

The Importance of Supertagging for Wide-Coverage CCG Parsing

Stephen Clark

School of Informatics
University of Edinburgh
2 Buccleuch Place, Edinburgh, UK
stephen.clark@ed.ac.uk

James R. Curran

School of Information Technologies
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Abstract

This paper describes the role of supertagging in a wide-coverage CCG parser which uses a log-linear model to select an analysis. The supertagger reduces the derivation space over which model estimation is performed, reducing the space required for discriminative training. It also dramatically increases the speed of the parser. We show that large increases in speed can be obtained by tightly integrating the supertagger with the CCG grammar and parser. This is the first work we are aware of to successfully integrate a supertagger with a full parser which uses an automatically extracted grammar. We also further reduce the derivation space using constraints on category combination. The result is an accurate wide-coverage CCG parser which is an order of magnitude faster than comparable systems for other linguistically motivated formalisms.

1 Introduction

Lexicalised grammar formalisms such as Lexicalized Tree Adjoining Grammar (LTAG) and Combinatory Categorical Grammar (CCG) assign one or more syntactic structures to each word in a sentence which are then manipulated by the parser. Supertagging was introduced for LTAG as a way of increasing parsing efficiency by reducing the number of structures assigned to each word (Bangalore and Joshi, 1999). Supertagging has more recently been applied to CCG (Clark, 2002; Curran and Clark, 2003).

Supertagging accuracy is relatively high for manually constructed LTAGs (Bangalore and Joshi, 1999). However, for LTAGs extracted automatically from the Penn Treebank, performance is much lower (Chen et al., 1999; Chen et al., 2002). In fact, performance for such grammars is below that needed for successful integration into a full parser (Sarkar et al., 2000). In this paper we demonstrate that CCG supertagging accuracy is not only sufficient for accurate and robust parsing using an automatically extracted grammar, but also offers several practical advantages.

Our wide-coverage CCG parser uses a log-linear model to select an analysis. The model parameters are estimated using a discriminative method, that is, one which requires all incorrect parses for a sentence as well as the correct parse. Since an automatically extracted CCG grammar can produce an extremely large number of parses, the use of a supertagger is crucial in limiting the total number of parses for the training data to a computationally manageable number.

The supertagger is also crucial for increasing the speed of the parser. We show that spectacular increases in speed can be obtained, without affecting accuracy or coverage, by tightly integrating the supertagger with the CCG grammar and parser. To achieve maximum speed, the supertagger initially assigns only a small number of CCG categories to each word, and the parser only requests more categories from the supertagger if it cannot provide an analysis. We also demonstrate how extra constraints on the category combinations, and the application of beam search using the parsing model, can further increase parsing speed.

This is the first work we are aware of to successfully integrate a supertagger with a full parser which uses a lexicalised grammar automatically extracted from the Penn Treebank. We also report significantly higher parsing speeds on newspaper text than any previously reported for a full wide-coverage parser. Our results confirm that wide-coverage CCG parsing is feasible for many large-scale NLP tasks.

2 CCG Supertagging

Parsing using CCG can be viewed as a two-stage process: first assign lexical categories to the words in the sentence, and then combine the categories together using CCG's combinatory rules.¹ The first stage can be accomplished by simply assigning to each word all categories from the word's entry in the lexicon (Hockenmaier, 2003).

¹See Steedman (2000) for an introduction to CCG, and see Clark et al. (2002) and Hockenmaier (2003) for an introduction to wide-coverage parsing using CCG.

The WSJ is a publication that I enjoy reading
 $\overline{NP/N}$ \overline{N} $\overline{(S[decl]\backslash NP)/NP}$ $\overline{NP/N}$ \overline{N} $\overline{(NP\backslash NP)/(S[decl]/NP)}$ \overline{NP} $\overline{(S[decl]\backslash NP)/(S[ng]\backslash NP)}$ $\overline{(S[ng]\backslash NP)/NP}$

Figure 1: Example sentence with CCG lexical categories

frequency cut-off	# cat types	# cat tokens in 2-21 not in cat set	# sentences in 2-21 with missing cat	# cat tokens in 00 not in cat set	# sentences in 00 with missing cat
1	1 225	0	0	12 (0.03%)	12 (0.6%)
10	409	1 933 (0.2%)	1 712 (4.3%)	79 (0.2%)	69 (3.6%)

Table 1: Statistics for the lexical category set

An alternative is to use a statistical tagging approach to assign one or more categories. A statistical model can be used to determine the most likely categories given the word’s context. The advantage of this *supertagging* approach is that the number of categories assigned to each word can be reduced, with a correspondingly massive reduction in the number of derivations.

Bangalore and Joshi (1999) use a standard Markov model tagger to assign LTAG elementary trees to words. Here we use the Maximum Entropy models described in Curran and Clark (2003). An advantage of the Maximum Entropy approach is that it is easy to encode a wide range of potentially useful information as features; for example, Clark (2002) has shown that POS tags provide useful information for supertagging. The next section describes the set of lexical categories used by our supertagger and parser.

2.1 The Lexical Category Set

The set of lexical categories is obtained from CCGbank (Hockenmaier and Steedman, 2002; Hockenmaier, 2003), a corpus of CCG normal-form derivations derived semi-automatically from the Penn Treebank. Following Clark (2002), we apply a frequency cutoff to the training set, only using those categories which appear at least 10 times in sections 2-21. Figure 1 gives an example sentence supertagged with the correct CCG lexical categories.

Table 1 gives the number of different category types and shows the coverage on training (seen) and development (unseen) data (section 00 from CCGbank). The table also gives statistics for the complete set containing every lexical category type in CCGbank.² These figures show that using a frequency cutoff can significantly reduce the size of the category set with only a small loss in coverage.

²The numbers differ slightly from those reported in Clark (2002) since a newer version of CCGbank is being used here.

Clark (2002) compares the size of grammars extracted from CCGbank with automatically extracted LTAGs. The grammars of Chen and Vijay-Shanker (2000) contain between 2,000 and 9,000 tree frames, depending on the parameters used in the extraction process, significantly more elementary structures than the number of lexical categories derived from CCGbank. We hypothesise this is a key factor in the higher accuracy for supertagging using a CCG grammar compared with an automatically extracted LTAG.

2.2 The Tagging Model

The supertagger uses probabilities $p(y|x)$ where y is a lexical category and x is a context. The conditional probabilities have the following log-linear form:

$$p(y|x) = \frac{1}{Z(x)} e^{\sum_i \lambda_i f_i(y,x)} \quad (1)$$

where f_i is a feature, λ_i is the corresponding weight, and $Z(x)$ is a normalisation constant. The context is a 5-word window surrounding the target word. Features are defined for each word in the window and for the POS tag of each word. Curran and Clark (2003) describes the model and explains how Generalised Iterative Scaling, together with a Gaussian prior for smoothing, can be used to set the weights.

The supertagger in Curran and Clark (2003) finds the single most probable category sequence given the sentence, and uses additional features defined in terms of the previously assigned categories. The per-word accuracy is between 91 and 92% on unseen data in CCGbank; however, Clark (2002) shows this is not high enough for integration into a parser since the large number of incorrect categories results in a significant loss in coverage.

Clark (2002) shows how the models in (1) can be used to define a *multi-tagger* which can assign more than one category to a word. For each word in the sentence, the multi-tagger assigns all those cat-

β	CATS/ WORD	ACC	SENT ACC	ACC (POS)	SENT ACC
0.1	1.4	97.0	62.6	96.4	57.4
0.075	1.5	97.4	65.9	96.8	60.6
0.05	1.7	97.8	70.2	97.3	64.4
0.01	2.9	98.5	78.4	98.2	74.2
0.01 _{k=100}	3.5	98.9	83.6	98.6	78.9
0	21.9	99.1	84.8	99.0	83.0

Table 2: Supertagger accuracy on section 00

egories whose probability according to (1) is within some factor, β , of the highest probability category for the word.

We follow Clark (2002) in ignoring the features based on the previously assigned categories; therefore every tagging decision is local and the Viterbi algorithm is not required. This simple approach has the advantage of being very efficient, and we find that it is accurate enough to enable highly accurate parsing. However, a method which used the forward-backward algorithm to sum over all possible sequences, or some other method which took into account category sequence information, may well improve the results.

For words seen at least k times in the training data, the tagger can only assign categories appearing in the word’s entry in the *tag dictionary*. Each entry in the tag dictionary is a list of all the categories seen with that word in the training data. For words seen less than k times, we use an alternative dictionary based on the word’s POS tag: the tagger can only assign categories that have been seen with the POS tag in the training data. A value of $k = 20$ was used in this work, and sections 2-21 of CCGbank were used as training data.

Table 2 gives the per-word accuracy (acc) on section 00 for various values of β , together with the average number of categories per word. The *sent acc* column gives the percentage of sentences whose words are all supertagged correctly. The figures for $\beta = 0.01_{k=100}$ correspond to a value of 100 for the tag dictionary parameter k . The set of categories assigned to a word is considered correct if it contains the correct category. The table gives results for gold standard POS tags and, in the final 2 columns, for POS tags automatically assigned by the Curran and Clark (2003) tagger. The drop in accuracy is expected given the importance of POS tags as features.

The figures for $\beta = 0$ are obtained by assigning all categories to a word from the word’s entry in the tag dictionary. For words which appear less than 20 times in the training data, the dictionary based on

the word’s POS tag is used. The table demonstrates the significant reduction in the average number of categories that can be achieved through the use of a supertagger. To give one example, the number of categories in the tag dictionary’s entry for the word *is* is 45 (only considering categories which have appeared at least 10 times in the training data). However, in the sentence *Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.*, the supertagger correctly assigns 1 category to *is* for $\beta = 0.1$, and 3 categories for $\beta = 0.01$.

3 The Parser

The parser is described in detail in Clark and Curran (2004). It takes POS tagged sentences as input with each word assigned a set of lexical categories. A packed chart is used to efficiently represent all of the possible analyses for a sentence, and the CKY chart parsing algorithm described in Steedman (2000) is used to build the chart.

Clark and Curran (2004) evaluate a number of log-linear parsing models for CCG. In this paper we use the normal-form model, which defines probabilities with the conditional log-linear form in (1), where y is a derivation and x is a sentence. Features are defined in terms of the local trees in the derivation, including lexical head information and word-word dependencies. The normal-form derivations in CCGbank provide the gold standard training data. The feature set we use is from the best performing normal-form model in Clark and Curran (2004).

For a given sentence the output of the parser is a dependency structure corresponding to the most probable derivation, which can be found using the Viterbi algorithm. The dependency relations are defined in terms of the argument slots of CCG lexical categories. Clark et al. (2002) and Clark and Curran (2004) give a detailed description of the dependency structures.

3.1 Model Estimation

In Clark and Curran (2004) we describe a discriminative method for estimating the parameters of a log-linear parsing model. The estimation method maximises the following objective function:

$$\begin{aligned}
 L'(\Lambda) &= L(\Lambda) - G(\Lambda) \\
 &= \log \prod_{j=1}^m P_{\Lambda}(d_j | S_j) - \sum_{i=1}^n \frac{\lambda_i^2}{2\sigma^2}
 \end{aligned} \tag{2}$$

The data consists of sentences S_1, \dots, S_m , together with gold standard normal-form derivations, d_1, \dots, d_m . $L(\Lambda)$ is the log-likelihood of model Λ , and $G(\Lambda)$ is a Gaussian prior term used to avoid

overfitting (n is the number of features; λ_i is the weight for feature f_i ; and σ is a parameter of the Gaussian). The objective function is optimised using L-BFGS (Nocedal and Wright, 1999), an iterative algorithm from the numerical optimisation literature.

The algorithm requires the gradient of the objective function, and the value of the objective function, at each iteration. Calculation of these values requires all derivations for each sentence in the training data. In Clark and Curran (2004) we describe efficient methods for performing the calculations using packed charts. However, a very large amount of memory is still needed to store the packed charts for the complete training data even though the representation is very compact; in Clark and Curran (2003) we report a memory usage of 30 GB. To handle this we have developed a parallel implementation of the estimation algorithm which runs on a Beowulf cluster.

The need for large high-performance computing resources is a disadvantage of our earlier approach. In the next section we show how use of the supertagger, combined with normal-form constraints on the derivations, can significantly reduce the memory requirements for the model estimation.

4 Generating Parser Training Data

Since the training data contains the correct lexical categories, we ensure the correct category is assigned to each word when generating the packed charts for model estimation. Whilst training the parser, the supertagger can be thought of as supplying a number of plausible but incorrect categories for each word; these, together with the correct categories, determine the parts of the parse space that are used in the estimation process. We would like to keep the packed charts as small as possible, but not lose accuracy in the resulting parser. Section 4.2 discusses the use of various settings on the supertagger. The next section describes how normal-form constraints can further reduce the derivation space.

4.1 Normal-Form Constraints

As well as the supertagger, we use two additional strategies for reducing the derivation space. The first, following Hockenmaier (2003), is to only allow categories to combine if the combination has been seen in sections 2-21 of CCGbank. For example, *NP/NP* could combine with *NP/NP* according to CCG’s combinatory rules (by forward composition), but since this particular combination does not appear in CCGbank the parser does not allow it.

The second strategy is to use Eisner’s normal-form constraints (Eisner, 1996). The constraints

SUPERTAGGING/PARSING CONSTRAINTS	USAGE	
	DISK	MEMORY
$\beta = 0.01 \rightarrow 0.05 \rightarrow 0.1$	17 GB	31 GB
CCGbank constraints	13 GB	23 GB
Eisner constraints	9 GB	16 GB
$\beta = 0.05 \rightarrow 0.1$	2 GB	4 GB

Table 3: Space requirements for model training data

prevent any constituent which is the result of a forward (backward) composition serving as the primary functor in another forward (backward) composition or a forward (backward) application. Eisner only deals with a grammar without type-raising, and so the constraints do not guarantee a normal-form parse when using a grammar extracted from CCGbank. However, the constraints are still useful in restricting the derivation space. As far as we are aware, this is the first demonstration of the utility of such constraints for a wide-coverage CCG parser.

4.2 Results (Space Requirements)

Table 3 shows the effect of different supertagger settings, and the normal-form constraints, on the size of the packed charts used for model estimation. The disk usage is the space taken on disk by the charts, and the memory usage is the space taken in memory during the estimation process. The training sentences are parsed using a number of nodes from a 64-node Beowulf cluster.³ The time taken to parse the training sentences depends on the supertagging and parsing constraints, and the number of nodes used, but is typically around 30 minutes.

The first row of the table corresponds to using the least restrictive β value of 0.01, and reverting to $\beta = 0.05$, and finally $\beta = 0.1$, if the chart size exceeds some threshold. The threshold was set at 300,000 nodes in the chart. Packed charts are created for approximately 94% of the sentences in sections 2-21 of CCGbank. The coverage is not 100% because, for some sentences, the parser cannot provide an analysis, and some charts exceed the node limit even at the $\beta = 0.1$ level. This strategy was used in our earlier work (Clark and Curran, 2003) and, as the table shows, results in very large charts.

Note that, even with this relaxed setting on the supertagger, the number of categories assigned to each word is only around 3 on average. This suggests that it is only through use of the supertagger that we are able to estimate a log-linear parsing model on all of the training data at all, since without it the memory

³The figures in the table are estimates based on a sample of the nodes in the cluster.

requirements would be far too great, even for the entire 64-node cluster.⁴

The second row shows the reduction in size if the parser is only allowed to combine categories which have combined in the training data. This significantly reduces the number of categories created using the composition rules, and also prevents the creation of unlikely categories using rule combinations not seen in CCGbank. The results show that the memory and disk usage are reduced by approximately 25% using these constraints.

The third row shows a further reduction in size when using the Eisner normal-form constraints. Even with the CCGbank rule constraints, the parser still builds many non-normal-form derivations, since CCGbank does contain cases of composition and type-raising. (These are used to analyse some coordination and extraction cases, for example.) The combination of the two types of normal-form constraints reduces the memory requirements by 48% over the original approach. In Clark and Curran (2004) we show that the parsing model resulting from training data generated in this way produces state-of-the-art CCG dependency recovery: 84.6 F-score over labelled dependencies.

The final row corresponds to a more restrictive setting on the supertagger, in which a value of $\beta = 0.05$ is used initially and $\beta = 0.1$ is used if the node limit is exceeded. The two types of normal-form constraints are also used. In Clark and Curran (2004) we show that using this more restrictive setting has a small negative impact on the accuracy of the resulting parser (about 0.6 F-score over labelled dependencies). However, the memory requirement for training the model is now only 4 GB, a reduction of 87% compared with the original approach.

5 Parsing Unseen Data

The previous section showed how to combine the supertagger and parser for the purpose of creating training data, assuming the correct category for each word is known. In this section we describe our approach to tightly integrating the supertagger and parser for parsing unseen data.

Our previous approach to parsing unseen data (Clark et al., 2002; Clark and Curran, 2003) was to use the least restrictive setting of the supertagger which still allows a reasonable compromise between speed and accuracy. Our philosophy was to give the parser the greatest possibility of finding the correct parse, by giving it as many categories as possible, while still retaining reasonable efficiency.

⁴Another possible solution would be to use sampling methods, e.g. Osborne (2000).

SUPERTAGGING/PARSING CONSTRAINTS	TIME SEC	SENTS /SEC	WORDS /SEC
$\beta = 0.01 \rightarrow \dots \rightarrow 0.1$	3 523	0.7	16
CCGbank constraints	1 181	2.0	46
Eisner constraints	995	2.4	55
$\beta = 0.1 \rightarrow \dots 0.01_{k=100}$	608	3.9	90
CCGbank constraints	124	19.4	440
Eisner constraints	100	24.0	546
Parser beam	67	35.8	814
94% coverage	49	49.0	1 114
Parser beam	46	52.2	1 186
Oracle	18	133.4	3 031

Table 4: Parse times for section 23

The problem with this approach is that, for some sentences, the number of categories in the chart still gets extremely large and so parsing is unacceptably slow. Hence we applied a limit to the number of categories in the chart, as in the previous section, and reverted to a more restrictive setting of the supertagger if the limit was exceeded. We first used a value of $\beta = 0.01$, and then reverted to $\beta = 0.05$, and finally $\beta = 0.1$.

In this paper we take the opposite approach: we start with a very restrictive setting of the supertagger, and only assign more categories if the parser cannot find an analysis spanning the sentence. In this way the parser interacts much more closely with the supertagger. In effect, the parser is using the grammar to decide if the categories provided by the supertagger are acceptable, and if not the parser requests more categories. The parser uses the 5 levels given in Table 2, starting with $\beta = 0.1$ and moving through the levels to $\beta = 0.01_{k=100}$.

The advantage of this approach is that parsing speeds are much higher. We also show that our new approach slightly increases parsing accuracy over the previous method. This suggests that, given our current parsing model, it is better to rely largely on the supertagger to provide the correct categories rather than use the parsing model to select the correct categories from a very large derivation space.

5.1 Results (Parse Times)

The results in this section are all using the best performing normal-form model in Clark and Curran (2004), which corresponds to row 3 in Table 3. All experiments were run on a 2.8 GHz Intel Xeon P4 with 2 GB RAM.

Table 4 gives parse times for the 2,401 sentences in section 23 of CCGbank. The final two columns give the number of sentences, and the number of

β	CATS/ WORD	0.1 FIRST		0.01 FIRST	
		PARSES	%	PARSES	%
0.1	1.4	1689	88.4	0	0.0
0.075	1.5	43	2.3	7	0.4
0.05	1.7	51	2.7	39	2.0
0.01	2.9	79	4.1	1816	95.1
0.01 _{k=100}	3.5	33	1.7	33	1.7
NO SPAN		15	0.8	15	0.8

Table 5: Supertagger β levels used on section 00

words, parsed per second. For all of the figures reported on section 23, unless stated otherwise, the parser is able to provide an analysis for 98.5% of the sentences. The parse times and speeds include the failed sentences, but do not include the time taken by the supertagger; however, the supertagger is extremely efficient, and takes less than 6 seconds to supertag section 23, most of which consists of load time for the Maximum Entropy model.

The first three rows correspond to our strategy of earlier work by starting with the least restrictive setting of the supertagger. The first value of β is 0.01; if the parser cannot find a spanning analysis, this is changed to $\beta = 0.01_{k=100}$; if the node limit is exceeded (for these experiments set at 1,000,000), β is changed to 0.05. If the node limit is still exceeded, β is changed to 0.075, and finally 0.1. The second row has the CCGbank rule restriction applied, and the third row the Eisner normal-form restrictions.

The next three rows correspond to our new strategy of starting with the least restrictive setting of the supertagger ($\beta = 0.1$), and moving through the settings if the parser cannot find a spanning analysis. The table shows that the normal-form constraints have a significant impact on the speed, reducing the parse times for the old strategy by 72%, and reducing the times for the new strategy by 84%. The new strategy also has a spectacular impact on the speed compared with the old strategy, reducing the times by 83% without the normal-form constraints and 90% with the constraints.

The 94% coverage row corresponds to using only the first two supertagging levels; the parser ignores the sentence if it cannot get an analysis at the $\beta = 0.05$ level. The percentage of sentences without an analysis is now 6%, but the parser is extremely fast, processing almost 50 sentences a second. This configuration of the system would be useful for obtaining data for lexical knowledge acquisition, for example, for which large amounts of data are required.

The oracle row shows the parser speed when it is provided with only the correct lexical categories.

The parser is extremely fast, and in Clark and Curran (2004) we show that the F-score for labelled dependencies is almost 98%. This demonstrates the large amount of information in the lexical categories, and the potential for improving parser accuracy and efficiency by improving the supertagger.

Finally, the first *parser beam* row corresponds to the parser using a beam search to further reduce the derivation space. The beam search works by pruning categories from the chart: a category can only be part of a derivation if its beam score is within some factor, α , of the highest scoring category for that cell in the chart. Here we simply use the exponential of the inside score of a category as the beam score; the inside score for a category c is the sum over all sub-derivations dominated by c of the weights of the features in those sub-derivations (see Clark and Curran (2004)).⁵

The value of α that we use here reduces the accuracy of the parser on section 00 by a small amount (0.3% labelled F-score), but has a significant impact on parser speed, reducing the parse times by a further 33%. The final *parser beam* row combines the beam search with the fast, reduced coverage configuration of the parser, producing speeds of over 50 sentences per second.

Table 5 gives the percentage of sentences which are parsed at each supertagger level, for both the new and old parsing strategies. The results show that, for the old approach, most of the sentences are parsed using the least restrictive setting of the supertagger ($\beta = 0.01$); conversely, for the new approach, most of the sentences are parsed using the most restrictive setting ($\beta = 0.1$).

As well as investigating parser efficiency, we have also evaluated the accuracy of the parser on section 00 of CCGbank, using both parsing strategies together with the normal-form constraints. The new strategy *increases* the F-score over labelled dependencies by approximately 0.5%, leading to the figures reported in Clark and Curran (2004).

5.2 Comparison with Other Work

The only other work we are aware of to investigate the impact of supertagging on parsing efficiency is the work of Sarkar et al. (2000) for LTAG. Sarkar et al. did find that LTAG supertagging increased parsing speed, but at a significant cost in coverage: only 1,324 sentences out of a test set of 2,250 received a parse. The parse times reported are also not as good as those reported here: the time taken to parse the 2,250 test sentences was over 5 hours.

⁵Multiplying by an estimate of the outside score may improve the efficacy of the beam.

Kaplan et al. (2004) report high parsing speeds for a deep parsing system which uses an LFG grammar: 1.9 sentences per second for 560 sentences from section 23 of the Penn Treebank. They also report speeds for the publicly available Collins parser (Collins, 1999): 2.8 sentences per second for the same set. The best speeds we have reported for the CCG parser are an order of magnitude faster.

6 Conclusions

This paper has shown that by tightly integrating a supertagger with a CCG parser, very fast parse times can be achieved for Penn Treebank WSJ text. As far as we are aware, the times reported here are an order of magnitude faster than any reported for comparable systems using linguistically motivated grammar formalisms. The techniques we have presented in this paper increase the speed of the parser by a factor of 77. This makes this parser suitable for large-scale NLP tasks.

The results also suggest that further improvements can be obtained by improving the supertagger, which should be possible given the simple tagging approach currently being used.

The novel parsing strategy of allowing the grammar to decide if the supertagging is likely to be correct suggests a number of interesting possibilities. In particular, we would like to investigate only repairing those areas of the chart that are most likely to contain errors, rather than parsing the sentence from scratch using a new set of lexical categories. This could further increase parsing efficiency.

Acknowledgements

We would like to thank Julia Hockenmaier, whose work creating the CCGbank made this research possible, and Mark Steedman for his advice and guidance. This research was supported by EPSRC grant GR/M96889, and a Commonwealth scholarship and a Sydney University Travelling scholarship to the second author.

References

- Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.
- John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGS from the Penn Treebank. In *Proceedings of IWPT 2000*, Trento, Italy.
- John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the 9th Meeting of EACL*, Bergen, Norway.
- John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. 2002. Reranking an N-gram su-
- pertagger. In *Proceedings of the TAG+ Workshop*, pages 259–268, Venice, Italy.
- Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the EMNLP Conference*, pages 97–104, Sapporo, Japan.
- Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, Barcelona, Spain.
- Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334, Philadelphia, PA.
- Stephen Clark. 2002. A supertagger for Combinatory Categorical Grammar. In *Proceedings of the TAG+ Workshop*, pages 19–24, Venice, Italy.
- Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.
- James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.
- Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorical Grammar. In *Proceedings of the 34th Meeting of the ACL*, pages 79–86, Santa Cruz, CA.
- Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third LREC Conference*, pages 1974–1981, Las Palmas, Spain.
- Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the HLT/NAACL Conference*, Boston, MA.
- Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York, USA.
- Miles Osborne. 2000. Estimation of stochastic attribute-value grammars using an informative sample. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 586–592, Saarbrücken, Germany.
- Anoop Sarkar, Fei Xia, and Aravind Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proceedings of the COLING Workshop on Efficiency in Large-Scale Parsing Systems*, pages 37–42, Luxembourg.
- Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.