

# Cross-Domain Dependency Parsing Using a Deep Linguistic Grammar

Yi Zhang

Dept of Computational Linguistics  
Saarland University  
LT-Lab, DFKI GmbH  
D-66123 Saarbrücken, Germany  
yzhang@coli.uni-sb.de

Rui Wang

Dept of Computational Linguistics  
Saarland University  
66123 Saarbrücken, Germany  
rwang@coli.uni-sb.de

## Abstract

Pure statistical parsing systems achieves high in-domain accuracy but performs poorly out-domain. In this paper, we propose two different approaches to produce syntactic dependency structures using a large-scale hand-crafted HPSG grammar. The dependency backbone of an HPSG analysis is used to provide general linguistic insights which, when combined with state-of-the-art statistical dependency parsing models, achieves performance improvements on out-domain tests.<sup>†</sup>

## 1 Introduction

Syntactic dependency parsing is attracting more and more research focus in recent years, partially due to its theory-neutral representation, but also thanks to its wide deployment in various NLP tasks (machine translation, textual entailment recognition, question answering, information extraction, etc.). In combination with machine learning methods, several statistical dependency parsing models have reached comparable high parsing accuracy (McDonald et al., 2005b; Nivre et al., 2007b). In the meantime, successful continuation of CoNLL Shared Tasks since 2006 (Buchholz and Marsi, 2006; Nivre et al., 2007a; Surdeanu et al., 2008) have witnessed how easy it has become to train a statistical syntactic dependency parser provided that there is annotated treebank.

While the dissemination continues towards various languages, several issues arise with such purely data-driven approaches. One common observation is that statistical parser performance drops significantly when tested on a dataset *different* from the training set. For instance, when using the Wall Street Journal (WSJ) sections of the Penn

Treebank (Marcus et al., 1993) as training set, tests on BROWN Sections typically result in a 6-8% drop in labeled attachment scores, although the average sentence length is much shorter in BROWN than that in WSJ. The common interpretation is that the test set is heterogeneous to the training set, hence in a different “domain” (in a loose sense). The typical cause of this is that the model overfits the training domain. The concerns over random choice of training corpus leading to linguistically inadequate parsing systems increase over time.

While the statistical revolution in the field of computational linguistics gaining high publicity, the conventional symbolic grammar-based parsing approaches have undergone a quiet period of development during the past decade, and reemerged very recently with several large scale grammar-driven parsing systems, benefiting from the combination of well-established linguistic theories and data-driven stochastic models. The obvious advantage of such systems over pure statistical parsers is their usage of hand-coded linguistic knowledge irrespective of the training data. A common problem with grammar-based parser is the lack of robustness. Also it is difficult to derive grammar compatible annotations to train the statistical components.

## 2 Parser Domain Adaptation

In recent years, two statistical dependency parsing systems, *MaltParser* (Nivre et al., 2007b) and *MSTParser* (McDonald et al., 2005b), representing different threads of research in data-driven machine learning approaches have obtained high publicity, for their state-of-the-art performances in open competitions such as CoNLL Shared Tasks. *MaltParser* follows the *transition-based* approach, where parsing is done through a series of actions deterministically predicted by an *oracle* model. *MSTParser*, on the other hand, follows the *graph-based* approach where the best parse tree is acquired by searching for a spanning tree

<sup>†</sup>The first author thanks the German Excellence Cluster of Multimodal Computing and Interaction for the support of the work. The second author is funded by the PIRE PhD scholarship program.

which maximizes the score on either a partially or a fully connected graph with all words in the sentence as nodes (Eisner, 1996; McDonald et al., 2005b).

As reported in various evaluation competitions, the two systems achieved comparable performances. More recently, approaches of combining these two parsers achieved even better dependency accuracy (Nivre and McDonald, 2008). Granted for the differences between their approaches, both systems heavily rely on machine learning methods to estimate the parsing model from an annotated corpus as training set. Due to the heavy cost of developing high quality large scale syntactically annotated corpora, even for a resource-rich language like English, only very few of them meets the criteria for training a general purpose statistical parsing model. For instance, the text style of WSJ is newswire, and most of the sentences are statements. Being lack of non-statements in the training data could cause problems, when the testing data contain many interrogative or imperative sentences as in the BROWN corpus. Therefore, the unbalanced distribution of linguistic phenomena in the training data leads to inadequate parser output structures. Also, the financial domain specific terminology seen in WSJ can skew the interpretation of daily life sentences seen in BROWN.

There has been a substantial amount of work on parser adaptation, especially from WSJ to BROWN. Gildea (2001) compared results from different combinations of the training and testing data to demonstrate that the size of the feature model can be reduced via excluding “domain-dependent” features, while the performance could still be preserved. Furthermore, he also pointed out that if the additional training data is heterogeneous from the original one, the parser will not obtain a substantially better performance. Bacchiani et al. (2006) generalized the previous approaches using a maximum a posteriori (MAP) framework and proposed both supervised and unsupervised adaptation of statistical parsers. McClosky et al. (2006) and McClosky et al. (2008) have shown that out-domain parser performance can be improved with self-training on a large amount of unlabeled data. Most of these approaches focused on the machine learning perspective instead of the linguistic knowledge embraced in the parsers. Little study has been reported on approaches of incorporating linguistic features to make the parser less dependent on the

nature of training and testing dataset, without resorting to huge amount of unlabeled out-domain data. In addition, most of the previous work have been focusing on constituent-based parsing, while the domain adaptation of the dependency parsing has not been fully explored.

Taking a different approach towards parsing, grammar-based parsers appear to have much linguistic knowledge encoded within the grammars. In recent years, several of these linguistically motivated grammar-driven parsing systems achieved high accuracy which are comparable to the treebank-based statistical parsers. Notably are the constraint-based linguistic frameworks with mathematical rigor, and provide grammatical analyses for a large variety of phenomena. For instance, the Head-Driven Phrase Structure Grammar (Pollard and Sag, 1994) has been successfully applied in several parsing systems for more than a dozen of languages. Some of these grammars, such as the English Resource Grammar (ERG; Flickinger (2002)), have undergone over decades of continuous development, and provide precise linguistic analyses for a broad range of phenomena. These linguistic knowledge are encoded in highly generalized form according to linguists’ reflection for the target languages, and tend to be largely independent from any specific domain.

The main issue of parsing with precision grammars is that broad coverage and high precision on linguistic phenomena do not directly guarantee robustness of the parser with noisy real world texts. Also, the detailed linguistic analysis is not always of the highest interest to all NLP applications. It is not always straightforward to scale down the detailed analyses embraced by deep grammars to a shallower representation which is more accessible for specific NLP tasks. On the other hand, since the dependency representation is relatively theory-neutral, it is possible to convert from other frameworks into its backbone representation in dependencies. For HPSG, this is further assisted by the clear marking of head daughters in headed phrases. Although the statistical components of the grammar-driven parser might be still biased by the training domain, the hand-coded grammar rules guarantee the basic linguistic constraints to be met. This not to say that domain adaptation is not an issue for grammar-based parsing systems, but the built-in linguistic knowledge can be ex-

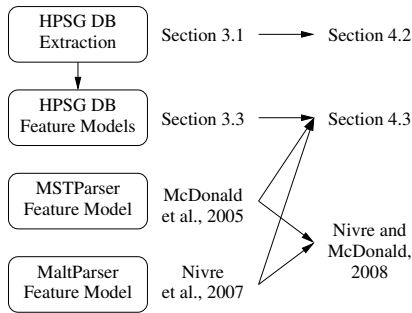


Figure 1: Different dependency parsing models and their combinations. DB stands for dependency backbone.

explored to reduce the performance drop in pure statistical approaches.

### 3 Dependency Parsing with HPSG

In this section, we explore two possible applications of the HPSG parsing onto the syntactic dependency parsing task. One is to extract dependency backbone from the HPSG analyses of the sentences and directly convert them into the target representation; the other way is to encode the HPSG outputs as additional features into the existing statistical dependency parsing models. In the previous work, Nivre and McDonald (2008) have integrated *MSTParser* and *MaltParser* by feeding one parser’s output as features into the other. The relationships between our work and their work are roughly shown in Figure 1.

#### 3.1 Extracting Dependency Backbone from HPSG Derivation Tree

Given a sentence, each parse produced by the parser is represented by a typed feature structure, which recursively embeds smaller feature structures for lower level phrases or words. For the purpose of dependency backbone extraction, we only look at the derivation tree which corresponds to the constituent tree of an HPSG analysis, with all non-terminal nodes labeled by the names of the grammar rules applied. Figure 2 shows an example. Note that all grammar rules in ERG are either unary or binary, giving us relatively *deep* trees when compared with annotations such as Penn Treebank. Conceptually, this conversion is similar to the conversions from deeper structures to GR representations reported by Clark and Curran (2007) and Miyao et al. (2007).

The dependency backbone extraction works by first identifying the head daughter for each bi-

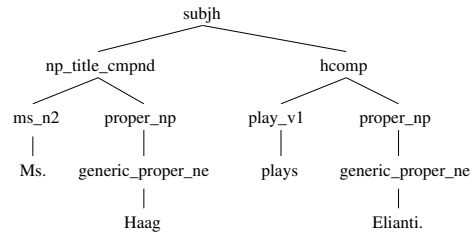


Figure 2: An example of an HPSG derivation tree with ERG

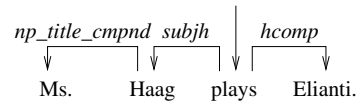


Figure 3: An HPSG dependency backbone structure

nary grammar rule, and then propagating the head word of the head daughter upwards to their parents, and finally creating a dependency relation, labeled with the HPSG rule name of the parent node, from the head word of the parent to the head word of the non-head daughter. See Figure 3 for an example of such an extracted backbone.

For the experiments in this paper, we used July-08 version of the ERG, which contains in total 185 grammar rules (morphological rules are not counted). Among them, 61 are unary rules, and 124 are binary. Many of the binary rules are clearly marked as headed phrases. The grammar also indicates whether the head is on the left (*head-initial*) or on the right (*head-final*). However, there are still quite a few binary rules which are not marked as headed-phrases (according to the linguistic theory), e.g. rules to handle coordinations, appositions, compound nouns, etc. For these rules, we refer to the conversion of the Penn Treebank into dependency structures used in the CoNLL 2008 Shared Task, and mark the heads of these rules in a way that will arrive at a compatible dependency backbone. For instance, the left most daughters of coordination rules are marked as heads. In combination with the right-branching analysis of coordination in ERG, this leads to the same dependency attachment in the CoNLL syntax. Eventually, 37 binary rules are marked with a head daughter on the left, and 86 with a head daughter on the right.

Although the extracted dependency is similar to the CoNLL shared task dependency structures, minor systematic differences still exist for some phe-

nomena. For example, the possessive “s” is annotated to be governed by its preceding word in CoNLL dependency; while in HPSG, it is treated as the head of a “specifier-head” construction, hence governing the preceding word in the dependency backbone. With several simple tree rewriting rules, we are able to fix the most frequent inconsistencies. With the rule-based backbone extraction and repair, we can finally turn our HPSG parser outputs into dependency structures<sup>1</sup>. The unlabeled attachment agreement between the HPSG backbone and CoNLL dependency annotation will be shown in Section 4.2.

### 3.2 Robust Parsing with HPSG

As mentioned in Section 2, one pitfall of using a precision-oriented grammar in parsing is its lack of robustness. Even with a large scale broad coverage grammar like ERG, using our settings we only achieved 75% of sentential coverage<sup>2</sup>. Given that the grammar has never been fine-tuned for the financial domain, such coverage is very encouraging. But still, the remaining unparsed sentences comprise a big coverage gap.

Different strategies can be taken here. One can either keep the high precision by only looking at full parses from the HPSG parser, of which the analyses are completely admitted by grammar constraints. Or one can trade precision for extra robustness by looking at the most probable incomplete analysis. Several partial parsing strategies have been proposed (Kasper et al., 1999; Zhang and Kordoni, 2008) as the robust fallbacks for the parser when no available analysis can be derived. In our experiment, we select the sequence of most likely fragment analyses according to their local disambiguation scores as the partial parse. When combined with the dependency backbone extraction, partial parses generate disjoint tree fragments. We simply attach all fragments onto the virtual root node.

<sup>1</sup>It is also possible map from HPSG rule names (together with the part-of-speech of head and dependent) to CoNLL dependency labels. This remains to be explored in the future.

<sup>2</sup>More recent study shows that with carefully designed retokenization and preprocessing rules, over 80% sentential coverage can be achieved on the WSJ sections of the Penn Treebank data using the same version of ERG. The numbers reported in this paper are based on a simpler preprocessor, using rather strict time/memory limits for the parser. Hence the coverage number reported here should not be taken as an absolute measure of grammar performance.

### 3.3 Using Feature-Based Models

Besides directly using the dependency backbone of the HPSG output, we could also use it for building feature-based models of statistical dependency parsers. Since we focus on the domain adaptation issue, we incorporate a less domain dependent language resource (i.e. the HPSG parsing outputs using ERG) into the features models of statistical parsers. As modern grammar-based parsers has achieved high runtime efficiency (with our HPSG parser parsing at an average speed of  $\sim 3$  sentences per second), this adds up to an acceptable overhead.

#### 3.3.1 Feature Model with MSTParser

As mentioned before, MSTParser is a graph-based statistical dependency parser, whose learning procedure can be viewed as the assignment of different weights to all kinds of dependency arcs. Therefore, the feature model focuses on each kind of head-child pair in the dependency tree, and mainly contains four categories of features (McDonald et al., 2005a): basic uni-gram features, basic bi-gram features, in-between POS features, and surrounding POS features. It is emphasized by the authors that the last two categories contribute a large improvement to the performance and bring the parser to the state-of-the-art accuracy.

Therefore, we extend this feature set by adding four more feature categories, which are similar to the original ones, but the dependency relation was replaced by the dependency backbone of the HPSG outputs. The extended feature set is shown in Table 1.

#### 3.3.2 Feature Model with MaltParser

MaltParser is another trend of dependency parser, which is based on transitions. The learning procedure is to train a statistical model, which can help the parser to decide which operation to take at each parsing status. The basic data structures are a stack, where the constructed dependency graph is stored, and an input queue, where the unprocessed data are put. Therefore, the feature model focuses on the tokens close to the top of the stack and also the head of the queue.

Provided with the original features used in MaltParser, we add extra ones about the top token in the stack and the head token of the queue derived from the HPSG dependency backbone. The extended feature set is shown in Table 2 (the new features are listed separately).

<b>Uni-gram Features:</b> $h-w, h-p; h-w; h-p; c-w, c-p; c-w; c-p$
<b>Bi-gram Features:</b> $h-w, h-p, c-w, c-p; h-p, c-w, c-p; h-w, c-w, c-p; h-w, h-p, c-p; h-w, h-p, c-w; h-w, c-w; h-p, c-p$
<b>POS Features of words in between:</b> $h-p, b-p, c-p$
<b>POS Features of words surround:</b> $h-p, h-p+1, c-p-1, c-p; h-p-1, h-p, c-p-1, c-p; h-p, h-p+1, c-p, c-p+1; h-p-1, h-p, c-p, c-p+1$

Table 1: The Extra Feature Set for `MSTParser`.  $h$ : the HPSG head of the current token;  $c$ : the current token;  $b$ : each token in between;  $-1/+1$ : the previous/next token;  $w$ : word form;  $p$ : POS

<b>POS Features:</b> $s[0]-p; s[1]-p; i[0]-p; i[1]-p; i[2]-p; i[3]-p$
<b>Word Form Features:</b> $s[0]-h-w; s[0]-w; i[0]-w; i[1]-w$
<b>Dependency Features:</b> $s[0]-lmc-d; s[0]-d; s[0]-rmc-d; i[0]-lmc-d$
<b>New Features:</b> $s[0]-hh-w; s[0]-hh-p; s[0]-hr; i[0]-hh-w; i[0]-hh-p; i[0]-hr$

Table 2: The Extended Feature Set for `MaltParser`.  $s[0]/s[1]$ : the first and second token on the top of the stack;  $i[0]/i[1]/i[2]/i[3]$ : front tokens in the input queue;  $h$ : head of the token;  $hh$ : HPSG DB head of the token;  $w$ : word form;  $p$ : POS;  $d$ : dependency relation;  $hr$ : HPSG rule;  $lmc/rmc$ : left-/right-most child

With the extra features, we hope that the training of the statistical model will not overfit the in-domain data, but be able to deal with domain independent linguistic phenomena as well.

## 4 Experiment Results & Error Analyses

To evaluate the performance of our different dependency parsing models, we tested our approaches on several dependency treebanks for English in a similar spirit to the `CoNLL 2006-2008 Shared Tasks`. In this section, we will first describe the datasets, then present the results. An error analysis is also carried out to show both pros and cons of different models.

### 4.1 Datasets

In previous years of `CoNLL Shared Tasks`, several datasets have been created for the purpose of dependency parser evaluation. Most of them are converted automatically from existing treebanks in various forms. Our experiments adhere to the `CoNLL 2008 dependency syntax` (Yamada et al. 2003, Johansson et al. 2007) which was used to convert Penn-Treebank constituent trees into single-head, single-root, traceless and non-projective dependencies.

**WSJ** This dataset comprises of three portions. The larger part is converted from the Penn Treebank Wall Street Journal Sections #2–#21, and is used for training statistical dependency parsing models; the smaller part, which covers sentences from Section #23, is used for testing.

**Brown** This dataset contains a subset of converted sentences from `BROWN` sections of the Penn Treebank. It is used for the out-domain test.

**PCchemtb** This dataset was extracted from the `PennBioIE CYP` corpus, containing 195 sentences from biomedical domain. The same dataset has been used for the domain adaptation track of the `CoNLL 2007 Shared Task`. Although the original annotation scheme is similar to the Penn Treebank, the dependency extraction setting is slightly different to the `CoNLLWSJ` dependencies (e.g. the coordinations).

**Childes** This is another out-domain test set from the children language component of the `TalkBank`, containing dialogs between parents and children. This is the other datasets used in the domain adaptation track of the `CoNLL 2007 Shared Task`. The dataset is annotated with unlabeled dependencies. As have been reported by others, several systematic differences in the original `CHILDES` annotation scheme has led to the poor system performances on this track of the Shared Task in 2007. Two main differences concern a) root attachments, and b) coordinations. With several simple heuristics, we change the annotation scheme of the original dataset to match the Penn Treebank-based datasets. The new dataset is referred to as `CHILDES*`.

### 4.2 HPSG Backbone as Dependency Parser

First we test the agreement between HPSG dependency backbone and `CoNLL` dependency. While approximating a target dependency structure with rule-based conversion is not the main focus of this work, the agreement between two representations gives indication on how similar and consistent the two representations are, and a rough impression of whether the feature-based models can benefit from the HPSG backbone.

	# sentence	$\phi$ w/s	DB(F)%	DB(P)%
WSJ	2399	24.04	50.68	63.85
BROWN	425	16.96	66.36	76.25
PCHEMTB	195	25.65	50.27	61.60
CHILDES*	666	7.51	67.37	70.66
WSJ-P	1796 (75%)	22.25	71.33	–
BROWN-P	375 (88%)	15.74	80.04	–
PCHEMTB-P	147 (75%)	23.99	69.27	–
CHILDES*-P	595 (89%)	7.49	73.91	–

Table 3: Agreement between HPSG dependency backbone and CoNLL 2008 dependency in unlabeled attachment score. DB(F): full parsing mode; DB(P): partial parsing mode; Punctuations are excluded from the evaluation.

The PET parser, an efficient parser HPSG parser is used in combination with ERG to parse the test sets. Note that the training set is not used. The grammar is not adapted for any of these specific domain. To pick the most probable reading from HPSG parsing outputs, we used a discriminative parse selection model as described in (Toutanova et al., 2002) trained on the LOGON Treebank (Oepen et al., 2004), which is significantly different from any of the test domain. The treebank contains about 9K sentences for which HPSG analyses are manually disambiguated. The difference in annotation make it difficult to simply merge this HPSG treebank into the training set of the dependency parser. Also, as Gildea (2001) suggests, adding such heterogeneous data to the training set will not automatically lead to performance improvement. It should be noted that domain adaptation also presents a challenge to the disambiguation model of the HPSG parser. All datasets we use in our should be considered out-domain to the HPSG disambiguation model.

Table 3 shows the agreement between the HPSG backbone and CoNLL dependency in unlabeled attachment score (UAS). The parser is set in either full parsing or partial parsing mode. Partial parsing is used as a fallback when full parse is not available. UAS are reported on all complete test sets, as well as fully parsed subsets (suffixed with “-p”).

It is not surprising to see that, without a decent fallback strategy, the full parse HPSG backbone suffers from insufficient coverage. Since the grammar coverage is statistically correlated to the average sentence length, the worst performance is observed for the PCHEMTB. Although sentences in CHILDES\* are significantly shorter than those

in BROWN, there is a fairly large amount of less well-formed sentences (either as a nature of child language, or due to the transcription from spoken dialogs). This leads to the close performance between these two datasets. PCHEMTB appears to be the most difficult one for the HPSG parser. The partial parsing fallback sets up a good safe net for sentences that fail to parse. Without resorting to any external resource, the performance was significantly improved on all complete test sets.

When we set the coverage of the HPSG grammar aside and only compare performance on the subsets of these datasets which are fully parsed by the HPSG grammar, the unlabeled attachment score jumps up significantly. Most notable is that the dependency backbone achieved over 80% UAS on BROWN, which is close to the performance of state-of-the-art statistical dependency parsing systems trained on WSJ (see Table 5 and Table 4). The performance difference across data sets correlates to varying levels of difficulties in linguists’ view. Our error analysis does confirm that frequent errors occur in WSJ test with financial terminology missing from the grammar lexicon. The relative performance difference between the WSJ and BROWN test is contrary to the results observed for statistical parsers trained on WSJ.

To further investigate the effect of HPSG parse disambiguation model on the dependency backbone accuracy, we used a set of 222 sentences from section of WSJ which have been parsed with ERG and manually disambiguated. Comparing to the WSJ-P result in Table 3, we improved the agreement with CoNLL dependency by another 8% (an upper-bound in case of a perfect disambiguation model).

### 4.3 Statistical Dependency Parsing with HPSG Features

Similar evaluations were carried out for the statistical parsers using extra HPSG dependency backbone as features. It should be noted that the performance comparison between MSTParser and MaltParser is not the aim of this experiment, and the difference might be introduced by the specific settings we use for each parser. Instead, performance variance using different feature models is the main subject. Also, performance drop on out-domain tests shows how domain dependent the feature models are.

For MaltParser, we use Arc-Eager algo-

rithm, and polynomial kernel with  $d = 2$ . For `MSTParser`, we use 1st order features and a projective decoder (Eisner, 1996).

When incorporating HPSG features, two settings are used. The `PARTIAL` model is derived by robust-parsing the entire training data set and extract features from every sentence to train a unified model. When testing, the `PARTIAL` model is used alone to determine the dependency structures of the input sentences. The `FULL` model, on the other hand is only trained on the full parsed subset of sentences, and only used to predict dependency structures for sentences that the grammar parses. For the unparsed sentences, the original models without HPSG features are used.

Parser performances are measured using both labeled and unlabeled attachment scores (LAS/UAS). For unlabeled CHILDES\* data, only UAS numbers are reported. Table 4 and 5 summarize results for `MSTParser` and `MaltParser`, respectively.

With both parsers, we see slight performance drops with both HPSG feature models on in-domain tests (WSJ), compared with the original models. However, on out-domain tests, full-parse HPSG feature models consistently outperform the original models for both parsers. The difference is even larger when only the HPSG fully parsed subsets of the test sets are concerned. When we look at the performance difference between in-domain and out-domain tests for each feature model, we observe that the drop is significantly smaller for the extended models with HPSG features.

We should note that we have not done any feature selection for our HPSG feature models. Nor have we used the best known configurations of the existing parsers (e.g. second order features in `MSTParser`). Admittedly the results on `PCHEMTB` are lower than the best reported results in `CoNLL 2007 Shared Task`, we shall note that we are not using any in-domain unlabeled data. Also, the poor performance of the HPSG parser on this dataset indicates that the parser performance drop is more related to domain-specific phenomena and not general linguistic knowledge. Nevertheless, the drops when compared to in-domain tests are constantly decreased with the help of HPSG analyses features. With the results on `BROWN`, the performance of our HPSG feature models will rank 2<sup>nd</sup> on the out-domain test for the `CoNLL 2008 Shared Task`.

Unlike the observations in Section 4.2, the partial parsing mode does not work well as a fallback in the feature models. In most cases, its performances are between the original models and the full-parse HPSG feature models. The partial parsing features obscure the linguistic certainty of grammatical structures produced in the full model. When used as features, such uncertainty leads to further confusion. Practically, falling back to the original models works better when HPSG full parse is not available.

#### 4.4 Error Analyses

Qualitative error analysis is also performed. Since our work focuses on the domain adaptation, we manually compare the outputs of the original statistical models, the dependency backbone, and the feature-based models on the out-domain data, i.e. the `BROWN` data set (both labeled and unlabeled results) and the `CHILDES*` data set (only unlabeled results).

For the dependency attachment (i.e. unlabeled dependency relation), fine-grained HPSG features do help the parser to deal with colloquial sentences, such as “What’s wrong with you?”. The original parser wrongly takes “what” as the root of the dependency tree and “s” is attached to “what”. The dependency backbone correctly finds out the root, and thus guide the extended model to make the right prediction. A correct structure of “..., were now neither active nor really relaxed.” is also predicted by our model, while the original model wrongly attaches “really” to “nor” and “relaxed” to “were”. The rich linguistic knowledge from the HPSG outputs also shows its usefulness. For example, in a sentence from the `CHILDES*` data, “Did you put dolly’s shoes on?”, the verb phrase “put on” can be captured by the HPSG backbone, while the original model attaches “on” to the adjacent token “shoes”.

For the dependency labels, the most difficulty comes from the prepositions. For example, “Scotty drove home alone in the Plymouth”, all the systems get the head of “in” correct, which is “drove”. However, none of the dependency labels is correct. The original model predicts the “`DIR`” relation, the extended feature-based model says “`TMP`”, but the gold standard annotation is “`LOC`”. This is because the HPSG dependency backbone knows that “in the Plymouth” is an adjunct of “drove”, but whether it is a temporal or

	Original		PARTIAL		FULL	
	LAS%	UAS%	LAS%	UAS%	LAS%	UAS%
WSJ	87.38	90.35	87.06	90.03	86.87	89.91
BROWN	80.46 (-6.92)	86.26 (-4.09)	80.55 (-6.51)	86.17 (-3.86)	<b>80.92 (-5.95)</b>	<b>86.58 (-3.33)</b>
PCHEMTB	53.37 (-33.8)	62.11 (-28.24)	54.69 (-32.37)	64.09 (-25.94)	56.45 (-30.42)	65.77 (-24.14)
CHILDES*	–	72.17 (-18.18)	–	74.91 (-15.12)	–	<b>75.64 (-14.27)</b>
WSJ-P	87.86	90.88	87.78	90.85	87.12	90.25
BROWN-P	81.58 (-6.28)	87.41 (-3.47)	81.92 (-5.86)	87.51 (-3.34)	<b>82.14 (-4.98)</b>	<b>87.80 (-2.45)</b>
PCHEMTB-P	56.32 (-31.54)	65.26 (-25.63)	59.36 (-28.42)	69.20 (-21.65)	60.69 (-26.43)	70.45 (-19.80)
CHILDES*-P	–	72.88 (-18.00)	–	76.02 (-14.83)	–	<b>76.76 (-13.49)</b>

Table 4: Performance of the MSTParser with different feature models. Numbers in parentheses are performance drops in out-domain tests, comparing to in-domain results. The upper part represents the results on the complete data sets, and the lower part is on the fully parsed subsets, indicated by “-P”.

	Original		PARTIAL		FULL	
	LAS%	UAS%	LAS%	UAS%	LAS%	UAS%
WSJ	86.47	88.97	85.39	88.10	85.66	88.40
BROWN	79.41 (-7.06)	84.75 (-4.22)	79.10 (-6.29)	84.58 (-3.52)	<b>79.56 (-6.10)</b>	<b>85.24 (-3.16)</b>
PCHEMTB	61.05 (-25.42)	71.32 (-17.65)	61.01 (-24.38)	70.99 (-17.11)	60.93 (-24.73)	70.89 (-17.51)
CHILDES*	–	74.97 (-14.00)	–	75.64 (-12.46)	–	<b>76.18 (-12.22)</b>
WSJ-P	86.99	89.58	86.09	88.83	85.82	88.76
BROWN-P	80.43 (-6.56)	85.78 (-3.80)	80.46 (-5.63)	85.94 (-2.89)	<b>80.62 (-5.20)</b>	<b>86.38 (-2.38)</b>
PCHEMTB-P	63.33 (-23.66)	73.54 (-16.04)	63.27 (-22.82)	73.31 (-15.52)	63.16 (-22.66)	73.06 (-15.70)
CHILDES*-P	–	75.95 (-13.63)	–	77.05 (-11.78)	–	<b>77.30 (-11.46)</b>

Table 5: Performance of the MaltParser with different feature models.

locative expression cannot be easily predicted at the pure syntactic level. This also suggests a joint learning of syntactic and semantic dependencies, as proposed in the CoNLL 2008 Shared Task.

Instances of wrong HPSG analyses have also been observed as one source of errors. For most of the cases, a correct reading exists, but not picked by our parse selection model. This happens more often with the WSJ test set, partially contributing to the low performance.

## 5 Conclusion & Future Work

Similar to our work, Sagae et al. (2007) also considered the combination of dependency parsing with an HPSG parser, although their work was to use statistical dependency parser outputs as soft constraints to improve the HPSG parsing. Nevertheless, a similar backbone extraction algorithm was used to map between different representations. Similar work also exists in the constituent-based approaches, where CFG backbones were used to improve the efficiency and robustness of HPSG parsers (Matsuzaki et al., 2007; Zhang and Kordoni, 2008).

In this paper, we restricted our investigation on the syntactic evaluation using labeled/unlabeled attachment scores. Recent discussions in the parsing community about meaningful cross-

framework evaluation metrics have suggested to use measures that are semantically informed. In this spirit, Zhang et al. (2008) showed that the semantic outputs of the same HPSG parser helps in the semantic role labeling task. Consistent with the results reported in this paper, more improvement was achieved on the out-domain tests in their work as well.

Although the experiments presented in this paper were carried out on a HPSG grammar for English, the method can be easily adapted to work with other grammar frameworks (e.g. LFG, CCG, TAG, etc.), as well as on languages other than English. We chose to use a hand-crafted grammar, so that the effect of training corpus on the deep parser is minimized (with the exception of the lexical coverage and disambiguation model).

As mentioned in Section 4.4, the performance of our HPSG parse selection model varies across different domains. This indicates that, although the deep grammar embraces domain independent linguistic knowledge, the lexical coverage and the disambiguation process among permissible readings is still domain dependent. With the mapping between HPSG analyses and their dependency backbones, one can potentially use existing dependency treebanks to help overcome the insufficient data problem for deep parse selection models.

## References

- Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. Map adaptation of stochastic grammars. *Computer speech and language*, 20(1):41–68.
- Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, New York City, USA.
- Stephen Clark and James Curran. 2007. Formalism-independent parser evaluation with ccg and depbank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic.
- Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhagen, Denmark.
- Dan Flickinger. 2002. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun’ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications.
- Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, Pittsburgh, USA.
- Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, C.J. Rupp, and Karsten Worm. 1999. Charting the depths of robust speech processing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pages 405–412, Maryland, USA.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.
- Takuya Matsuzaki, Yusuke Miyao, and Jun’ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1671–1676, Hyderabad, India.
- David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia.
- David McClosky, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 561–568, Manchester, UK.
- Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 91–98, Ann Arbor, Michigan.
- Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.
- Yusuke Miyao, Kenji Sagae, and Jun’ichi Tsujii. 2007. Towards framework-independent evaluation of deep linguistic parsers. In *Proceedings of the GEAR07 Workshop*, pages 238–258, Stanford, CA.
- Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June.
- 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 EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.
- Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.
- Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Veldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Som å kapp-ete med trollet? Towards MRS-Based Norwegian-English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, USA.
- Carl J. Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.
- Kenji Sagae, Yusuke Miyao, and Jun’ichi Tsujii. 2007. Hpsg parsing with shallow dependency constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 624–631, Prague, Czech Republic.
- Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning (CoNLL-2008)*, Manchester, UK.
- Kristina Toutanova, Christopher D. Manning, Stuart M. Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse ranking for a rich HPSG grammar. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories (TLT 2002)*, pages 253–263, Sozopol, Bulgaria.
- Yi Zhang and Valia Kordoni. 2008. Robust Parsing with a Large HPSG Grammar. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC’08)*, Marrakech, Morocco.
- Yi Zhang, Rui Wang, and Hans Uszkoreit. 2008. Hybrid Learning of Dependency Structures from Heterogeneous Linguistic Resources. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 198–202, Manchester, UK.