HPSG Parsing with Shallow Dependency Constraints

Kenji Sagae[1] and **Yusuke Miyao**[1] and **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{sagae,yusuke,tsujii}@is.s.u-tokyo.ac.jp

Abstract

We present a novel framework that combines strengths from surface syntactic parsing and deep syntactic parsing to increase deep parsing accuracy, specifically by combining dependency and HPSG parsing. We show that by using surface dependencies to constrain the application of wide-coverage HPSG rules, we can benefit from a number of parsing techniques designed for high-accuracy dependency parsing, while actually performing deep syntactic analysis. Our framework results in a 1.4% absolute improvement over a state-of-the-art approach for wide coverage HPSG parsing.

1 Introduction

Several efficient, accurate and robust approaches to data-driven dependency parsing have been proposed recently (Nivre and Scholz, 2004; McDonald et al., 2005; Buchholz and Marsi, 2006) for syntactic analysis of natural language using bilexical dependency relations (Eisner, 1996). Much of the appeal of these approaches is tied to the use of a simple formalism, which allows for the use of efficient parsing algorithms, as well as straightforward ways to train discriminative models to perform disambiguation. At the same time, there is growing interest in parsing with more sophisticated lexicalized grammar formalisms, such as Lexical Functional Grammar (LFG) (Bresnan, 1982), Lexicalized Tree Adjoining Grammar (LTAG) (Schabes et al., 1988), Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) and Combinatory Categorial Grammar (CCG) (Steedman, 2000), which represent deep syntactic structures that cannot be expressed in a shallower formalism designed to represent only aspects of surface syntax, such as the dependency formalism used in current mainstream dependency parsing.

We present a novel framework that combines strengths from surface syntactic parsing and deep syntactic parsing, specifically by combining dependency and HPSG parsing. We show that, by using surface dependencies to constrain the application of wide-coverage HPSG rules, we can benefit from a number of parsing techniques designed for high-accuracy dependency parsing, while actually performing deep syntactic analysis. From the point of view of HPSG parsing, accuracy can be improved significantly through the use of highly accurate discriminative dependency models, without the difficulties involved in adapting these models to a more complex and linguistically sophisticated formalism. In addition, improvements in dependency parsing accuracy are converted directly into improvements in HPSG parsing accuracy. From the point of view of dependency parsing, the application of HPSG rules to structures generated by a surface dependency model provides a principled and linguistically motivated way to identify deep syntactic phenomena, such as long-distance dependencies, raising and control.

We begin by describing our dependency and HPSG parsing approaches in section 2. In section 3, we present our framework for HPSG parsing with shallow dependency constraints, and in section 4 we

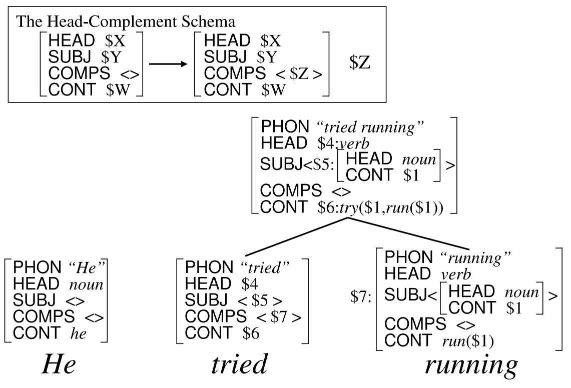
The Head-Complement Schema
HEAD $X SUBJ $Y COMPS <> CONT $W
HEAD $X SUBJ $Y COMPS < $Z > CONT $W
$Z
PHON "tried running" HEAD $4:verb SUBJ<$5: HEAD noun CONT $1 > COMPS <> CONT $6:try($1,run($1))
PHON "He" HEAD noun SUBJ <> COMPS <> CONT he
PHON "tried" HEAD $4 SUBJ < $5 > COMPS < $7 > CONT $6
$7: PHON "running" HEAD verb SUBJ< HEAD noun CONT $1 > COMPS <> CONT run($1)
He
tried
running

Figure 1: HPSG parsing

evaluate this framework empirically. Sections 5 and 6 discuss related work and conclusions.

2 Fast dependency parsing and wide-coverage HPSG parsing

2.1 Data-driven dependency parsing

Because we use dependency parsing as a step in deep parsing, it is important that we choose a parsing approach that is not only accurate, but also efficient. The deterministic shift/reduce classifier-based dependency parsing approach (Nivre and Scholz, 2004) has been shown to offer state-of-the-art accuracy (Nivre et al., 2006) with high efficiency due to a greedy search strategy. Our approach is based on Nivre and Scholz's approach, using support vector machines for classification of shift/reduce actions.

2.2 Wide-coverage HPSG parsing

HPSG (Pollard and Sag, 1994) is a syntactic theory based on lexicalized grammar formalism. In HPSG, a small number of *schemas* explain general construction rules, and a large number of *lexical entries* express word-specific syntactic/semantic constraints. Figure 1 shows an example of the process of HPSG parsing. First, lexical entries are assigned to each word in a sentence. In Figure 1, lexical entries express subcategorization frames and predicate argument structures. Parsing proceeds by applying schemas to lexical entries. In this example, the Head-Complement Schema is applied to the lexical entries of "*tried*" and "*running*". We then obtain a phrasal structure for "*tried running*". By repeatedly applying schemas to lexical/phrasal structures,

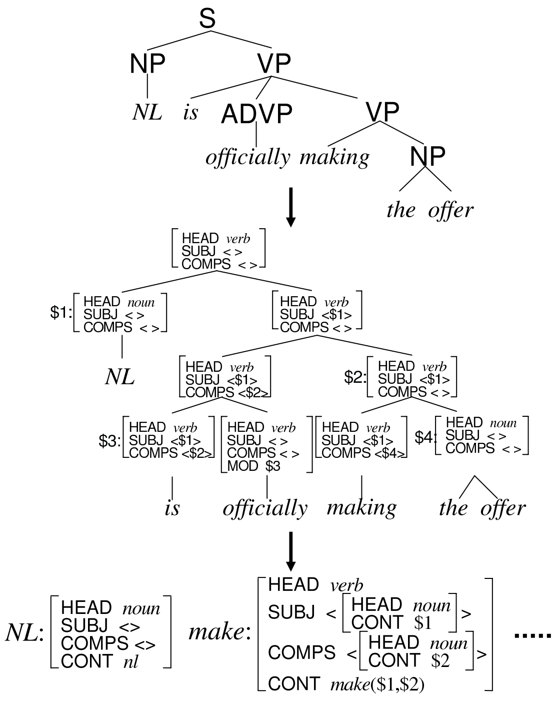
S
NP
VP
NL
is
ADVP
VP
officially
making
NP
the offer
HEAD verb SUBJ < > COMPS < >
$1: HEAD noun SUBJ < > COMPS < >
HEAD verb SUBJ <$1> COMPS < >
NL
HEAD verb SUBJ <$1> COMPS <$2>
$2: HEAD verb SUBJ <$1> COMPS < >
$3: HEAD verb SUBJ <$1> COMPS <$2>
HEAD verb SUBJ < > COMPS < > MOD $3
HEAD verb SUBJ <$1> COMPS <$4>
$4: HEAD noun SUBJ < > COMPS < >
is
officially
making
the offer
NL: HEAD noun SUBJ <> COMPS <> CONT nl
make: HEAD verb SUBJ < HEAD noun CONT $1 > COMPS < HEAD noun CONT $2 > CONT make($1,$2)
.....

Figure 2: Extracting HPSG lexical entries from the Penn Treebank

we finally obtain an HPSG parse tree that covers the entire sentence.

In this paper, we use an HPSG parser developed by Miyao and Tsujii (2005). This parser has a wide-coverage HPSG lexicon which is extracted from the Penn Treebank. Figure 2 illustrates their method for extraction of HPSG lexical entries. First, given a parse tree from the Penn Treebank (top), HPSG-style constraints are added and an HPSG-style parse tree is obtained (middle). Lexical entries are then extracted from the terminal nodes of the HPSG parse tree (bottom). This way, in addition to a wide-coverage lexicon, we also obtain an HPSG treebank, which can be used as training data for disambiguation models.

The disambiguation model of this parser is based on a maximum entropy model (Berger et al., 1996). The probability $p(T|W)$ of an HPSG parse tree T for the sentence $W = \langle w_1, \ldots, w_n \rangle$ is given as:

$$\begin{aligned} p(T|W) &= p(T|L,W)p(L|W) \\ &= \frac{1}{Z}\exp\left(\sum_i \lambda_i f_i(T)\right)\prod_j p(l_j|W), \end{aligned}$$

where $L = \langle l_1, \ldots, l_n \rangle$ are lexical entries and

$p(l_i|W)$ is the *supertagging probability*, i.e., the probability of assignining the lexical entry l_i to w_i (Ninomiya et al., 2006). The probability $p(T|L, W)$ is a maximum entropy model on HPSG parse trees, where Z is a normalization factor, and feature functions $f_i(T)$ represent syntactic characteristics, such as head words, lengths of phrases, and applied schemas. Given the HPSG treebank as training data, the model parameters λ_i are estimated so as to maximize the log-likelihood of the training data (Malouf, 2002).

3 HPSG parsing with dependency constraints

While a number of fairly straightforward models can be applied successfully to dependency parsing, designing and training HPSG parsing models has been regarded as a significantly more complex task. Although it seems intuitive that a more sophisticated linguistic formalism should be more difficult to parameterize properly, we argue that the difference in complexity between HPSG and dependency structures can be seen as incremental, and that the use of accurate and efficient techniques to determine the surface dependency structure of a sentence provides valuable information that aids HPSG disambiguation. This is largely because HPSG is based on a lexicalized grammar formalism, and as such its syntactic structures have an underlying dependency backbone. However, HPSG syntactic structures includes long-distance dependencies, and the underlying dependency structure described by and HPSG structure is a directed acyclic graph, not a dependency tree (as used by mainstream approaches to data-driven dependency parsing). This difference manifests itself in words that have multiple heads. For example, in the sentence *I tried to run*, the pronoun *I* is a dependent of *tried* and of *run*. This makes it possible to represent that *I* is the subject of both verbs, precisely the kind of information that cannot be represented in dependency parsing. If we ignore long-distance dependencies, however, HPSG structures can be seen as lexicalized trees that can be easily converted into dependency trees.

Given that for an HPSG representation of the syntactic structure of a sentence we can determine a dependency tree by removing long-distance dependencies, we can use dependency parsing techniques (such as the deterministic dependency parsing approach mentioned in section 2.1) to determine the underlying dependency trees in HPSG structures. This is the basis for the parsing framework presented here. In this approach, deep dependency analysis is done in two stages. First, a dependency parser determines the shallow dependency tree for the input sentence. This shallow dependency tree corresponds to the underlying dependency graph of the HPSG structure for the input sentence, without dependencies that roughly correspond to deep syntax. The second step is to perform HPSG parsing, as described in section 2.2, but using the shallow dependency tree to constrain the application of HPSG rules. We now discuss these two steps in more detail.

3.1 Determining shallow dependencies in HPSG structures using dependency parsing

In order to apply a data-driven dependency approach to the task of identifying the shallow dependency tree in HPSG structures, we first need a corpus of such dependency trees to serve as training data. We created a dependency training corpus based on the Penn Treebank (Marcus et al., 1993), or more specifically on the HPSG Treebank generated from the Penn Treebank (see section 2.2). For each HPSG structure in the HPSG Treebank, a dependency tree is extracted in two steps. First, the HPSG tree is converted into a CFG-style tree, simply by removing long-distance dependency links between nodes. A dependency tree is then extracted from the resulting lexicalized CFG-style tree, as is commonly done for converting constituent trees into dependency trees after the application of a head-percolation table (Collins, 1999).

Once a dependency training corpus is available, it is used to train a dependency parser as described in section 2.1. This is done by training a classifier to determine parser actions based on local features that represent the current state of the parser (Nivre and Scholz, 2004; Sagae and Lavie, 2005). Training data for the classifier is obtained by applying the parsing algorithm over the training sentences (for which the correct dependency structures are known) and recording the appropriate parser actions that result in the formation of the correct dependency trees, coupled with the features that represent the state of the parser mentioned in section 2.1. An evaluation of the resulting dependency parser and its efficacy in aiding HPSG parsing is presented in section 4.

3.2 Parsing with dependency constraints

Given a set of dependencies, the bottom-up process of HPSG parsing can be constrained so that it does not violate the given dependencies. This can be achieved by a simple extension of the parsing algorithm, as follows. During parsing, we store the lexical head of each partial parse tree. In each schema application, we can determine which child is the head; for example, the left child is the head when we apply the Head-Complement Schema. Given this information and lexical heads, the parser can identify the dependency produced by this schema application, and can therefore judge whether the schema application violates the dependency constraints.

This method forces the HPSG parser to produce parse trees that strictly conform to the output of the dependency parser. However, this means that the HPSG parser outputs no successful parse results when it cannot find the parse tree that is completely consistent with the given dependencies. This situation may occur when the dependency parser produces structures that are not covered in the HPSG grammar. This is especially likely with a fully data-driven dependency parser that uses local classification, since its output may not be globally consistent grammatically. In addition, the HPSG grammar is extracted from the HPSG Treebank using a corpus-based procedure, and it does not necessarily cover all possible grammatical phenomena in unseen text (Miyao and Tsujii, 2005).

We therefore propose an extension of this approach that uses predetermined dependencies as *soft* constraints. Violations of schema applications are detected in the same way as before, but instead of strictly prohibiting schema applications, we penalize the log-likelihood of partial parse trees created by schema applications that violate the dependencies constraints. Given a negative value α, we add α to the log-probability of a partial parse tree when the schema application violates the dependency constraints. That is, when a parse tree violates n dependencies, the log-probability of the parse tree is lowered by $n\alpha$. The meta parameter α is determined so as to maximize the accuracy on the development set.

Soft dependency constraints can be implemented as explained above as a straightforward extension of the parsing algorithm. In addition, it is easily integrated with beam thresholding methods of parsing. Because beam thresholding discards partial parse trees that have low log-probabilities, we can expect that the parser would discard partial parse trees based on violation of the dependency constraints.

4 Experiments

We evaluate the accuracy of HPSG parsing with dependency constraints on the HPSG Treebank (Miyao et al., 2003), which is extracted from the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993)[1]. Sections 02-21 were used for training (for HPSG and dependency parsers), section 22 was used as development data, and final testing was performed on section 23. Following previous work on wide-coverage parsing with lexicalized grammars using the Penn Treebank, we evaluate the parser by measuring the accuracy of predicate-argument relations in the parser's output. A predicate-argument relation is defined as a tuple $\langle \sigma, w_h, a, w_a \rangle$, where σ is the predicate type (e.g. adjective, intransitive verb), w_h is the head word of the predicate, a is the argument label (MODARG, ARG1, ... , ARG4), and w_a is the head word of the argument. Labeled precision (LP)/labeled recall (LR) is the ratio of tuples correctly identified by the parser. These predicate-argument relations cover the full range of syntactic dependencies produced by the HPSG parser (including, long-distance dependencies, raising and control, in addition to surface dependencies).

In the experiments presented in this section, input sentences were automatically tagged with parts-of-speech with about 97% accuracy, using a maximum entropy POS tagger. We also report results on parsing text with gold standard POS tags, where explicitly noted. This provides an upper-bound on what can be expected if a more sophisticated multi-tagging scheme (James R. Curran and Vadas, 2006) is used, instead of hard assignment of single tags in a preprocessing step as done here.

[1]The extraction software can be obtained from http://www-tsujii.is.s.u-tokyo.ac.jp/enju.

4.1 Baseline

HPSG parsing results using the same HPSG grammar and treebank have recently been reported by Miyao and Tsujii (2005) and Ninomia et al. (2006). By running the HPSG parser described in section 2.2 on the development data *without* dependency constraints, we obtain similar values of LP (86.8%) and LR (85.6%) as those reported by Miyao and Tsujii (Miyao and Tsujii, 2005). Using the extremely lexicalized framework of (Ninomiya et al., 2006) by performing supertagging before parsing, we obtain similar accuracy as Ninomiya et al. (87.1% LP and 85.9% LR).

4.2 Dependency constraints and the penalty parameter

Parsing the development data with *hard dependency constraints* confirmed the intuition that these constraints often describe dependency structures that do not conform to HPSG schema used in parsing, resulting in parse failures. To determine the upper-bound on HPSG parsing with hard dependency constraints, we set the HPSG parser to disallow the application of any rules that result in the creation of dependencies that violate *gold standard dependencies*. This results in high precision (96.7%), but recall is low (82.3%) due to parse failures caused by lack of grammatical coverage [2]. Using dependencies produced by the shift-reduce SVM parser, we obtain 91.5% LP and 65.7% LR. This represents a large gain in precision over the baseline, but an even greater loss in recall, which limits the usefulness of the parser, and severely hurts the appeal of hard constraints.

We focus the rest of our experiments on parsing with *soft dependency constraints*. As explained in section 3, this involves setting the penalty parameter α. During parsing, we subtract α from the log-probability of applying any schema that violates the dependency constraints given to the HPSG parser. Figure 3 illustrates the effect of α when gold standard dependencies (and gold standard POS tags) are used. We note that setting $\alpha = 0$ causes the parser to ignore dependency constraints, providing baseline performance. Conversely, setting a high enough value ($\alpha = 30$ is sufficient, in practice) causes any substructures that violate the dependency constraints to be used only when they are absolutely necessary to produce a valid parse for the input sentence. In figure 3, this corresponds to an upper-bound on the accuracy of parsing with soft dependency constraints (94.7% f-score), since gold standard dependencies are used.

[2]Although the HPSG grammar does not have perfect coverage of unseen text, it supports complete and *mostly* correct analyses for all sentences in the development set. However, when we require *completely* correct analyses by using hard constraints, lack of coverage may cause parse failures.

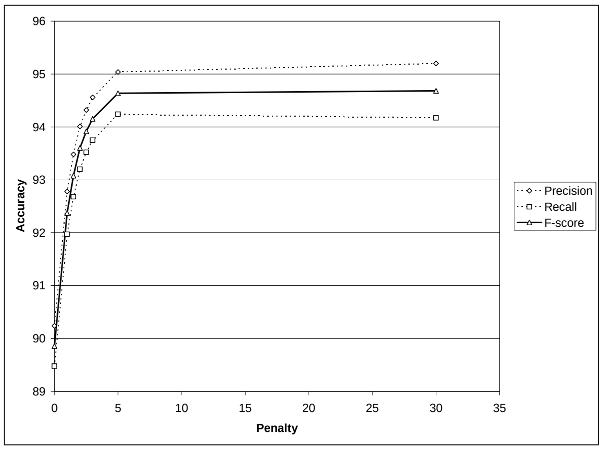

Figure 3: The effect of α on HPSG parsing constrained by gold standard dependencies.

We set α empirically with simple hill climbing on the development set. Because it is expected that the optimal value of α depends on the accuracy of the surface dependency parser, we set separate values for parsing with a POS tagger or with gold standard POS tags. Figure 4 shows the accuracy of HPSG predicate-argument relations obtained with dependency constraints determined by dependency parsing with gold standard POS tags. With both automatically assigned and gold standard POS tags, we observe an improvement of about 0.6% in precision, recall and f-score, when the optimal α value is used in each case. While this corresponds to a relative error reduction of over 6% (or 12%, if we consider the upper-bound dictated by imperfect grammatical coverage), a more interesting aspect of this framework is that it allows techniques designed for improving dependency accuracy to improve HPSG parsing accuracy directly, as we illustrate next.

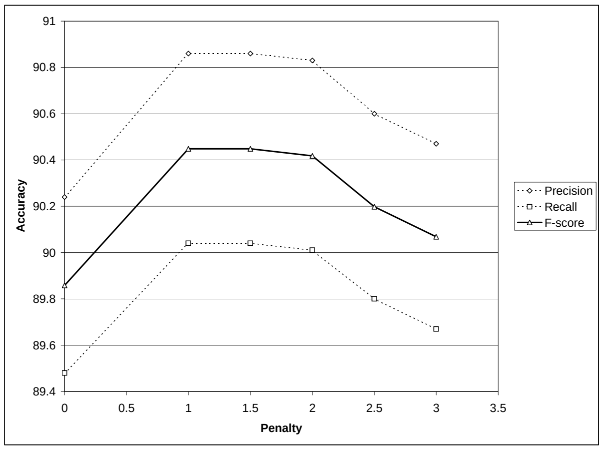

Figure 4: The effect of α on HPSG parsing constrained by the output of a dependency parser using gold standard POS tags.

4.3 Determining constraints with dependency parser combination

Parser combination has been shown to be a powerful way to obtain very high accuracy in dependency parsing (Sagae and Lavie, 2006). Using dependency constraints allows us to improve HPSG parsing accuracy simply by using an existing parser combination approach. As a first step, we train two additional parsers with the dependencies extracted from the HPSG Treebank. The first uses the same shift-reduce framework described in section 2.1, but it process the input from right to left (RL). This has been found to work well in previous work on dependency parser combination (Zeman and Žabokrtský, 2005; Sagae and Lavie, 2006). The second parser is MSTParser, the large-margin maximum spanning tree parser described in (McDonald et al., 2005)[3].

We examine the use of two combination schemes: one using two parsers, and one using three parsers. The first combination approach is to keep only dependencies for which there is agreement between the two parsers. In other words, dependencies that are proposed by one parser but not the other are simply discarded. Using the left-to-right shift-reduce parser and MSTParser, we find that this results in very high precision of surface dependencies on the development data. In the second approach, combination of the three dependency parsers is done according to the maximum spanning tree combination scheme of Sagae and Lavie (2006), which results in high accuracy of surface dependencies. For each of the combination approaches, we use the resulting dependencies as constraints for HPSG parsing, determining the optimal value of α on the development set in the same way as done for a single parser. Table 1 summarizes our experiments on development data using parser combinations to produce dependency constraints [4]. The two combination approaches are denoted as C1 and C2.

Parser	Dep	α	HPSG	Diff
none (baseline)	–	–	86.5	–
LR shift-reduce	91.2	1.5	87.1	0.6
RL shift-reduce	90.1	–	–	
MSTParser	91.0	–	–	
C1 (agreement)	96.8*	2.5	87.4	0.9
C2 (MST)	92.4	2.5	87.4	0.9

Table 1: Summary of results on development data. * The shallow accuracy of combination C1 corresponds to the dependency precision (no dependencies were reported for 8% of all words in the development set).

4.4 Results

Having determined α values on development data for the shift-reduce dependency parser, the two-parser agreement combination, and the three-parser maximum spanning tree combination, we parse the test data (section 23) using these three different sources of dependency constraints for HPSG parsing. Our final results are shown in table 2, where we also include the results published in (Ninomiya et al., 2006) for comparison purposes, and the result of using dependency constraints obtained with gold standard POS tags.

By using two unlabeled dependency parsers to provide soft dependency constraints, we obtain a 1% absolute improvement in precision and recall of predicate-argument identification in HPSG parsing over a strong baseline. Our baseline approach outperformed previously published results on this test

[3]Downloaded from http://sourceforge.net/projects/mstparser

[4]The accuracy figures for the dependency parsers is expressed as unlabeled accuracy of the surface dependencies only, and are not comparable to the HPSG parsing accuracy figures

Parser	LP	LR	F-score
HPSG Baseline	87.4	87.0	87.2
Shift-Reduce + HPSG	88.2	87.7	87.9
C1 + HPSG	**88.5**	**88.0**	**88.2**
C2 + HPSG	88.4	87.9	88.1
Baseline(gold)	89.8	89.4	89.6
Shift-Reduce(gold)	90.62	90.23	90.42
C1+HPSG(gold)	**90.9**	**90.4**	**90.6**
C2+HPSG(gold)	90.8	90.4	90.6
Miyao and Tsujii, 2005	85.0	84.3	84.6
Ninomiya et al., 2006	87.4	86.3	86.8

Table 2: Final results on test set. The first set of results show our HPSG baseline and HPSG with soft dependency constraints using three different sources of dependency constraints. The second set of results show the accuracy of the same parsers when gold part-of-speech tags are used. The third set of results is from existing published models on the same data.

set, and our best performing combination scheme obtains an absolute improvement of 1.4% over the best previously published results using the HPSG Treebank. It is interesting to note that the results obtained with dependency parser combinations C1 and C2 were very similar, even though in C1 only two parsers were used, and constraints were provided for about 92% of shallow dependencies (with accuracy higher than 96%). Clearly, precision is crucial in dependency constraints.

Finally, although it is necessary to perform dependency parsing to pre-compute dependency constraints, the total time required to perform the entire process of HPSG parsing with dependency constraints is close to that of the baseline HPSG approach. This is due to two reasons: (1) the dependency parsing approaches used to pre-compute constraints are several times faster than the baseline HPSG approach, and (2) the HPSG portion of the process is significantly faster when dependency constraints are used, since the constraints help sharpen the search space, making search more efficient. Using the baseline HPSG approach, it takes approximately 25 minutes to parse the test set. The total time required to parse the test set using HPSG with dependency constraints generated by the shift-reduce parser is 27 minutes. With combination C1, parsing time increases to 30 minutes, since two dependency parsers are used sequentially.

5 Related work

There are other approaches that combine shallow processing with deep parsing (Crysmann et al., 2002; Frank et al., 2003; Daum et al., 2003) to improve parsing *efficiency*. Typically, shallow parsing is used to create robust minimal recursion semantics, which are used as constraints to limit ambiguity during parsing. Our approach, in contrast, uses syntactic dependencies to achieve a significant improvement in the *accuracy* of wide-coverage HPSG parsing. Additionally, our approach is in many ways similar to supertagging (Bangalore and Joshi, 1999), which uses sequence labeling techniques as an efficient way to pre-compute parsing constraints (specifically, the assignment of lexical entries to input words).

6 Conclusion

We have presented a novel framework for taking advantage of the strengths of a shallow parsing approach and a deep parsing approach. We have shown that by constraining the application of rules in HPSG parsing according to results from a dependency parser, we can significantly improve the accuracy of deep parsing by using shallow syntactic analyses.

To illustrate how this framework allows for improvements in the accuracy of dependency parsing to be used directly to improve the accuracy of HPSG parsing, we showed that by combining the results of different dependency parsers using the search-based parsing ensemble approach of (Sagae and Lavie, 2006), we obtain improved HPSG parsing accuracy as a result of the improved dependency accuracy.

Although we have focused on the use of HPSG and dependency parsing, the general framework presented here can be applied to other lexicalized grammar formalisms, such as LTAG, CCG and LFG.

Acknowledgements

This research was partially supported by Grant-in-Aid for Specially Promoted Research 18002007.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

A. Berger, S. A. Della Pietra, and V. J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Joan Bresnan. 1982. *The mental representation of grammatical relations*. MIT Press.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Natural Language Learning*. New York, NY.

M. Collins. 1999. *Head-Driven Models for Natural Language Parsing*. Phd thesis, University of Pennsylvania.

Berthold Crysmann, Anette Frank, Bernd Kiefer, Stefan Mueller, Guenter Neumann, Jakub Piskorski, Ulrich Schaefer, Melanie Siegel, Hans Uszkoreit, Feiyu Xu, Markus Becker, and Hans-Ulrich Krieger. 2002. An integrated architecture for shallow and deep processing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*.

Michael Daum, Kilian A. Foth, and Wolfgang Menzel. 2003. Constraint-based integration of deep and shallow parsing techniques. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the International Conference on Computational Linguistics (COLING'96)*. Copenhagen, Denmark.

Anette Frank, Markus Becker, Berthold Crysmann, Bernd Kiefer, and Ulrich Schaefer. 2003. Integrated shallow and deep parsing: TopP meets HPSG. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pages 104–111.

Stephen Clark James R. Curran and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of COLING/ACL 2006*. Sydney, Australia.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 2002 Conference on Natural Language Learning*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewics. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19.

Ryan McDonald, Fernando Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Conference on Human Language Technologies/Empirical Methods in Natural Language Processing (HLT-EMNLP)*. Vancouver, Canada.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*. Ann Arbor, MI.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2003. Corpus oriented grammar development for aquiring a head-driven phrase structure grammar from the penn treebank. In *Proceedings of the Tenth Conference on Natural Language Learning*.

T. Ninomiya, T. Matsuzaki, Y. Tsuruoka, Y. Miyao, and J. Tsujii. 2006. Extremely lexicalized models for accurate and fast hpsg parsing. In *Proceedings of the 2006 Conference on Empirical Methods for Natural Language Processing (EMNLP 2006)*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70. Geneva, Switzerland.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit, and S. Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Natural Language Learning*. New York, NY.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technologies*. Vancouver, BC.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of the 2006 Meeting of the North American ACL*. New York, NY.

Yves Schabes, Anne Abeille, and Aravind Joshi. 1988. Parsing strategies with lexicalized grammars: Application to tree adjoining grammars. In *Proceedings of 12th COLING*.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press.

Daniel Zeman and Zdenek Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the International Workshop on Parsing Technologies*. Vancouver, Canada.