Enhancing Performance of Lexicalised Grammars

Rebecca Dridan†, Valia Kordoni†, Jeremy Nicholson†‡
†Dept of Computational Linguistics, Saarland University and DFKI GmbH, Germany
‡Dept of Computer Science and Software Engineering and NICTA, University of Melbourne, Australia
{rdrid,kordoni}@coli.uni-sb.de, jeremymn@csse.unimelb.edu.au

Abstract

This paper describes how external resources can be used to improve parser performance for heavily lexicalised grammars, looking at both robustness and efficiency. In terms of robustness, we try using different types of external data to increase lexical coverage, and find that simple POS tags have the most effect, increasing coverage on unseen data by up to 45%. We also show that filtering lexical items in a supertagging manner is very effective in increasing efficiency. Even using vanilla POS tags we achieve some efficiency gains, but when using detailed lexical types as supertags we manage to halve parsing time with minimal loss of coverage or precision.

1 Introduction

Heavily lexicalised grammars have been used in applications such as machine translation and information extraction because they can produce semantic structures which provide more information than less informed parsers. In particular, because of the structural and semantic information attached to lexicon items, these grammars do well at describing complex relationships, like non-projectivity and center embedding. However, the cost of this additional information sometimes makes deep parsers that use these grammars impractical. Firstly because, if the information is not available, the parsers may fail to produce an analysis, a failure of robustness. Secondly, the effect of analysing the extra information can slow the parser down, causing efficiency problems. This paper describes experiments aimed at improving parser performance in these two areas, by annotating the input given to one such deep parser, the PET parser (Callmeier, 2000), which uses lexicalised grammars developed under the HPSG formalism (Pollard and Sag, 1994).

2 Background

In all heavily lexicalised formalisms, such as LTAG, CCG, LFG and HPSG, the lexicon plays a key role in parsing. But a lexicon can never hope to contain all words in open domain text, and so lexical coverage is a central issue in boosting parser robustness. Some systems use heuristics based on numbers, capitalisation and perhaps morphology to guess the category of the unknown word (van Noord and Malouf, 2004), while others have focused on automatically expanding the lexicon (Baldwin, 2005; Hockenmaier et al., 2002; O'Donovan et al., 2005). Another method, described in Section 4, uses external resources such as part-of-speech (POS) tags to select generic lexical entries for out-of-vocabulary words. In all cases, we lose some of the depth of information the hand-crafted lexicon would provide, but an analysis is still produced, though possibly less than fully specified.

The central position of these detailed lexicons causes problems, not only of robustness, but also of efficiency and ambiguity. Many words may have five, six or more lexicon entries associated with them, and this can lead to an enormous search space for the parser. Various means of filtering this search space have been attempted. Kiefer et al. (1999) describes a method of filtering lexical items by specifying and checking for required prefixes and particles

Proceedings of ACL-08: HLT, pages 613–621,
Columbus, Ohio, USA, June 2008. ©2008 Association for Computational Linguistics

which is particularly effective for German, but also applicable to English. Other research has looked at using dependencies to restrict the parsing process (Sagae et al., 2007), but the most well known filtering method is supertagging. Originally described by Bangalore and Joshi (1994) for use in LTAG parsing, it has also been used very successfully for CCG (Clark, 2002). Supertagging is the process of assigning probable 'supertags' to words before parsing to restrict parser ambiguity, where a supertag is a tag that includes more specific information than the typical POS tags. The supertags used in each formalism differ, being elementary trees in LTAG and CCG categories for CCG. Section 3.2 describes an experiment akin to supertagging for HPSG, where the supertags are HPSG lexical types. Unlike elementary trees and CCG categories, which are predominantly syntactic categories, the HPSG lexical types contain a lot of semantic information, as well as syntactic.

In the case study we describe here, the tools, grammars and treebanks we use are taken from work carried out in the DELPH-IN[1] collaboration. This research is based on using HPSG along with Minimal Recursion Semantics (MRS: Copestake et al. (2001)) as a platform to develop deep natural language processing tools, with a focus on multilinguality. The grammars are designed to be bidirectional (used for generation as well as parsing) and so contain very specific linguistic information. In this work, we focus on techniques to improve parsing, not generation, but, as all the methods involve pre-processing and do not change the grammar itself, we do not affect the generation capabilities of the grammars. We use two of the DELPH-IN wide-coverage grammars: the English Resource Grammar (ERG: Copestake and Flickinger (2000)) and a German grammar, GG (Müller and Kasper, 2000; Crysmann, 2003). We also use the PET parser, and the [incr tsdb()] system profiler and treebanking tool (Oepen, 2001) for evaluation.

3 Parser Restriction

An exhaustive parser, such as PET, by default produces every parse licensed by the grammar. However, in many application scenarios, this is unnecessary and time consuming. The benefits of using a deep parser with a lexicalised grammar are the precision and depth of the analysis produced, but this depth comes from making many fine distinctions which greatly increases the parser search space, making parsing slow. By restricting the lexical items considered during parsing, we improve the efficiency of a parser with a possible trade-off of losing correct parses. For example, the noun phrase reading of *The dog barks* is a correct parse, although unlikely. By blocking the use of *barks* as a noun in this case, we lose this reading. This may be an acceptable trade-off in some applications that can make use of the detailed information, but only if it can be delivered in reasonable time. An example of such an application is the real-time speech translation system developed in the Verbmobil project (Wahlster, 2000), which integrated deep parsing results, where available, into its appointment scheduling and travel planning dialogues. In these experiments we look at two methods of restricting the parser, first by using POS tags and then using lexical types. To control the trade-off between efficiency and precision, we vary which lexical items are restricted according to a likelihood threshold from the respective taggers. Only open class words are restricted, since it is the gross distinctions between, for instance, noun and verb that we would like to utilise. Any differences between categories for closed class words are more subtle and we feel the parser is best left to make these distinctions without restriction. The data set used for these experiments is the *jh5* section of the treebank released with the ERG. This text consists of edited written English in the domain of Norwegian hiking instructions from the LOGON project (Oepen et al., 2004).

3.1 Part of Speech Tags

We use TreeTagger (Schmid, 1994) to produce POS tags and then open class words are restricted if the POS tagger assigned a tag with a probability over a certain threshold. A lower threshold will lead to faster parsing, but at the expense of losing more correct parses. We experiment with various thresholds, and results are shown in Table 1. Since a gold standard treebank for our data set was available, it was possible to evaluate the accuracy of the parser. Evaluation of deep parsing results is often reported only in terms of coverage (number of sentences which re-

[1]http://wiki.delph-in.net/

Threshold	Coverage	Precision	Time
gold	93.5%	92.2%	N/A
unrestricted	93.3%	92.4%	0.67s
1.00	90.7%	91.9%	0.59s
0.98	88.8%	89.3%	0.49s
0.95	88.4%	89.5%	0.48s
0.90	86.4%	88.5%	0.44s
0.80	84.3%	87.0%	0.43s
0.60	81.5%	87.3%	0.39s

Table 1: Results obtained when restricting the parser lexicon according to the POS tag, where words are restricted according to a threshold of POS probabilities.

Configuration	Coverage	Precision	Time
gold	93.5%	92.2%	N/A
unrestricted	93.3%	92.4%	0.67s
0.98 with POS	93.5%	91.9%	0.63s
0.95 with POS	93.1%	92.4%	0.48s
0.90 with POS	92.9%	92.3%	0.37s
0.80 with POS	91.8%	91.8%	0.31s
0.60 with POS	86.2%	93.5%	0.21s
0.98 no POS	92.9%	92.3%	0.62s
0.95 no POS	90.9%	91.0%	0.48s
0.90 no POS	87.7%	89.2%	0.42s
0.80 no POS	79.7%	84.6%	0.33s
0.60 no POS	67.0%	84.2%	0.23s

Table 2: Results obtained when restricting the parser lexicon according to the predicted lexical type, where words are restricted according to a threshold of tag probabilities. Two models, with and without POS tags as features, were used.

ceive an analysis), because, since the hand-crafted grammars are optimised for precision over coverage, the analyses are assumed to be correct. However, in this experiment, we are potentially 'diluting' the precision of the grammar by using external resources to remove parses and so it is important that we have some idea of how the accuracy is affected. In the table, precision is the percentage of sentences that, having produced at least one parse, produced a correct parse. A parse was judged to be correct if it exactly matched the gold standard tree in all aspects, syntactic and semantic.

The results show quite clearly how the coverage drops as the average parse time per sentence drops. In hybrid applications that can back-off to less informative analyses, this may be a reasonable trade-off, enabling detailed analyses in shorter times where possible, and using the shallower analyses otherwise.

3.2 Lexical Types

Another option for restricting the parser is to use the lexical types used by the grammar itself, in a similar method to that described by Prins and van Noord (2003). This could be considered a form of supertagging as used in LTAG and CCG. Restricting by lexical types should have the effect of reducing ambiguity further than POS tags can do, since one POS tag could still allow the use of multiple lexical items with compatible lexical types. On the other hand, it could be considered more difficult to tag accurately, since there are many more lexical types than POS tags (almost 900 in the ERG) and less training data is available.

While POS taggers such as TreeTagger are common, and there some supertaggers are available, notably that of Clark and Curran (2007) for CCG, no standard supertagger exists for HPSG. Consequently, we developed a Maximum Entropy model for supertagging using the OpenNLP implementation.[2] Similarly to Zhang and Kordoni (2006), we took training data from the gold–standard lexical types in the treebank associated with ERG (in our case, the July-07 version). For each token, we extracted features in two ways. One used features only from the input string itself: four characters from the beginning and end of the target word token, and two words of context (where available) either side of the target. The second used the features from the first, along with POS tags given by TreeTagger for the context tokens.

We held back the *jh5* section of the treebank for testing the Maximum Entropy model. Again, the lexical items that were to be restricted were controlled by a threshold, in this case the probability given by the maximum entropy model. Table 2 shows the results achieved by these two models, with the unrestricted results and the gold standard provided for comparison.

Here we see the same trends of falling coverage

[2]http://maxent.sourceforge.net/

with falling time for both models, with the POS tagged model consistently outperforming the word-form model. To give a clearer picture of the comparative performance of all three experiments, Figure 1 shows how the results vary with time for both models, and for the POS tag restricted experiment. Here we can see that the coverage and precision of the lexical type restriction experiment that uses the word-form model is just above that of the POS restricted one. However the POS tagged model clearly outperforms both, showing minimal loss of coverage or precision at a threshold which halved the average parsing time. At the lowest parsing time, we see that precision of the POS tagged model even goes up. This can be explained by noting that coverage here goes down, and obviously we are losing more incorrect parses than correct parses.

This echoes the main result from Prins and van Noord (2003), that filtering the lexical categories used by the parser can significantly reduce parsing time, while maintaining, or even improving, precision. The main differences between our method and that of Prins and van Noord are the training data and the tagging model. The key feature of their experiment was the use of 'unsupervised' training data, that is, the uncorrected output of their parser. In this experiment, we used gold standard training data, but much less of it (just under 200 000 words) and still achieved a very good precision. It would be interesting to see what amount of unsupervised parser output we would require to achieve the same level of precision. The other difference was the tagging model, maximum entropy versus Hidden Markov Model (HMM). We selected maximum entropy because Zhang and Kordoni (2006) had shown that they got better results using a maximum entropy tagger instead of a HMM one when predicting lexical types, albeit for a slightly different purpose. It is not possible to directly compare results between our experiments and those in Prins and van Noord, because of different languages, data sets and hardware, but it is worth noting that parsing times are much lower in our setup, perhaps more so than can be attributed to 4 years hardware improvement. While the range of sentence lengths appears to be very similar between the data sets, one possible reason for this could be the very large number of lexical categories used in their ALPINO system.

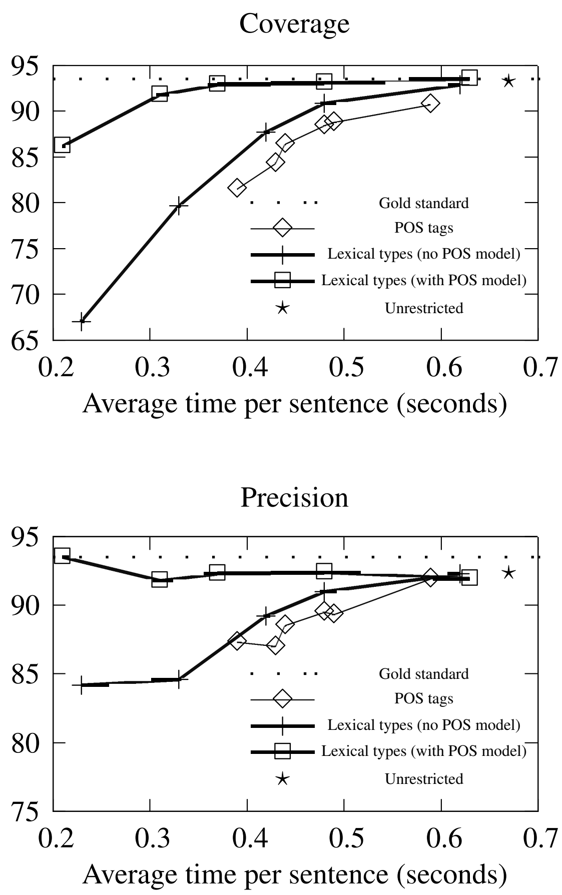

Figure 1: Coverage and precision varying with time for the three restriction experiments. Gold standard and unrestricted results shown for comparison.

While this experiment is similar to that of Clark and Curran (2007), it differs in that their supertagger assign categories to every word, while we look up every word in the lexicon and the tagger is used to filter what the lexicon returns, only if the tagger confidence is sufficiently high. As Table 2 shows, when we use the tags for which the tagger had a low confidence, we lose significant coverage. In order to run as a supertagger rather than a filter, the tagger would need to be much more accurate. While we can look at multi-tagging as an option, we believe much more training data would be needed to achieve a sufficient level of tag accuracy.

Increasing efficiency is important for enabling these heavily lexicalised grammars to bring the benefits of their deep analyses to applications, but similarly important is robustness. The following section is aimed at addressing this issue of robustness, again by using external information.

4 Unknown Word Handling

The lexical information available to the parser is what makes the depth of the analysis possible, and the default configuration of the parser uses an all-or-nothing approach, where a parse is not produced if all the lexical information is not available. However, in order to increase robustness, it is possible to use underspecified lexical information where a fully specified lexical item is not available. One method of doing this, built in to the PET parser, is to use POS tags to select generic lexical items, and hence allow a (less than fully specified) parse to be built.

The six data sets used for these experiments were chosen to give a range of languages and genres. Four sets are English text: *jh5* described in Section 3; *trec* consisting of questions from TREC and included in the treebanks released with the ERG; *a00* which is taken from the BNC and consists of factsheets and newsletters; and *depbank*, the 700 sentences of the Briscoe and Carroll version of DepBank (Briscoe and Carroll, 2006) taken from the Wall Street Journal. The last two data sets are German text: *clef700* consisting of German questions taken from the CLEF competition and *eiche564* a sample of sentences taken from a treebank parsed with the German HPSG grammar, GG and consisting of transcribed German speech data concerning appointment scheduling from the Verbmobil project. Vital statistics of these data sets are described in Table 3.

	Language	Number of Sentences	Ave. Sentence Length
jh5	English	464	14.2
trec	English	693	6.9
a00	English	423	17.2
depbank	English	700	21.5
clef	German	700	7.5
eiche564	German	564	11.5

Table 3: Data sets used in input annotation experiments.

We used TreeTagger to POS tag the six data sets, with the tagger configured to assign multiple tags, where the probability of the less likely tags was at least half that of the most likely tag. The data was input using a PET input chart (PIC), which allows POS tags to be assigned to each token, and then parsed each with the PET parser.[3] All English data sets used the July-07 CVS version of the ERG and the German sets used the September_2007 version of GG. Unlike the experiments described in Section 3, adding POS tags in this way will have no effect on sentences which the parser is already able to parse. The POS tags will only be considered when the parser has no lexicon entry for a given word, and hence can only increase coverage. Results are shown in Table 4, comparing the coverage over each set to that obtained without using POS tags to handle unknown words. Coverage here is defined as the percentage of sentences with at least one parse.

These results show very clearly one of the potential drawbacks of using a highly lexicalised grammar formalism like HPSG: unknown words are one of the main causes of parse failure, as quantified in Baldwin et al. (2004) and Nicholson et al. (2008). In the results here, we see that for *jh5*, *trec* and *eiche564*, adding unknown word handling made almost no difference, since the grammars (specifically the lexicons) have been tuned for these data sets. On the other hand, over unseen texts, adding unknown word handling made a dramatic difference to the coverage. This motivates strategies like the POS tag annotation used here, as well as the work on deep lexical acquisition (DLA) described in Zhang and Kordoni (2006) and Baldwin (2005), since no grammar could ever hope to cover all words used within a language.

As mentioned in Section 3, coverage is not the only evaluation metric that should be considered, particularly when adding potentially less precise information to the parsing process (in this case POS tags). Since the primary effect of adding POS tags is shown with those data sets for which we do not have gold standard treebanks, evaluating accuracy in this case is more difficult. However, in order to give some idea of the effects on precision, a sample of 100 sentences from the *a00* data set was evaluated for accuracy, for this and the following experiments.

[3]Subversion revision 384

In this instance, we found there was only a slight drop in precision, where the original analyses had a precision of 82% and the precision of the analyses when POS tags were used was 80%.

Since the parser has the means to accept named entity (NE) information in the input, we also experimented with using generic lexical items generated from NE data. We used SProUT (Becker et al., 2002) to tag the data sets and used PET's inbuilt NE handling mechanism to add NE items to the input, associated with the appropriate word tokens. This works slightly differently from the POS annotation mechanism, in that NE items are considered by the parser, even when the associated words are in the lexicon. This has the effect of increasing the number of analyses produced for sentences that already have a full lexical span, but could also increase coverage by enabling parses to be produced where there is no lexical span, or where no parse was possible because a token was not recognised as part of a name. In order to isolate the effect of the NE data, we ran one experiment where the input was annotated only with the SProUT data, and another where the POS tags were also added. These results are also in Table 4.

Again, we see coverage increases in the three unseen data sets, *a00*, *depbank* and *clef*, but not to the same extent as the POS tags. Examining the results in more detail, we find that the increases come almost exclusively from sentences without lexical span, rather than in sentences where a token was previously not recognised as part of a name. This means that the NE tagger is operating almost like a POS tagger that only tags proper nouns, and as the POS tagger tags proper nouns quite accurately, we find the NE tagger gives no benefit here. When examining the precision over our sample evaluation set from *a00*, we find that using the NE data alone adds no correct parses, while using NE data with POS tags actually removes correct parses when compared with POS alone, since the (in these cases, incorrect) NE data is preferred over the POS tags. It is possible that another named entity tagger would give better results, and this may be looked at in future experiments.

Other forms of external information might also be used to increase lexical coverage. Zhang and Kordoni (2006) reported a 20% coverage increase over baseline using a lexical type predictor for unknown words, and so we explored this avenue. The same maximum entropy tagger used in Section 3 was used and each open class word was tagged with its most likely lexical type, as predicted by the maximum entropy model. Table 5 shows the results, with the baseline and POS annotated results for comparison. As with the previous experiments, we see a coverage increase in those data sets which are considered unseen text for these grammars. Again it is clear that the use of POS tags as features obviously improves the maximum entropy model, since this second model has almost 10% better coverage on our unseen texts. However, lexical types do not appear to be as effective for increasing lexical coverage as the POS tags. One difference between the POS and lexical type taggers is that the POS tagger could produce multiple tags per word. Therefore, for the next experiment, we altered the lexical type tagger so it could also produce multiple tags. As with the TreeTagger configuration we used for POS annotation, extra lexical type tags were produced if they were at least half as probable as the most likely tag. A lower probability threshold of 0.01 was set, so that hundreds of tags of equal likelihood were not produced in the case where the tagger was unable to make an informed prediction. The results with multiple tagging are also shown in Table 5.

The multiple tagging version gives a coverage increase of between 2 and 10% over the single tag version of the tagger, but, at least for the English data sets, it is still less effective than straight-forward POS tagging. For the German unseen data set, *clef*, we do start getting above what the POS tagger can achieve. This may be in part because of the features used by the lexical type tagger — German, being a more morphologically rich language, may benefit more from the prefix and suffix features used in the tagger.

In terms of precision measured on our sample evaluation set, the single tag version of the lexical type tagger which used POS tag features achieved a very good precision of 87% where, of all the extra sentences that could now be parsed, only one did not have a correct parse. In an application where precision is considered much more important than coverage, this would be a good method of increasing coverage without loss of accuracy. The single tag version that did not use POS tags in the model achieved

	Baseline	with POS	NE only	NE+POS
jh5	93.1%	93.3%	93.1%	93.3%
trec	97.1%	97.5%	97.4%	97.7%
a00	50.1%	83.9%	53.0%	85.8%
depbank	36.3%	76.9%	51.1%	80.4%
clef	22.0%	67.7%	42.3%	75.3%
eiche564	63.8%	63.8%	64.0%	64.0%

Table 4: Parser coverage with baseline using no unknown word handling and unknown word handling using POS tags, SProUT named entity data as the only annotation, or SProUT tags in addition to POS annotation.

			Single Lexical Types		Multiple Lexical Types	
	Baseline	POS	-POS	+POS	-POS	+POS
jh5	93.1%	93.3%	93.3%	93.3%	93.5%	93.5%
trec	97.1%	97.5%	97.3%	97.4%	97.3%	97.4%
a00	50.1%	83.9%	63.8%	72.6%	65.7%	78.5%
depbank	36.3%	76.9%	51.7%	64.4%	53.9%	69.7%
clef	22.0%	67.7%	59.9%	66.8%	69.7%	76.9%
eiche564	63.8%	63.8%	63.8%	63.8%	63.8%	63.8%

Table 5: Parser coverage using a lexical type predictor for unknown word handling. The predictor was run in single tag mode, and then in multi-tag mode. Two different tagging models were used, with and without POS tags as features.

the same precision as with using only POS tags, but without the same increase in coverage. On the other hand, the multiple tagging versions, which at least started approaching the coverage of the POS tag experiment, dropped to a precision of around 76%.

From the results of Section 3, one might expect that at least the lexical type method of handling unknown words might at least lead to quicker parsing than when using POS tags, however POS tags are used differently in this situation. When POS tags are used to restrict the parser, any lexicon entry that unifies with the generic part-of-speech lexical category can be used by the parser. That is, when the word is restricted to, for example, a *verb*, any lexical item with one of the numerous more specific verb categories can be used. In contrast, in these experiments, the lexicon plays no part. The POS tag causes one underspecified lexical item (per POS tag) to be considered in parsing. While these underspecified items may allow more analyses to be built than if the exact category was used, the main contribution to parsing time turned out to be the number of tags assigned to each word, whether that was a POS tag or a lexical type. The POS tagger assigned multiple tags much less frequently than the multiple tagging lexical type tagger and so had a faster average parsing time. The single tagging lexical type tagger had only slightly fewer tags assigned overall, and hence was slightly faster, but at the expense of a significantly lower coverage.

5 Conclusion

The work reported here shows the benefits that can be gained by utilising external resources to annotate parser input in highly lexicalised grammar formalisms. Even something as simple and readily available (for languages likely to have lexicalised grammars) as a POS tagger can massively increase the parser coverage on unseen text. While annotating with named entity data or a lexical type supertagger were also found to increase coverage, the POS tagger had the greatest effect with up to 45% coverage increase on unseen text.

In terms of efficiency, POS tags were also shown to speed up parsing by filtering unlikely lexicon items, but better results were achieved in this case by using a lexical type supertagger. Again encouraging the use of external resources, the supertagging was found to be much more effective when POS tags were used to train the tagging model, and in this configuration, managed to halve the parsing time with minimal effect on coverage or precision.

6 Further Work

A number of avenues of future research were suggested by the observations made during this work. In terms of robustness and increasing lexical coverage, more work into using lexical types for unknown words could be explored. In light of the encouraging results for German, one area to look at is the effect of different features for different languages. Use of back-off models might also be worth considering when the tagger probabilities are low.

Different methods of using the supertagger could also be explored. The experiment reported here used the single most probable type for restricting the lexicon entries used by the parser. Two extensions of this are obvious. The first is to use multiple tags over a certain threshold, by either inputting multiple types as was done for the unknown word handling, or by using a generic type that is compatible with all the predicted types over a certain threshold. The other possible direction to try is to not check the predicted type against the lexicon, but to simply construct a lexical item from the most likely type, given a (high) threshold probability. This would be similar to the CCG supertagging mechanism and is likely to give generous speedups at the possible expense of precision, but it would be illuminating to discover how this trade-off plays out in our setup.

References

Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. 2004. Road-testing the English Resource Grammar over the British National Corpus. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2047–50, Lisbon, Portugal.

Timothy Baldwin. 2005. Bootstrapping deep lexical resources: Resources for courses. In *Proceedings of the ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, pages 67–76, Ann Arbor, USA.

Srinivas Bangalore and Aravind K. Joshi. 1994. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th COLING Conference*, pages 154–160, Kyoto, Japan.

Markus Becker, Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. 2002. SProUT - Shallow Processing with Typed Feature Structures and Unification. In *Proceedings of the International Conference on NLP (ICON 2002)*, Mumbai, India.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalised statistical parser on the PARC DepBank. In *Proceedings of the 44th Annual Meeting of the ACL*, pages 41–48, Sydney, Australia.

Ulrich Callmeier. 2000. PET - a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–107.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark. 2002. Supertagging for combinatory categorical grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammar and Related Frameworks*, pages 101–106, Venice, Italy.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of the Second conference on Language Resources and Evaluation (LREC-2000)*, Athens, Greece.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Meeting of the ACL and 10th Conference of the EACL (ACL-EACL 2001)*, Toulouse, France.

Berthold Crysmann. 2003. On the efficient implementation of German verb placement in HPSG. In *Proceedings of RANLP 2003*, pages 112–116, Borovets, Bulgaria.

Julia Hockenmaier, Gann Bierner, and Jason Baldridge. 2002. Extending the coverage of a CCG system. *Research in Language and Computation*.

Bernd Kiefer, Hans-Ulrich Krieger, John Carroll, and Rob Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 473–480, Maryland, USA.

Stefan Müller and Walter Kasper. 2000. HPSG analysis of German. In *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 238–253. Springer, Berlin, Germany.

Jeremy Nicholson, Valia Kordoni, Yi Zhang, Timothy Baldwin, and Rebecca Dridan. 2008. Evaluating and extending the coverage of HPSG grammars. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2008)*, Marrakech, Morocco.

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III treebanks. *Computational Linguistics*, 31:pp 329–366.

Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. 2004. Somå 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.

Stephan Oepen. 2001. [incr tsdb()] – competence and performance laboratory. User manual, Computational Linguistics, Saarland University, Saarbrücken, Germany.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA.

Robbert Prins and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139.

Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG parsing with shallow dependency constraints. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 624–631, Prague, Czech Republic.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Gertjan van Noord and Robert Malouf. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop Beyond Shallow Analyses – Formalisms and statistical modelling for deep analyses*.

Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer-Verlag, Berlin.

Yi Zhang and Valia Kordoni. 2006. Automated deep lexical acquisition for robust open texts processing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 275–280, Genoa, Italy.