserts the missing tokens in the parser output, padding with dummy 'placeholder' heads and dependency labels.

Second, a more difficult issue is caused by occassional tokenization mismatches in ERG parses, as discussed above. Since eval.pl identifies tokens by their position in the sentence, any difference of tokenization will lead to invalid results. One option would be to treat all system outputs with token mismatches as parse failures, but this over-penalizes, as potentially correct dependencies among corresponding tokens are also removed from the parser output. For this reason, we modify the evaluation of dependency accuracy to use sub-string character ranges, instead of consecutive identifiers, to encode token identities. This way, tokenization mismatches local to some sub-segment of the input will not 'throw off' token correspondences in other parts of the string.[5] We will refer to this character-based variant of the standard CoNLL metrics as LAS$_c$ and UAS$_c$.

4 PCFG Parsing of HPSG Derivations

Formally, the HPSG analyses in the DeepBank and Redwoods treebanks transcend the class of context-free grammars, of course. Nevertheless, one can pragmatically look at an ERG derivation as if it were a context-free phrase structure tree. On this view, standard, off-the-shelf PCFG parsing techniques are applicable to the ERG treebanks. Zhang and Krieger (2011) explore this space experimentally, combining the ERG, Redwoods (but not DeepBank), and massive collections of automatically parsed text. Their study, however, does not consider parser efficiency.[6].

In contrast, our goal is to reflect on practical tradeoffs along multiple dimensions. We therefore focus on Berkeley, as one of the currently best-performing (and relatively efficient) PCFG engines. Due to its ability to internally rewrite node labels, this parser should be expected to adapt well also to ERG derivations. Compared to the phrase structure annotations in the PTB, there are two structural differences evident in Figure 1. First, the inventories of phrasal and lexical labels are larger, at around 250 and 1000, respectively, compared to only about two dozen phrasal categories and 45 parts of speech in the PTB. Second, ERG derivations contain more unary (non-branching)

[5] Where tokenization is identical for the gold and system outputs, the score given by this generalized metric is exactly the same as that of eval.pl. Unless indicated otherwise, punctuation marks are included in scoring.

[6] Their best PCFG results are only a few points F_1 below the full HPSG parser, using massive PCFGs and exact inference; parsing times in fact exceed those of the native HPSG parser

	Gaps	Time	TA_c	LAS_c	UAS_c
Berkeley	1+0	*1.0*	92.9	86.65	89.86
B&N	0+0	1.7	92.9	86.76	89.65
ERG_a	0+0	10	*97.8*	*92.87*	*93.95*
ERG_e	13+44	1.8	96.4	91.60	92.72

Table 3: Parse failures and token mismatches ('gaps'), efficiency, and tagging and dependency accuracy on WSJ.

rules, recording for example morphological variation or syntacto-semantic category changes.[7]

Table 2 summarizes a first series of experiments, seeking to tune the Berkeley parser for maximum accuracy on our development set, DeepBank Section 20. We experimented with preserving unary rules in ERG derivations or removing them (as they make no difference to the final DT analysis); we further ran experiments using the native ('long') ERG construction identifiers, their generalizations to 'short' labels as used in DT, and a variant with long labels for unary and short ones for branching rules ('mixed'). We report results for training with five or six split–merge cycles, where fewer iterations generally showed inferior accuracy, and larger values led to more parse failures ('gaps' in Table 2). There are some noticeable trade-offs across tagging accuracy, dependency accuracy, and coverage, without a single best performer along all three dimensions. As our primary interest across parsers is dependency accuracy, we select the configuration with unary rules and long labels, trained with five split–merge cycles, which seems to afford near-premium LAS at near-perfect coverage.[8]

5 In-Domain Results

Our first cross-paradigm comparison of the three parsers is against the WSJ in-domain test data, as summarized in Table 3. There are substantive differences between parsers both in terms of coverage, speed, and accuracy. Berkeley fails to return an analysis for one input, whereas ERG_e cannot parse 13 sentences (close to one percent of the test set); just as the 44 inputs where parser output deviates in tokenization from the treebank, this is likely an effect of the lexical pruning applied in this setup. At an average of one second per input, Berkeley is the fastest of our parsers; ERG_a is exactly one order of magnitude slower. However, the lexical pruning of Dridan (2013) in ERG_e leads to a speed-up of almost a factor of six, making this variant of PET perform comparable to B&N. Maybe the strongest differences, however, we observe in tagging and dependency accuracies: The two data-driven parsers perform very similarly (at close to 93% TA and around 86.7% LAS); the two ERG parsers are comparable too, but at accuracy levels that are four to six points higher in both TA and LAS. Compared to ERG_a, the faster ERG_e variant performs very slightly worse—which likely reflects penalization for missing coverage and token mismatches—but it nevertheless delivers much higher accuracy than the data-driven parsers. In subsequent experiments, we will thus focus only on ERG_e.

6 Error Analysis

The ERG parsers outperform the two data-driven parsers on the WSJ data. Through in-depth error analysis, we seek to identify parser-specific properties that can explain the observed differences. In the following, we look at (a) the accuracy of individual dependency types, (b) dependency accuracy relative to (predicted and gold) dependency length, and (c) the distribution of LAS over different lexical categories.

Among the different dependency types, we observe that the notion of an adjunct is difficult for all three parsers. One of the hardest dependency labels is hdn-aj (post-adjunction to a nominal head), the relation employed for relative clauses and prepositional phrases attaching to a nominal head. The most common error for this relation is verbal attachment.

It has been noted that dependency parsers may exhibit systematic performance differences with respect to dependency length (i.e. the distance between a head and its argument; McDonald & Nivre, 2007). In our experiments, we find that the parsers perform comparably on longer dependency arcs (upwards of fifteen words), with ERG_a constantly showing the highest accuracy, and Berkeley holding a slight edge over B&N as dependency length increases.

In Figure 3, one can eyeball accuracy levels per lexical category, where conjunctions (c) and various types of prepositions (p and pp) are the most difficult for all three parsers. That the DT analysis of coordination is challenging is unsurprising. Schwartz et al.

[7] Examples of morphological rules in Figure 1 include v_pas_odlr and v_n3s-bse_ilr, for past-participle and non-third person singular or base inflection, respectively. Also, there are two instances of bare noun phrase formation: hdn_bnp-pn_c and hdn_bnp-qnt_c.

[8] A welcome side-effect of this choice is that we end up using native ERG derivations without modifications.

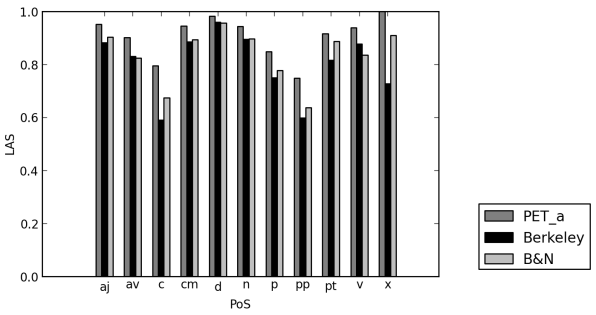


Figure 3: WSJ per-category dependency accuracies on coarse lexical head categories: adjective, adverb, conjunction, complementizer, determiner, noun, preposition, lexical prepositional phrase, punctuation, verb, and others.

(2012) show that choosing conjunctions as heads in coordinate structures is harder to parse for direct dependency parsers (while this analysis also is linguistically more expressive). Our results confirm this effect also for the PCFG and (though to a lesser degree) for ERG$_a$. At the same time, conjunctions are among the lexical categories for which ERG$_a$ most clearly outperforms the other parsers. Berkeley and B&N exhibit LAS error rates of around 35–41% for conjunctions, whereas the ERG$_a$ error rate is below 20%. For many of the coordinate structures parsed correctly by ERG$_a$ but not the other two, we found that attachment to root constitutes the most frequent error type—indicating that clausal coordination is particularly difficult for the data-driven parsers.

The attachment of prepositions constitutes a notorious difficulty in syntactic analysis. Unlike 'standard' PoS tag sets, ERG lexical types provide a more fine-grained analysis of prepositions, for example recognizing a lexicalized PP like *in full*, or making explicit the distinction between semantically contenful vs. vacuous prepositions. In our error analysis, we find that parser performance across the various prepositional sub-types varies a lot. For some prepositions, all parsers perform comparatively well; e.g. p_np_ptcl-of_le, for semantically vacuous *of*, ranks among the twenty most accurate lexical categories across the board. Other types of prepositions are among the categories exhibiting the highest error rates, e.g. p_np_i_le for 'common' prepositions, taking an NP argument and projecting intersective modifier semantics. Even so, Figure 3 shows that the attachment of prepositions (p and pp) is an area where ERG$_a$ excels most markedly. Three frequent prepositional lexical types that show the largest ERG$_a$ advantages are p_np_ptcl-of_le (*history of Linux*), p_np_ptcl_le (*look for peace*), and p_np_i_le (*talk about friends*). Looking more closely at inputs where the parsers disagree, they largely involve (usages of) prepositions which are lexically selected for by their head. In other words, most prepositions in isolation are ambiguous lexical items. However, it appears that lexical information about the argument structure of heads encoded in the grammar allows ERG$_a$ to analyse these prepositions (in context) much more accurately.

		Gaps	TA$_c$	LAS$_c$	UAS$_c$
CB	Berkeley	1+0	87.1	78.13	83.14
	B&N	0+0	87.7	77.70	82.96
	ERG$_e$	8+8	95.3	90.02	91.58
SC	Berkeley	1+0	87.2	79.81	85.10
	B&N	0+0	85.9	78.08	83.21
	ERG$_e$	11+7	94.9	89.94	91.26
VM	Berkeley	7+0	84.0	74.40	83.38
	B&N	0+0	83.1	75.28	82.86
	ERG$_e$	11+42	94.4	90.18	91.75
WS	Berkeley	7+0	87.7	80.31	85.09
	B&N	0+0	88.4	80.63	85.24
	ERG$_e$	4+12	96.9	90.64	91.76

Table 4: Cross-domain coverage (parse failures and token mismatches) and tagging and dependency accuracies.

7 Cross-Domain Results

To gauge the resilience of the different systems to domain and genre variation, we applied the same set of parsers—without re-training or other adaptation—to the additional Redwoods test data. Table 4 summarizes coverage and accuracy results across the four diverse samples. Again, Berkeley and B&N pattern alike, with Berkeley maybe slightly ahead in terms of dependency accuracy, but penalized on two of the test sets for parse failures. LAS for the two data-driven parsers ranges between 74% and 81%, up to 12 points below their WSJ performance. Though large, accuracy drops on a similar scale have been observed repeatedly for purely statistical systems when moving out of the WSJ domain without adaptation (Gildea, 2001; Nivre et al., 2007). In contrast, ERG$_e$ performance is more similar to WSJ results, with a maximum LAS drop of less than two points.[9] For

[9] It must be noted that, unlike the WSJ test data, some of these cross-domain data sets have been used in ERG development throughout the years, notably VM and CB, and thus the grammar is likely to have particularly good linguistic coverage of this data.

		Gaps	Lexical Types		PTB PoS Tags	
			LAS_c	UAS_c	LAS_c	UAS_c
WSJ	B&N	0+0	88.78	91.52	91.56	93.63
	ERG_e	13+9	92.38	93.53	92.38	93.53
CB	B&N	0+0	81.56	86.18	84.54	88.53
	ERG_e	8+4	90.77	92.21	90.77	92.21
SC	B&N	0+0	81.69	86.11	85.17	88.85
	ERG_e	11+0	90.13	91.86	90.13	91.86
VM	B&N	0+0	77.00	83.73	82.76	88.11
	ERG_e	10+0	91.55	93.08	91.55	93.08
WS	B&N	0+0	82.09	86.17	84.59	88.41
	ERG_e	4+0	91.61	92.62	91.61	92.62

Table 5: Coverage and dependency accuracies with PTB tokenization and either detailed or coarse lexical categories.

Wikipedia text (WS; previously unseen data for the ERG, just as for the other two), for example, both tagging and dependency accuracies are around ten points higher, an error reduction of more than 50%. From these results, it is evident that the general linguistic knowledge available in ERG parsing makes it far more resilient to variation in domain and text type.

8 Sanity: PTB Tokenization and PoS Tags

Up to this point, we have applied the two data-driven parsers in a setup that one might consider somewhat 'off-road'; although our experiments are on English, they involve unusual tokenization and lexical categories. For example, the ERG treatment of punctuation as 'pseudo-affixes' increases vocabulary size, which PET may be better equipped to handle due to its integrated treatment of morphological variation. In two concluding experiments, we seek to isolate the effects of tokenization conventions and granularity of lexical categories, taking advantage of optional output flexibility in the DT converter of Ivanova et al. (2012).[10] Table 5 confirms that tokenization does make a difference. In combination with fine-grained lexical categories still, B&N obtains LAS gains of two to three points, compared to smaller gains (around or below one point) for ERG_e.[11] However, in this setup our two earlier observations still hold true: ERG_e is substantially more accurate within the WSJ domain and far more resilient to domain and genre variation. When we simplify the syntactic analysis task and train and test B&N on coarse-grained PTB PoS tags only, in-domain differences between the two parsers are further reduced (to 0.8 points), but ERG_e still delivers an error reduction of ten percent compared to B&N. The picture in the cross-domain comparison is not qualitatively different, also in this simpler parsing task, with ERG_e maintaining accuracy levels comparable to WSJ, while B&N accuracies degrade markedly.

9 Discussion and Conclusion

Our experiments sought to contrast state-of-the-art representatives from three parsing paradigms on the task of producing bi-lexical syntactic dependencies for English. For the HPSG-derived DT scheme, we find that hybrid, grammar-driven parsing yields superior accuracy, both in- and in particular cross-domain, at processing times comparable to the currently best direct dependency parser. These results corroborate the Dutch findings of Plank and van Noord (2010) for English, where more training data is available and in comparison to more advanced data-driven parsers. In most of this work, we have focussed exclusively on parser inputs represented in the DeepBank and Redwoods treebanks, ignoring 15 percent of the original running text, for which the ERG and PET do not make available a gold-standard analysis. While a parser with partial coverage can be useful in some contexts, obviously the data-driven parsers must be credited for providing a syntactic analysis of (almost) all inputs. However, the ERG coverage gap can be straighforwardly addressed by falling back to another parser when necessary. Such a system combination would undoubtedly yield better tagging and dependency accuracies than the data-driven parsers by themselves, especially so in an open-domain setup. A secondary finding from our experiments is that PCFG parsing with Berkeley and conversion to DT dependencies yields equivalent or mildly more accurate analyses, at much greater efficiency. In future work, it would be interesting to include in this comparison other PCFG parsers and linear-time, transition-based dependency parsers, but a tentative generalization over our findings to date is that linguistically richer representations enable more accurate parsing.

Conversely, SC has hardly had a role in grammar engineering so far, and WS is genuinely unseen (for the current ERG and Redwoods release), i.e. treebankers were first exposed to it once the grammar and parser were frozen.

[10] As mapping from ERG derivations into PTB-style tokens and PoS tags is applied when converting to bi-lexical dependencies, we cannot easily include Berkeley in these final experiments.

[11] When converting to PTB-style tokenization, punctuation marks are always attached low in the DT scheme, to the immediately preceding or following token, effectively adding a large group of 'easy' dependencies.

Acknowledgments

We are grateful to our colleagues Emily M. Bender, Francis Bond, Rui Wang, and Yi Zhang for many helpful discussions and suggestions, as well as to our three anonymous reviewers for insightful comments. This work is in part funded by the Norwegian Research Council through its WeSearch project. Large-scale experimentation is made possible through access to the ABEL high-performance computing facilities at the University of Oslo, and we are grateful to the Scientific Computing staff at UiO, as well as to the Norwegian Metacenter for Computational Science, and the Norwegian tax payer.

References

Adolphs, P., Oepen, S., Callmeier, U., Crysmann, B., Flickinger, D., & Kiefer, B. (2008). Some fine points of hybrid natural language parsing. In *Proceedings of the 6th International Conference on Language Resources and Evaluation.* Marrakech, Morocco.

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., ... Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the workshop on speech and natural language* (p. 306 – 311). Pacific Grove, USA.

Bohnet, B., & Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning* (p. 1455 – 1465). Jeju Island, Korea.

Bos, J., et al. (Eds.). (2008). *Workshop on cross-framework and cross-domain parser evaluation.* Manchester, UK.

Briscoe, T., & Carroll, J. (2006). Evaluating the accuracy of an unlexicalised statistical parser on the PARC DepBank. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics* (p. 41 – 48). Sydney, Australia.

Callmeier, U. (2002). Preprocessing and encoding techniques in PET. In S. Oepen, D. Flickinger, J. Tsujii, & H. Uszkoreit (Eds.), *Collaborative language engineering. A case study in efficient grammar-based processing* (p. 127 – 140). Stanford, CA: CSLI Publications.

Carter, D. (1997). The TreeBanker. A tool for supervised training of parsed corpora. In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering* (p. 9 – 15). Madrid, Spain.

Cer, D., de Marneffe, M.-C., Jurafsky, D., & Manning, C. (2010). Parsing to Stanford Dependencies. Trade-offs between speed and accuracy. In *Proceedings of the 7th International Conference on Language Resources and Evaluation* (p. 1628 – 1632). Valletta, Malta.

Clark, S., & Curran, J. R. (2007). Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (p. 248 – 255). Prague, Czech Republic.

de Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation* (p. 1 – 8). Manchester, UK.

Dridan, R. (2013). Ubertagging. Joint segmentation and supertagging for English. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (p. 1 – 10). Seattle, WA, USA.

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6 (1)*, 15 – 28.

Flickinger, D., Zhang, Y., & Kordoni, V. (2012). DeepBank. A dynamically annotated treebank of the Wall Street Journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories* (p. 85 – 96). Lisbon, Portugal: Edições Colibri.

Foster, J., Cetinoglu, O., Wagner, J., Le Roux, J., Nivre, J., Hogan, D., & van Genabith, J. (2011). From news to comment. Resources and benchmarks for parsing the language of Web 2.0. In *Proceedings of the 2011 International Joint Conference on Natural Language Processing* (p. 893 – 901).

Fowler, T. A. D., & Penn, G. (2010). Accurate context-free parsing with Combinatory Categorial Grammar. In *Proceedings of the 48th Meeting of the Association for Computational Linguistics* (p. 335 – 344). Uppsala, Sweden.

Francis, W. N., & Kucera, H. (1982). *Frequency analysis of english usage*. New York: Houghton Mifflin Co.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (p. 167 – 202). Pittsburgh, USA.

Hockenmaier, J., & Steedman, M. (2007). CCGbank. A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, *33*, 355 – 396.

Ivanova, A., Oepen, S., & Øvrelid, L. (2013). Survey on parsing three dependency representations for English. In *Proceedings of the 51th Meeting of the Association for Computational Linguistics* (p. 31 – 37). Sofia, Bulgaria.

Ivanova, A., Oepen, S., Øvrelid, L., & Flickinger, D. (2012). Who did what to whom? A contrastive study of syntacto-semantic dependencies. In *Proceedings of the sixth linguistic annotation workshop* (p. 2 – 11). Jeju, Republic of Korea.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M., & Kaplan, R. M. (2003). The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora* (p. 1 – 8). Budapest, Hungary.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpora of English: The Penn Treebank. *Computational Linguistics*, *19*, 313 – 330.

McDonald, R. T., & Nivre, J. (2007). Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning* (p. 122 – 131). Prague, Czech Republic.

Miyao, Y., Sagae, K., & Tsujii, J. (2007). Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the 2007 Workshop on Grammar Engineering across Frameworks* (p. 238 – 258). Palo Alto, California.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. (2007). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning* (p. 915 – 932). Prague, Czech Republic.

Oepen, S., & Carroll, J. (2000). Ambiguity packing in constraint-based parsing. Practical results. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics* (p. 162 – 169). Seattle, WA, USA.

Oepen, S., Flickinger, D., Toutanova, K., & Manning, C. D. (2004). LinGO Redwoods. A rich and dynamic treebank for HPSG. *Research on Language and Computation*, *2*(4), 575 – 596.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics* (p. 433 – 440). Sydney, Australia.

Plank, B., & van Noord, G. (2010). Grammar-driven versus data-driven. Which parsing system is more affected by domain shifts? In *Proceedings of the 2010 Workshop on NLP and Linguistics: Finding the common ground* (p. 25 – 33). Uppsala, Sweden: Association for Computational Linguistics.

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Chicago, USA: The University of Chicago Press.

Schwartz, R., Abend, O., & Rappoport, A. (2012). Learnability-based syntactic annotation design. In *Proceedings of the 24th International Conference on Computational Linguistics.* Mumbai, India.

Wahlster, W. (Ed.). (2000). *Verbmobil. Foundations of speech-to-speech translation* (Artificial Intelligence ed.). Berlin, Germany: Springer.

Ytrestøl, G., Oepen, S., & Flickinger, D. (2009). Extracting and annotating Wikipedia sub-domains. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories* (p. 185 – 197). Groningen, The Netherlands.

Zhang, Y., & Krieger, H.-U. (2011). Large-scale corpus-driven PCFG approximation of an HPSG. In *Proceedings of the 12th International Conference on Parsing Technologies* (p. 198 – 208). Dublin, Ireland.

Zhang, Y., Oepen, S., & Carroll, J. (2007). Efficiency in unification-based n-best parsing. In *Proceedings of the 10th International Conference on Parsing Technologies* (p. 48 – 59). Prague, Czech Republic.

Zhang, Y., & Wang, R. (2009). Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (p. 378 – 386). Suntec, Singapore.