Pruning the Search Space of a Hand-Crafted Parsing System with a Probabilistic Parser

Aoife Cahill
Dublin City University
acahill@computing.dcu.ie

Tracy Holloway King
PARC
thking@parc.com

John T. Maxwell III
PARC
maxwell@parc.com

Abstract

The demand for deep linguistic analysis for huge volumes of data means that it is increasingly important that the time taken to parse such data is minimized. In the XLE parsing model which is a hand-crafted, unification-based parsing system, most of the time is spent on unification, searching for valid f-structures (dependency attribute-value matrices) within the space of the many valid c-structures (phrase structure trees). We carried out an experiment to determine whether pruning the search space at an earlier stage of the parsing process results in an improvement in the overall time taken to parse, while maintaining the quality of the f-structures produced. We retrained a state-of-the-art probabilistic parser and used it to pre-bracket input to the XLE, constraining the valid c-structure space for each sentence. We evaluated against the PARC 700 Dependency Bank and show that it is possible to decrease the time taken to parse by $\sim$18% while maintaining accuracy.

1 Introduction

When deep linguistic analysis of massive data is required (e.g. processing Wikipedia), it is crucial that the parsing time be minimized. The XLE English parsing system is a large-scale, hand-crafted, deep, unification-based system that processes raw text and produces both constituent-structures (phrase structure trees) and feature-structures (dependency attribute-value matrices). A typical breakdown of parsing time of XLE components is Morphology (1.6%), Chart (5.8%) and Unifier (92.6%).

The unification process is the bottleneck in the XLE parsing system. The grammar generates many valid c-structure trees for a particular sentence: the Unifier then processes all of these trees (as packed structures), and a log-linear disambiguation module can choose the most probable f-structure from the resulting valid f-structures. For example, the sentence "Growth is slower." has 84 valid c-structure trees according to the current English grammar;[1] however once the Unifier has processed all of these trees (in a packed form), only one c-structure and f-structure pair is valid (see Figure 1). In this instance, the log-linear disambiguation does not need to choose the most probable result.

The research question we pose is whether the search space can be pruned earlier before unification takes place. Bangalore and Joshi (1999), Clark and Curran (2004) and Matsuzaki et al. (2007) show that by using a super tagger before (CCG and HPSG) parsing, the space required for discriminitive training is drastically reduced. Supertagging is not widely used within the LFG framework, although there has been some work on using hypertags (Kinyon, 2000). Ninomiya et al. (2006) propose a method for faster HPSG parsing while maintaining accuracy by only using the probabilities of lexical entry selections (i.e. the supertags) in their discriminitive model. In the work presented here, we con-

[1]For example, *is* can be a copula, a progressive auxiliary or a passive auxiliary, while *slower* can either be an adjective or an adverb.

centrate on reducing the number of c-structure trees that the Unifier has to process, ideally to one tree. The hope was that this would speed up the parsing process, but how would it affect the quality of the f-structures? This is similar to the approach taken by Cahill et al. (2005) who do not use a hand-crafted complete unification system (rather an automatically acquired probabilistic approximation). They parse raw text into LFG f-structures by first parsing with a probabilistic CFG parser to choose the most probable c-structure. This is then passed to an automatic f-structure annotation algorithm which deterministically generates one f-structure for that tree.

The most compact way of doing this would be to integrate a statistical component to the parser that could rank the c-structure trees and only pass the most likely forward to the unification process. However, this would require a large rewrite of the system. So, we first wanted to investigate a "cheaper" alternative to determine the viability of the pruning strategy; this is the experiment reported in this paper. This is implemented by stipulating constituent boundaries in the input string, so that any c-structure that is incompatible with these constraints is invalid and will not be processed by the Unifier. This was done to some extent in Riezler et al. (2002) to automatically generate training data for the log-linear disambiguation component of XLE. Previous work obtained the constituent constraints (i.e. brackets) from the gold-standard trees in the Penn-II Treebank. However, to parse novel text, gold-standard trees are unavailable.

We used a state-of-the-art probabilistic parser to provide the bracketing constraints to XLE. These parsers are accurate (achieving accuracy of over 90% on Section 23 WSJ text), fast, and robust. The idea is that pre-parsing of the input text by a fast and accurate parser can prune the c-structure search space, reducing the amount of work done by the Unifier, speed up parsing and maintain the high quality of the f-structures produced.

The structure of this paper is as follows: Section 2 introduces the XLE parsing system. Section 3 describes a baseline experiment and based on the results suggests retraining the Bikel parser to improve results (Section 4). Section 5 describes experiments on the development set, from which we evaluate the most successful system against the PARC 700 test set (Section 6). Finally, Section 7 concludes.

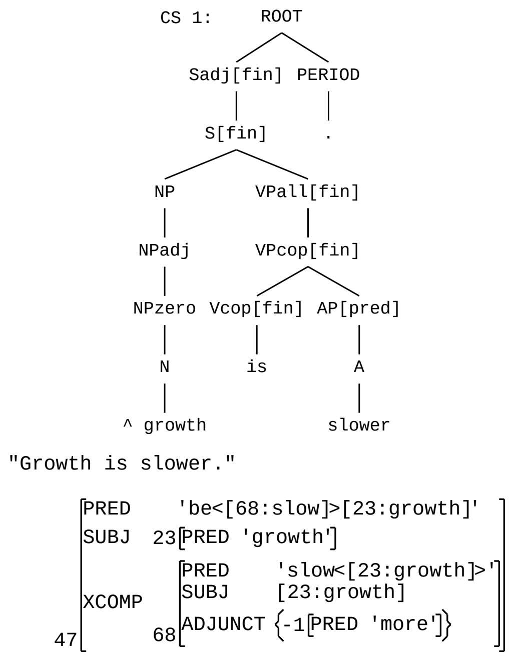

Figure 1: C- and F-Structure for "Growth is slower."

2 Background

In this section we introduce Lexical Functional Grammar, the grammar formalism underlying the XLE, and briefly describe the XLE parsing system.

2.1 Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) is a constraint-based theory of grammar. It (minimally) posits two levels of representation, c(onstituent)-structure and f(unctional)-structure. C-structure is represented by context-free phrase-structure trees, and captures surface grammatical configurations such as word order. The nodes in the trees are annotated with functional equations (attribute-value structure constraints) which are resolved to produce an f-structure. F-structures are recursive attribute-value matrices, representing abstract syntactic functions. F-structures approximate basic predicate-argument-adjunct structures or dependency relations. Figure 1 shows the c- and f-structure for the sentence "Growth is slower.".

```
Parser Output: (S1 (S (NP (NN Growth)) (VP (AUX is) (ADJP (JJR slower))) (.  .)))
      Labeled: \[S1 \[S Growth \[VP is \[ADJP slower\] \].\] \]
      Unlabeled:\[ \[ Growth \[ is \[ slower\] \].\] \]
```

Figure 2: Example of retained brackets from parser output to constrain the XLE parser

2.2 The XLE Parsing System

The XLE parsing system is a deep-grammar-based parsing system. The experiments reported in this paper use the English LFG grammar constructed as part of the ParGram project (Butt et al., 2002). This system incorporates sophisticated ambiguity-management technology so that all possible syntactic analyses of a sentence are computed in an efficient, packed representation (Maxwell and Kaplan, 1993). In accordance with LFG theory, the output includes not only standard context-free phrase-structure trees (c-structures) but also attribute-value matrices (f-structures) that explicitly encode predicate-argument relations and other meaningful properties. The f-structures can be deterministically mapped to dependency triples without any loss of information, using the built-in ordered rewrite system (Crouch et al., 2002). XLE selects the most probable analysis from the potentially large candidate set by means of a stochastic disambiguation component based on a log-linear probability model (Riezler et al., 2002) that works on the packed representations. The underlying parsing system also has built-in robustness mechanisms that allow it to parse strings that are outside the scope of the grammar as a list of fewest well-formed "fragments". Furthermore, performance parameters that bound parsing and disambiguation can be tuned for efficient but accurate operation. These parameters include at which point to timeout and return an error, the amount of stack memory to allocate, the number of new edges to add to the chart and at which point to start skimming (a process that guarantees XLE will finish processing a sentence in polynomial time by only carrying out a bounded amount of work on each remaining constituent after a time threshold has passed). For the experiments reported here, we did not fine-tune these parameters due to time constraints; so default values were arbitrarily set and the same values used for all parsing experiments.

3 Baseline experiments

We carried out a baseline experiment with two state-of-the-art parsers to establish what effect pre-bracketing the input to the XLE system has on the quality and number of the solutions produced. We used the Bikel () multi-threaded, head-driven chart-parsing engine developed at the University of Pennsylvania. The second parser is that described in Charniak and Johnson (2005). This parser uses a discriminative reranker that selects the most probable parse from the 50-best parses returned by a generative parser based on Charniak (2000).

We evaluated against the PARC 700 Dependency Bank (King et al., 2003) which provides gold-standard analyses for 700 sentences chosen at random from Section 23 of the Penn-II Treebank. The Dependency Bank was bootstrapped by parsing the 700 sentences with the XLE English grammar, and then manually correcting the output. The data is divided into two sets, a 140-sentence development set and a test set of 560 sentences (Kaplan et al., 2004).

We took the raw strings from the 140-sentence development set and parsed them with each of the state-of-the-art probabilistic parsers. As an upper bound for the baseline experiment, we use the brackets in the original Penn-II treebank trees for the 140 development set.

We then used the brackets from each parser output (or original treebank trees) to constrain the XLE parser. If the input to the XLE parser is bracketed, the parser will only generate c-structures that respect these brackets (i.e., only c-structures with brackets that are compatible with the input brackets are considered during the unification stage). Figure 2 gives an example of retained brackets from the parser output. We do not retain brackets around PRN (parenthetical phrase) or NP nodes as their structure often differed too much from XLE analyses of the same phrases. We passed pre-bracketed strings to the XLE and evaluated the output f-structures in terms of dependency triples against the 140-sentence subset of

	Non-Fragment		Fragment	
	Penn-XLE	Penn-XLE	Penn-XLE	Penn-XLE
	(lab.)	(unlab.)	(lab.)	(unlab.)
Total XLE parses (/140)	0	89	140	140
F-Score of subset	0	84.11	53.92	74.87
Overall F-Score	0	58.91	53.92	74.87

Table 1: Upper-bound results for original Penn-II trees

	Non-Fragment			Fragment		
	XLE	Bikel-XLE (lab.)	Bikel-XLE (unlab.)	XLE	Bikel-XLE (lab.)	Bikel-XLE (unlab.)
Total XLE Parses (/140)	119	0	84	135	140	140
F-Score of Subset	81.57	0	84.23	78.72	54.37	73.71
Overall F-Score	72.01	0	55.06	76.13	54.37	*73.71
	XLE	CJ-XLE (lab.)	CJ-XLE (unlab.)	XLE	CJ-XLE (lab.)	CJ-XLE (unlab.)
Total XLE Parses (/140)	119	0	86	135	139	139
F-Score of Subset	81.57	0	86.57	78.72	53.96	75.64
Overall F-Score	72.01	0	58.04	76.13	53.48	*74.98

Table 2: Bikel (2002) and Charniak and Johnson (2005) out-of-the-box baseline results

the PARC 700 Dependency Bank.

The results of the baseline experiments are given in Tables 1 and 2. Table 1 gives the upper bound results if we use the gold standard Penn treebank to bracket the input to XLE. Table 2 compares the XLE (fragment and non-fragment) grammar to the system where the input is pre-parsed by each parser. XLE fragment grammars provide a back-off when parsing fails: the grammar is relaxed and the parser builds a fragment parse of the well-formed chunks. We compare the parsers in terms of total number of parses (out of 140) and the f-score of the subset of sentences successfully parsed. We also combine these scores to give an overall f-score, where the system scores 0 for each sentence it could not parse. When testing for statistical significance between systems, we compare the overall f-score values. Figures marked with an asterisk are not statistically significantly different at the 95% level.[2]

The results show that using unlabeled brackets achieves reasonable f-scores with the non-fragment grammar. Using the labeled bracketing from the output of both parsers causes XLE to always fail when parsing. This is because the labels in the output of parsers trained on the Penn-II treebank differ considerably from the labels on c-structure trees produced by XLE. Interestingly, the f-scores for both the CJ-XLE and Bikel-XLE systems are very similar to the upper bounds. The gold standard upper bound is not as high as expected because the Penn trees used to produce the gold bracketed input are not always compatible with the XLE-style trees. As a simple example, the tree in Figure 1 differs from the parse tree for the same sentence in the Penn Treebank (Figure 3). The most obvious difference is the labels on the nodes. However, even in this small example, there are structural differences, e.g. the position of the period. In general, the larger the tree, the greater the difference in both labeling and structure between the Penn trees and the XLE-style trees. Therefore, the next step was to retrain a parser to produce trees with structures the same as XLE-style trees and with XLE English grammar labels on the nodes. For this experiment we use the Bikel () parser, as it is more suited to being retrained on a new treebank annotation scheme.

4 Retraining the Bikel parser

We retrained the Bikel parser so that it produces trees like those outputted by the XLE parsing system (e.g. Figure 1). To do this, we first created a training corpus, and then modified the parser to deal with this new data.

Since there is no manually-created treebank of

[2]We use the approximate randomization test (Noreen, 1989) to test for significance.

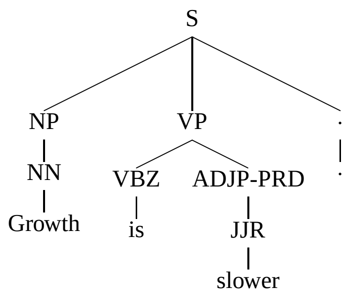

Figure 3: Penn Treebank tree for "Growth is slower."

XLE-style trees, we created one automatically from sections 02-21 of the Penn-II Treebank. We took the raw strings from those sections and marked up NP and SBAR constituents using the brackets from the gold standard Penn treebank. The NP constituents are labeled, and the SBAR unlabeled (i.e. the SBAR constituents are forced to exist in the XLE parse, but the label on them is not constrained to be SBAR). We also tagged verbs, adjectives and nouns, based on the gold standard POS tags.

We parsed the 39,832 marked-up sentences in the standard training corpus and used the XLE disambiguation module to choose the most probable c- and f-structure pair for each sentence. Ideally we would have had an expert choose these. We automatically extracted the c-structure trees produced by the XLE and performed some automatic post-processing.[3] This resulted in an automatically created training corpus of 27,873 XLE-style trees. The 11,959 missing trees were mainly due to the XLE parses not being compatible with the bracketed input, but sometimes due to time and memory constraints.

Using the automatically-created training corpus of XLE-style trees, we retrained the Bikel parser on this data. This required adding a new language module ("XLE-English") to the Bikel parser, and regenerating head-finding rules for the XLE-style trees.

5 Experiments

Once we had a retrained version of the Bikel parser that parses novel text into XLE-style trees, we carried out a number of experiments on our development set in order to establish the optimum settings for the evaluation against the PARC 700 test set.

[3]The postprocessing included removing morphological information and the brackets from the original markup.

	All Sentences	
	XLE	Bikel-XLE
	Non-fragment grammar Labeled brackets	
Total Parsing Time	964	336
Total XLE Parses (/140)	119	77
F-Score of Subset	81.57	86.11
Overall F-Score	72.01	52.84
	Non-fragment grammar Unlabeled brackets	
Total Parsing Time	964	380
Total XLE Parses (/140)	119	89
F-Score of Subset	81.57	85.62
Overall F-Score	72.01	59.34
	Fragment grammar Labeled brackets	
Total Parsing Time	1143	390
Total XLE Parses (/140)	135	140
F-Score of Subset	78.72	71.86
Overall F-Score	76.13	71.86
	Fragment grammar Unlabeled brackets	
Total Parsing Time	1143	423
Total XLE Parses (/140)	135	140
F-Score of Subset	78.72	74.51
Overall F-Score	76.13	*74.51

Table 3: Bikel-XLE Initial Experiments

5.1 Pre-bracketing

We automatically pre-processed the raw strings from the 140-sentence development set. This made systematic changes to the tokens so that the retrained Bikel parser can parse them. The changes included removing quotes, converting *a* and *an* to *_a*, converting *n't* to *_not*, etc. We parsed the pre-processed strings with the new Bikel parser.

We carried out four initial experiments, experimenting with both labeled and unlabeled brackets and XLE fragment and non-fragment grammars. Table 3 gives the results for these experiments. We compare the parsers in terms of time, total number of parses (out of 140), the f-score of the subset of sentences successfully parsed and the overall f-score if the system achieves a score of 0 for all sentences it does not parse. The time taken for the Bikel-XLE system includes the time taken for the Bikel parser to parse the sentences, as well as the time taken for XLE to process the bracketed input.

Table 3 shows that using the non-fragment grammar, the Bikel-XLE system performs better on the subset of sentences parsed than XLE system alone, though the results are not statistically significantly better overall, since the coverage is much lower. The number of bracketed sentences that can be parsed by XLE increases if the brackets are unlabeled. The table also shows that the XLE system performs much better than Bikel-XLE when using the fragment grammars. Although the Bikel-XLE system is quite a bit faster, there is a drop in f-score; however this is not statistically significant when the brackets are unlabeled.

5.2 Pre-tagging

We performed some error analysis on the output of the Bikel-XLE system and noticed that a considerable number of errors were due to mis-tagging. So, we pre-tagged the input to the Bikel parser using the MXPOST tagger (Ratnaparkhi, 1996). The results for the non-fragment grammars are presented in Table 4. Pre-tagging with MXPOST, however, does not result in a statistically significantly higher result than parsing untagged input, although more sentences can be parsed by both systems. Pre-tagging also adds an extra time overhead cost.

		No pretags	MXPOST tags
	XLE	Bikel-XLE	Bikel-XLE
	Unlabeled		
Total Parsing Time	964	380	493
# XLE Parses (/140)	119	89	92
F-Score of Subset	81.57	85.62	84.98
Overall F-Score	72.01	59.34	*61.11
	Labeled		
Total Parsing Time	964	336	407
# XLE Parses (/140)	119	77	80
F-Score of Subset	81.57	86.11	85.87
Overall F-Score	72.01	52.84	*54.91

Table 4: MXPOST pre-tagged, Non-fragment grammar

5.3 Pruning

The Bikel parser can be customized to allow different levels of pruning. The above experiments were carried out using the default level. We carried out experiments with three levels of pruning.[4] The results are given in Table 5 for the experiment with labeled brackets and the non-fragment XLE grammar. More pruning generally results in fewer and lower-quality parses. The biggest gain is with pruning level 1, where the number and quality of bracketed sentences that can be parsed with XLE remains the same as with the default level. This is because Bikel with pruning level 1 does not relax the constraints when parsing fails and does not waste time parsing sentences that cannot be parsed in bracketed form by XLE.

[4]The default level of pruning starts at 3.5, has a maximum of 4 and relaxes constraints when parsing fails. Level 1 pruning is the same as the default except the constraints are never relaxed. Level 2 pruning has a start value of 3.5 and a maximum value of 3.5. Level 3 pruning has a start and maximum value of 3.

	Default	L1	L2	L3
Total Parsing Time	336	137	137	106
# XLE Parses (/140)	77	77	76	75
F-Score of Subset	86.11	86.11	86.04	85.87
Overall F-Score	52.84	*52.84	*52.43	*52.36

Table 5: Pruning with Non-fragment grammar, Labeled brackets, Levels default-3

5.4 Hybrid systems

Although pre-parsing with Bikel results in faster XLE parsing time and high-quality f-structures (when examining only the quality of the sentences that can be parsed by the Bikel-XLE system), the coverage of this system remains poor, therefore the overall f-score remains poor. One solution is to build a hybrid two-pass system. During the first pass all sentences are pre-parsed by Bikel and the bracketed output is parsed by the XLE non-fragment grammar. In the second pass, the sentences that were not parsed during the first pass are parsed with the XLE fragment grammar. We carried out a number of experiments with hybrid systems and the results are given in Table 6.

The results show that again labeled brackets result in a statistically significant increase in f-score, although the time taken is almost the same as the XLE fragment grammar alone. Coverage increases by 1 sentence. Using unlabeled brackets results in 3 additional sentences receiving parses, and parsing time is improved by ∼12%; however the increase in f-score is not statistically significant.

Table 7 gives the results for hybrid systems with pruning using labeled brackets. The more pruning that the Bikel parser does, the faster the system, but the quality of the f-structures begins to deteri-

	XLE (frag)	Bikel-XLE hybrid (labeled)	Bikel-XLE hybrid (unlabeled)
Total Parsing Time	1143	1121	1001
Total XLE Parses (/140)	135	136	138
F-Score of Subset	78.72	79.85	79.51
Overall F-Score	76.13	77.61	*78.28

Table 6: Hybrid systems compared to the XLE fragment grammar alone

	XLE (frag)	Bikel-XLE hybrid (level 1)	Bikel-XLE hybrid (level 2)	Bikel-XLE hybrid (level 3)
Total Parsing Time	1143	918	920	885
Total XLE Parses (/140)	135	136	136	136
F-Score of Subset	78.72	79.85	79.79	79.76
Overall F-Score	76.13	77.61	77.55	77.53

Table 7: Hybrid systems with pruning compared to the XLE fragment grammar alone

orate. The best system is the Bikel-XLE hybrid system with labeled brackets and pruning level 1. This system achieves a statistically significant increase in f-score over the XLE fragment grammar alone, decreases the time taken to parse by almost 20% and increases coverage by 1 sentence. Therefore, we chose this system to perform our final evaluation against the PARC 700 Dependency Bank.

6 Evaluation against the PARC 700

We evaluated the system that performs best on the development set against the 560-sentence test set of the PARC 700 Dependency Bank. The results are given in Table 8. The hybrid system achieves an 18% decrease in parsing time, a slight improvement in coverage of 0.9%, and a 1.12% improvement in overall f-structure quality.

	XLE (frag)	Bikel-XLE hybrid (labeled, prune 1)
Total Parsing Time	4967	4077
Total XLE Parses (/560)	537	542
F-Score of Subset	80.13	80.63
Overall F-Score	77.04	78.16

Table 8: PARC 700 evaluation of the Hybrid system compared to the XLE fragment grammar alone

7 Conclusions

We successfully used a state-of-the-art probabilistic parser in combination with a hand-crafted system to improve parsing time while maintaining the quality of the output produced. Our hybrid system consists of two phases. During phase one, pre-processed, tokenized text is parsed with a retrained Bikel parser. We use the labeled brackets in the output to constrain the c-structures generated by the XLE parsing system. In the second phase, we use the XLE fragment grammar to parse any remaining sentences that have not received a parse in the first phase.

Given the slight increase in overall f-score performance, the speed up in parsing time (∼18%) can justify more complicated processing architecture for some applications.[5] The main disadvantage of the current system is that the input to the Bikel parser needs to be tokenized, whereas XLE processes raw text. One solution to this is to use a state-of-the-art probabilistic parser that accepts untokenized input (such as Charniak and Johnson, 2005) and retrain it as described in Section 4.

Kaplan et al. (2004) compared time and accuracy of a version of the Collins parser tuned to maximize speed and accuracy to an earlier version of the XLE parser. Although the XLE parser was more accurate, the parsing time was a factor of 1.49 slower (time converting Collins trees to dependencies was not counted in the parse time; time to produce f-structures from c-structures was counted in the XLE parse time). The hybrid system here narrows the speed gap while maintaining greater accuracy.

The original hope behind using the brackets to constrain the XLE c-structure generation was that

[5]For example, in massive data applications, if the parsing task takes 30 days, reducing this by 18% saves more than 5 days.

the brackets would force the XLE to choose only one tree. However, the brackets were sometimes ambiguous, and sometimes more than one valid tree was found. In the final evaluation against the PARC 700 test set, the average number of optimal solutions was 4.05; so the log-linear disambiguation module still had to chose the most probable f-structure. However, this is considerably less to choose from than the average of 341 optimal solutions produced by the XLE fragment grammar for the same sentences when unbracketed.

Based on the results of this experiment we have integrated a statistical component into the XLE parser itself. With this architecture the packed c-structure trees are pruned before unification without needing to preprocess the input text. The XLE c-structure pruning results in a $\sim$30% reduction in parse time on the Wikipedia with little loss in precision. We hope to report on this in the near future.

Acknowledgments

The research in this paper was partly funded by Science Foundation Ireland grant 04/BR/CS0370.

References

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

Dan Bikel. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *Proceedings of HLT, YEAR = 2002, pages = 24–27, address = San Diego, CA,*.

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. 2002. The Parallel Grammar Project. In *Proceedings of Workshop on Grammar Engineering and Evaluation*, pages 1–7, Taiwan.

Aoife Cahill, Martin Forst, Michael Burke, Mairead McCarthy, Ruth O'Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. 2005. Treebank-based acquisition of multilingual unification grammar resources. *Journal of Research on Language and Computation*, pages 247–279.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL*, pages 173–180, Ann Arbor, Michigan.

Eugene Charniak. 2000. A maximum entropy inspired parser. In *Proceedings of NAACL*, pages 132–139, Seattle, WA.

Stephen Clark and James R. Curran. 2004. The Importance of Supertagging for Wide-Coverage CCG Parsing . In *Proceedings of COLING*, pages 282–288, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Richard Crouch, Ron Kaplan, Tracy Holloway King, and Stefan Riezler. 2002. A comparison of evaluation metrics for a broad coverage parser. In *Proceedings of the LREC Workshop: Beyond PARSEVAL*, pages 67–74, Las Palmas, Canary Islands, Spain.

Ron Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar, a Formal System for Grammatical Representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Ron Kaplan, Stefan Riezler, Tracy Holloway King, John T. Maxwell, Alexander Vasserman, and Richard Crouch. 2004. Speed and Accuracy in Shallow and Deep Stochastic Parsing. In *Proceedings of HLT-NAACL*, pages 97–104, Boston, MA.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ron Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of LINC*, pages 1–8, Budapest, Hungary.

Alexandra Kinyon. 2000. Hypertags. In *Proceedings of COLING*, pages 446–452, Saarbrücken.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG Parsing with Supertagging and CFG-filtering. In *Proceedings of IJCAI*, pages 1671–1676, India.

John T. Maxwell and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, **19**(4):571–590.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely Lexicalized Models for Accurate and Fast HPSG Parsing. In *Proceedings of EMNLP*, pages 155–163, Australia.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses: An Introduction*. Wiley, New York.

Adwait Ratnaparkhi. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of EMNLP*, pages 133–142, Philadelphia, PA.

Stefan Riezler, Tracy King, Ronald Kaplan, Richard Crouch, John T. Maxwell, and Mark Johnson. 2002. Parsing the Wall Street Journal using a Lexical-Functional Grammar and Discriminative Estimation Techniques. In *Proceedings of ACL*, pages 271–278, Philadelphia, PA.