Speeding up LFG Parsing Using C-Structure Pruning

Aoife Cahill‡ John T. Maxwell III† Paul Meurer§ Christian Rohrer‡ Victoria Rosén¶

‡IMS, University of Stuttgart, Germany, {cahillae, rohrer}@ims.uni-stuttgart.de
†Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, maxwell@parc.com
§Unifob Aksis, Bergen, Norway, paul.meurer@aksis.uib.no
¶Unifob Aksis and University of Bergen, Norway, victoria@uib.no

Abstract

In this paper we present a method for greatly reducing parse times in LFG parsing, while at the same time maintaining parse accuracy. We evaluate the methodology on data from English, German and Norwegian and show that the same patterns hold across languages. We achieve a speedup of 67% on the English data and 49% on the German data. On a small amount of data for Norwegian, we achieve a speedup of 40%, although with more training data we expect this figure to increase.

1 Introduction

Efficient parsing of large amounts of natural language is extremely important for any real-world application. The XLE 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 with the English grammar is Morphology (1.6%), Chart (5.8%) and Unifier (92.6%). It is clear that the major bottleneck in processing is in unification. Cahill et al. (2007) carried out a preliminary experiment to test the theory that if fewer c-structures were passed to the unifier, overall parsing times would improve, while the accuracy of parsing would remain stable. Their experiments used state-of-the-art probabilistic treebank-based parsers to automatically mark certain constituents on the input sentences, limiting the number of c-structures the XLE parsing system would build. They achieved an 18% speedup in parse times, while maintaining the accuracy of the output f-structures. The experiments presented in Cahill et al. (2007) used the XLE system as a black box and did not make any changes to it. However, the results were encouraging enough for a c-structure pruning mechanism to be fully integrated into the XLE system.

The paper is structured as follows: we present the pruning model that has been integrated into the XLE system (Section 2), and how it can be applied successfully to more than one language. We present experiments for English (Section 3), German (Section 4) and Norwegian (Section 5) showing that for both German and English, a significant improvement in speed is achieved, while the quality of the f-structures remains stable. For Norwegian a speedup is also achieved, but more training data is required to sustain the accuracy of the f-structures. In Section 7 we present an error analysis on the German data. We then relate the work presented in this paper to similar efficient parsing strategies (Section 8) before concluding in Section 9.

2 XLE and the C-Structure Pruning Mechanism

The XLE system is designed to deal with large amounts of data in a robust manner. There are several mechanisms which facilitate this, including fragmenting and skimming. Fragmenting is called when the grammar is unable to provide a complete parse for the input sentence, and a fragment analysis of largest possible chunks is built. Skimming is called when too much time or memory has been used by XLE. Any constituents that have not been fully processed are “skimmed”, which means that the amount of work carried out in processing the constituent is limited. This guarantees that XLE will finish processing the sentence in polynomial time.

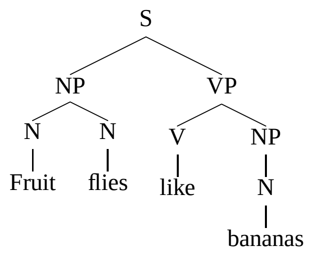
XLE uses a chart-based mechanism for building parses, and has been complemented with a c-structure pruning mechanism to speed up parsing time. During pruning, subtrees at a particular cell in the chart are pruned if their probabilities are not higher than a certain threshold. The chart pruner uses a simple stochastic CFG model. The probability of a tree is the product of the probabilities of each of the rules used to form the tree, including the rules that lead to lexical items (such as N → dog). The probability of a rule is basically the number of times that that particular form of the rule occurs in the training data divided by the number of times the rule’s category occurs in the training data, plus a smoothing term. This is similar to the pruning described in Charniak and Johnson (2005) where edges in a coarse-grained parse forest are pruned to allow full evaluation with fine-grained categories.

The pruner prunes at the level of individual constituents in the chart. It calculates the probabilities of each of the subtrees of a constituent and compares them. The probability of each subtree is compared with the best subtree probability for that constituent. If a subtree’s probability is lower than the best probability by a given factor, then the subtree is pruned. In practice, the threshold is the natural logarithm of the factor used. So a value of 5 means that a subtree will be pruned if its probability is about a factor of 150 less than the best probability.

If two different subtrees have different numbers of morphemes under them, then the probability model is biased towards the subtree that has fewer morphemes (since there are fewer probabilities multiplied together). XLE counteracts this by normalizing the probabilities based on the difference in length.

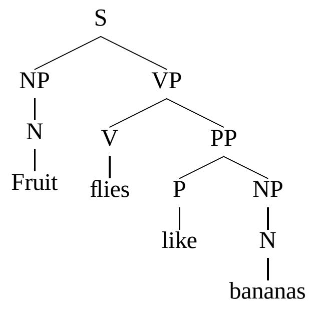
To illustrate how this works, we give the following example. The string *Fruit flies like bananas* has two different analyses. Figures 1 and 2 give their analyses along with hypothetical probabilities for each rule.

These two analyses come together at the S constituent that spans the whole sentence. The probability of the first analysis is 8.4375E-14. The probability of the second analysis is 4.21875E-12. This means that the probability of the second analysis is 50 times higher than the probability of the first analysis. If the threshold is less than the natural logarithm of 50 (about 3.9), then the subtree of the first analysis will be pruned from the S constituent.



S	→	NP VP	0.5000
NP	→	N N	0.1500
N	→	Fruit	0.0010
N	→	flies	0.0015
VP	→	V NP	0.2000
V	→	like	0.0050
NP	→	N	0.5000
N	→	bananas	0.0015
			8.4375E-14

Figure 1: Analysis (1) for the string *Fruit flies like bananas* with hypothetical probabilities



S	→	NP VP	0.5000
NP	→	N	0.5000
N	→	Fruit	0.0010
V	→	flies	0.0025
VP	→	V PP	0.1000
P	→	like	0.0500
PP	→	P NP	0.9000
NP	→	bananas	0.0015
			4.21875E-12

Figure 2: Analysis (2) for the string *Fruit flies like bananas* with hypothetical probabilities

3 Experiments on English

We carried out a number of parsing experiments to test the effect of c-structure pruning, both in terms of time and accuracy. We trained the c-structure pruning algorithm on the standard sections of Penn Treebank Wall Street Journal Text (Marcus et al., 1994). The training data consists of the original WSJ strings, marked up with some of the Penn

Treebank constituent information. We marked up NPs and SBARs as well as adjective and verbal POS categories. This is meant to guide the training process, so that it does learn from parses that are not compatible with the original treebank analysis. We evaluated against the PARC 700 Dependency Bank (King et al., 2003), splitting it into 140 sentences as development data and the remaining unseen 560 for final testing (as in Kaplan et al. (2004)). We experimented with different values of the pruning cutoff on the development set; the results are given in Table 1.

The results show that the lower the cutoff value, the quicker the sentences can be parsed. Using a cutoff of 4, the development sentences can be parsed in 100 CPU seconds, while with a cutoff of 10, the same experiment takes 182 seconds. With no cutoff, the experiment takes 288 CPU seconds. However, this increase in speed comes at a price. The number of fragment parses increases, i.e. there are more sentences that fail to be analyzed with a complete spanning parse. With no pruning, the number of fragment parses is 23, while with the most aggressive pruning factor of 4, there are 39 fragment parses. There are also many more skimmed sentences with no c-structure pruning, which impacts negatively on the results. The oracle f-score with no pruning is 83.07, but with pruning (at all thresholds) the oracle f-score is higher. This is due to less skimming when pruning is activated, since the more subtrees that are pruned, the less likely the XLE system is to run over the time or memory limits needed to trigger skimming.

Having established that a cutoff of 5 performs best on the development data, we carried out the final evaluation on the 560-sentence test set using this cutoff. The results are given in Table 2. There is a 67% speedup in parsing the 560 sentences, and the most probable f-score increases significantly from 79.93 to 82.83. The oracle f-score also increases, while there is a decrease in the random f-score. This shows that we are throwing away good solutions during pruning, but that overall the results improve. Part of this again is due to the fact that with no pruning, skimming is triggered much more often. With a pruning factor of 5, there are no skimmed sentences. There is also one sentence that timed out with no pruning, which also lowers the most probable and oracle f-scores.

Pruning Level	None	5
Total Time	1204	392
Most Probable F-Score	79.93	82.83
Oracle F-Score	84.75	87.79
Random F-Score	75.47	74.31
# Fragment Parses	96	91
# Time Outs	1	0
# Skimmed Sents	33	0

Table 2: Results of c-structure pruning experiments on English test data

4 Experiments on German

We carried out a similar set of experiments on German data to test whether the methodology described above ported to a language other than English. In the case of German, the typical time of XLE components is: Morphology (22.5%), Chart (3.5%) and Unifier (74%). As training data we used the TIGER corpus (Brants et al., 2002). Setting aside 2000 sentences for development and testing, we used the remaining 48,474 sentences as training data. In order to create the partially bracketed input required for training, we converted the original TIGER graphs into Penn-style trees with empty nodes and retained bracketed constituents of the type NP, S, PN and AP. The training data was parsed by the German ParGram LFG (Rohrer and Forst, 2006). This resulted in 25,677 full parses, 21,279 fragmented parses and 1,518 parse failures.[1] There are 52,959 features in the final pruning model.

To establish the optimal pruning settings for German, we split the 2,000 saved sentences into 371 development sentences and 1495 test sentences for final evaluation. We evaluated against the TiGer Dependency Bank (Forst et al., 2004) (TiGerDB), a dependency-based gold standard for German parsers that encodes grammatical relations similar to, though more fine-grained than, the ones in the TIGER Treebank as well as morphosyntactic features. We experimented with the same pruning levels as in the English experiments. The results are given in Table 3.

The results on the development set show a similar trend to the English results. A cutoff of 4 results in the fastest system, however at the expense

[1]The reason there are more fragment parses than, for example, the results reported in Rohrer and Forst (2006) is that the bracketed input constrains the parser to only return parses compatible with the bracketed input. If there is no solution compatible with the brackets, then a fragment parse is returned.

Pruning Level	None	4	5	6	7	8	9	10
Oracle F-Score	83.07	84.50	85.47	**85.75**	85.57	85.57	85.02	84.10
Time (CPU seconds)	288	**100**	109	123	132	151	156	182
# Time Outs	0	0		0	0	0	0	0
# Fragments	**23**	39	36	31	29	27	27	24
# Skimmed Sents	8	0	0	1	1	1	1	3

Table 1: Results of c-structure pruning experiments on English development data

Pruning Level	None	4	5	6	7	8	9	10
Oracle F-Score	83.69	83.45	**84.02**	82.86	82.82	82.95	83.03	82.81
Time (CPU seconds)	1313	**331**	465	871	962	1151	1168	1163
# Time Outs	6	**0**	**0**	5	5	5	5	6
# Fragments	**65**	104	93	81	74	73	73	68

Table 3: Results of c-structure pruning experiments on German development data

Pruning Level	None	5
Total Time	3300	1655
Most Probable F-Score	82.63	82.73
Oracle F-Score	84.96	84.79
Random F-Score	73.58	73.72
# Fragment Parses	324	381
# Time Outs	2	2

Table 4: Results of c-structure pruning experiments on German test data

of accuracy. A cutoff of 5 seems to provide the best tradeoff between time and accuracy. Again, most of the gain in oracle f-score is due to fewer timeouts, rather than improved f-structures. In the German development set, a cutoff of 5 leads to a speedup of over 64% and a small increase in oracle f-score of 0.33 points. Therefore, for the final evaluation on the unseen test-set, we choose a cutoff of 5. The results are given in Table 4. We achieve a speedup of 49% and a non-significant increase in most probable f-score of 0.094. The time spent by the system on morphology is much higher for German than for English. If we only take the unification stage of the process into account, the German experiments show a speedup of 65.5%.

5 Experiments on Norwegian

As there is no treebank currently available for Norwegian, we were unable to train the c-structure pruning mechanism for Norwegian in the same way as was done for English and German. There is, however, some LFG-parsed data that has been completely disambiguated using the techniques described in Rosén et al. (2006). In total there are 937 sentences from various text genres including Norwegian hiking guides, Sophie's World and the Norwegian Wikipedia. We also use this disambiguated data as a gold standard for evaluation. The typical time of XLE components with the Norwegian grammar is: Morphology (1.6%), Chart (11.2%) and Unifier (87.2%).

From the disambiguated text, we can automatically extract partially bracketed sentences as input to the c-structure pruning training method. We can also extract sentences for training that are partially disambiguated, but these cannot be used as part of the test data. To do this, we extract the bracketed string for each solution. If all the solutions produce the same bracketed string, then this is added to the training data. This results in an average of 4556 features. As the data set is small, we do not split it into development, training and test sections as was done for English and German. Instead we carry out a 10-fold cross validation over the entire set. The results for each pruning level are given in Table 5.

The results in Table 5 show that the pattern that held for English and German does not quite hold for Norwegian. While, as expected, the time taken to parse the test set is greatly reduced when using c-structure pruning, there is also a negative impact on the quality of the f-structures. One reason for this is that there are now sentences that could previously be parsed, and that now no longer can be parsed, even with a fragment grammar.[2] With c-structure pruning, the number of fragment parses increases for all thresholds, apart from 10. It is also difficult to compare the Norwegian experiment to the English and German, since the gold standard is constrained to only consist of sentences that can be parsed by the grammar. Theoretically the oracle f-score for the experiment with no prun-

[2]With an extended fragment grammar, this would not happen.

Pruning Level	None	4	5	6	7	8	9	10
Oracle F-Score	**98.76**	94.45	95.60	96.40	96.90	97.52	98.00	98.33
Time (CPU seconds)	218.8	**106.2**	107.4	109.3	112	116.2	124	130.7
# Time Outs	0	0	0	0	0	0	0	0
# Parse Failures	**0.2**	5.7	3.9	2	3.2	4.2	4.6	4.2
# Fragments	1.3	7.7	6.5	4.7	2.8	1.8	1.5	**1.2**

Table 5: Results of c-structure pruning 10-fold cross validation experiments on Norwegian data

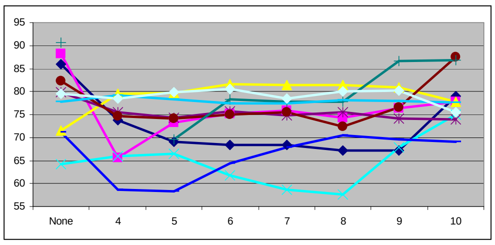


Figure 3: The lower-bound results for each of the 10 cross validation runs across the thresholds

ing should be 100. The slight drop is due to a slightly different morphological analyzer used in the final experiments that treats compound nouns differently. A threshold of 10 gives the best results, with a speedup of 40% and a drop in f-score of 0.43 points. It is difficult to choose the "best" threshold, as the amount of training data is probably not enough to get an accurate picture of the data. For example, Figure 3 shows the lower-bound results for each of the 10 runs. It is difficult to see a clear pattern for all the runs, indicating that the amount of training data is probably not enough for a reliable experiment.

6 Size of Training Data Corpus

The size of the Norwegian training corpus is considerably smaller than the training corpora for English or German, so the question remains how much training data we need in order for the c-structure pruning to deliver reliable results. In order to establish a rough estimate for the size of training corpus required, we carried out an experiment on the German TIGER training corpus.

We randomly divided the TIGER training corpus into sets of 500 sentences. We plot the learning curve of the c-structure pruning mechanism in Figure 4, examining the effect of increasing the size of the training corpus on the oracle f-score on the development set of 371 sentences. The curve shows that, for the German data, the highest oracle f-score of 84.98 was achieved with a training corpus of 32,000 sentences. Although the curve fluctuates, the general trend is that the more training data, the better the oracle f-score.[3]

7 Error Analysis

Given that we are removing some subtrees during parsing, it can sometimes happen that the desired analysis gets pruned. We will take German as an example, and look at some of these cases.

7.1 Separable particles vs pronominal adverbs

The word *dagegen* ("against it") can be a separable prefix (VPART) or a pronominal adverb (PADV). The verb *protestieren* ("to protest") does not take *dagegen* as separable prefix. The verb *stimmen* ("to agree") however does. If we parse the sentence in (1) with the verb *protestieren* and activate pruning, we do not get a complete parse. If we parse the same sentence with *stimmen* as in (2) we do get a complete parse. If we replace *dagegen* by *dafür*, which in the current version of the German LFG can only be a pronominal adverb, the sentence in (3) gets a parse. We also notice that if we parse a sentence, as in (4), where *dagegen* occurs in a position where our grammar does not allow separable prefixes to occur, we get a complete parse for the sentence. These examples show that the pruning mechanism has learned to prune the separable prefix reading of words that can be both separable prefixes and pronominal adverbs.

(1)	Sie	protestieren	dagegen.
	they	protest	against-it
	'They protest against it.'		

(2)	Sie	stimmen	dagegen.
	they	vote	against-it
	'They vote against it.'		

[3]Unexpectedly, the curve begins to decline after 32,000 sentences. However, the differences in f-score are not statistically significant (using the approximate randomization test). Running the same experiment with a different random seed results in a similarly shaped graph, but any decline in f-score when training on more data was not statistically significant at the 99% level.

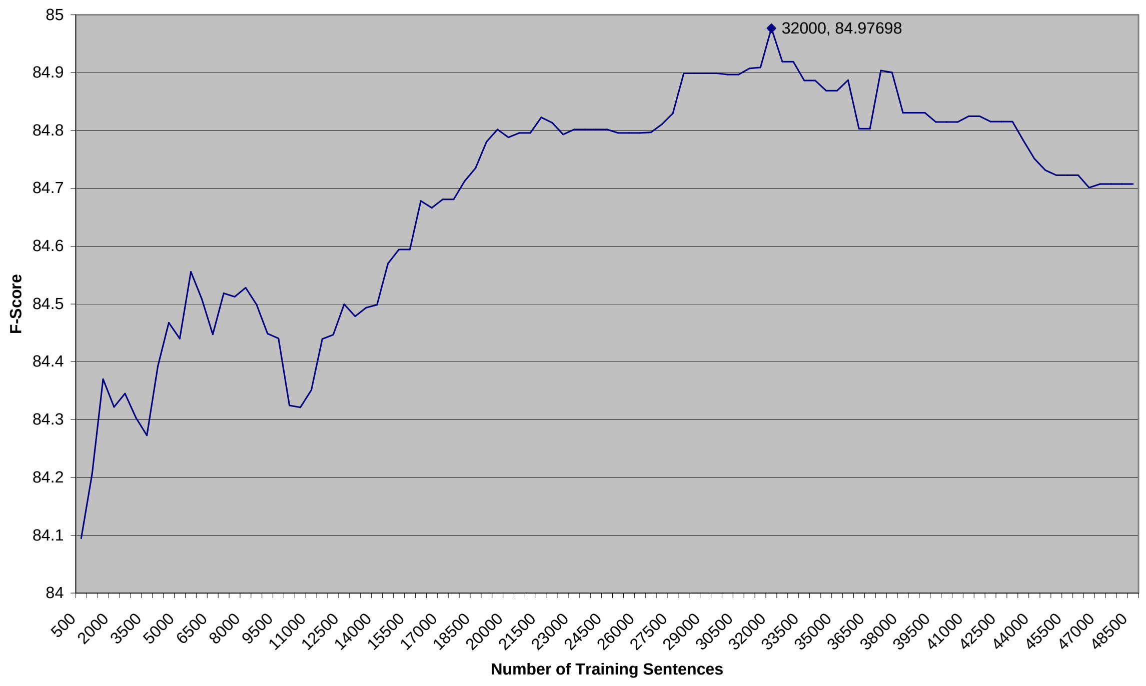


Figure 4: The effect of increasing the size of the training data on the oracle f-score

(3) Er protestiert dafür.
he protests for-it
'He protests in favour of it.'

(4) Dagegen protestiert er.
against-it protests he
'Against it, he protests.'

7.2 Derived nominal vs non-derived nominal

The word *Morden* can be the dative plural of the noun *Mord* ("murder") or the nominalized form of the verb *morden* ("to murder"). With c-structure pruning activated (at level 5), the nominalized reading, as in (6), gets pruned, whereas the dative plural reading is not (5). At pruning level 6, both readings are assigned a full parse. We see similar pruning of nominalized readings as in (7). If we look in more detail at the raw counts for related subtrees gathered from the training data, we see that the common noun reading for *Morden* occurs 156 times, while the nominalized reading only occurs three times. With more training data, the c-structure pruning mechanism could possibly learn when to prune correctly in such cases.

(5) Er redet von Morden.
he speaks of murders
'He speaks of murders.'

(6) Das Morden will nicht enden.
the murdering wants not end
'The murdering does not want to end.'

(7) Das Arbeiten endet.
the working ends
'The operation ends.'

7.3 Personal pronouns which also function as determiners

There are a number of words in German that can function both as personal pronouns and determiners. If we take, for example, the word *ihr*, which can mean "her", "their", "to-her", "you-pl" etc., the reading as a determiner gets pruned as well as some occurrences as a pronoun. In example (8), we get a complete parse for the sentence with the dative pronoun reading of *ihr*. However, in example (9), the determiner reading is pruned and we fail to get a complete parse. In example (10), we also fail to get a complete parse, but in example (11), we do get a complete parse. There is a parameter we can set that sets a confidence value in certain tags. So, for example, we set the confidence value of INFL-F_BASE[det] (the tag given to the determiner reading of personal pronouns) to be 0.5, which says that we are 50% confident that the tag INFL-F_BASE[det] is correct. This results in examples 8, 9 and 11 receiving a complete parse, with the pruning threshold set to 5.

(8) Er gibt es ihr.
he gives it her
'He gives it to her.'

(9) Ihr Auto fährt.
her/their car drives
'Her/Their car drives.

(10) Ihr kommt.
you(pl) come
'You come.'

(11) Er vertraut ihr.
he trusts her
'He trusts her.'

7.4 Coordination of Proper Nouns

Training the German c-structure pruning mechanism on the TIGER treebank resulted in a peculiar phenomenon when parsing coordinated proper nouns. If we parse four coordinated proper nouns with c-structure pruning activated as in (12), we get a complete parse. However, as soon as we add a fifth proper noun as in (13), we get a fragment parse. This is only the case with proper nouns, since the sentence in (14) which coordinates common nouns gets a complete parse. Interestingly, if we coordinate *n* proper nouns plus one common noun, we also get a complete parse. The reason for this is that proper noun coordination is less common than common noun coordination in our training set.

(12) Hans, Fritz, Emil und Maria singen.
'Hans, Fritz, Emil and Maria sing.'

(13) Hans, Fritz, Emil, Walter und Maria singen.
'Hans, Fritz, Emil, Walter and Maria sing.'

(14) Hunde, Katzen, Esel, Pferde und Affen kommen.
'Dogs, cats, donkeys, horses and apes come.'

(15) Hans, Fritz, Emil, Walter, Maria und Kinder singen.
'Hans, Fritz, Emil, Walter, Maria and children sing.'

We ran a further experiment to test what effect adding targeted training data had on c-structure pruning. We automatically extracted a specialized corpus of 31,845 sentences from the Huge German Corpus. This corpus is a collection of 200 million words of newspaper and other text. The sentences we extracted all contained examples of proper noun coordination and had been automatically chunked. Training on this sub-corpus as well as the original TIGER training data did have the desired effect of now parsing example (13) with c-structure pruning activated.

8 Related Work

Ninomiya et al. (2005) investigate beam thresholding based on the local width to improve the speed of a probabilistic HPSG parser. In each cell of a CYK chart, the method keeps only a portion of the edges which have higher figure of merits compared to the other edges in the same cell. In particular, each cell keeps the edges whose figure of merit is greater than α_{max} - δ, where α_{max} is the highest figure of merit among the edges in the chart. The term "beam thresholding" is a little confusing, since a beam search is not necessary – instead, the CYK chart is pruned directly. For this reason, we prefer the term "chart pruning" instead.

Clark and Curran (2007) describe the use of a supertagger with a CCG parser. A supertagger is like a tagger but with subcategorization information included. Chart pruners and supertaggers are conceptually complementary, since chart pruners prune edges with the same span and the same category, whereas supertaggers prune (lexical) edges with the same span and different categories. Ninomiya et al. (2005) showed that combining a chunk parser with beam thresholding produced better results than either technique alone. So adding a supertagger should improve the results described in this paper.

Zhang et al. (2007) describe a technique to selectively unpack an HPSG parse forest to apply maximum entropy features and get the n-best parses. XLE already does something similar when it applies maximum entropy features to get the n-best feature structures after having obtained a packed representation of all of the valid feature structures. The current paper shows that pruning the c-structure chart before doing (packed) unification speeds up the process of getting a packed representation of all the valid feature structures (except the ones that may have been pruned).

9 Conclusions

In this paper we have presented a c-structure pruning mechanism which has been integrated into the XLE LFG parsing system. By pruning the number of c-structures built in the chart, the next stage of processing, the unifier, has considerably less work to do. This results in a speedup of 67% for English, 49% for German and 40% for Norwegian. The amount of training data for Norwegian was much less than that for English or German, therefore further work is required to fully investigate the effect of c-structure pruning. However, the results, even from the small training data, were encouraging and show the same general patterns as English and German. We showed that for the German training data, 32,000 sentences was the optimal number in order to achieve the highest oracle f-score. There remains some work to be done in tuning the parameters for the c-structure pruning, as our error analysis shows. Of course, with statistical methods one can never be guaranteed that the correct parse will be produced; however we can adjust the parameters to account for known problems. We have shown that the c-structure pruning mechanism described is an efficient way of reducing parse times, while maintaining the accuracy of the overall system.

Acknowledgements

The work presented in this paper was supported by the COINS project as part of the linguistic Collaborative Research Centre (SFB 732) at the University of Stuttgart and by the Norwegian Research Council through the LOGON and TREPIL projects.

References

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

Cahill, Aoife, Tracy Holloway King, and John T. Maxwell III. 2007. Pruning the Search Space of a Hand-Crafted Parsing System with a Probabilistic Parser. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 65–72, Prague, Czech Republic, June. Association for Computational Linguistics.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Clark, Stephen and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, 33(4):493–552.

Forst, Martin, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, and Valia Kordoni. 2004. Towards a dependency-based gold standard for German parsers – The TiGer Dependency Bank. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora (LINC '04)*, Geneva, Switzerland.

Kaplan, Ronald M., John T. Maxwell, Tracy H. King, and Richard Crouch. 2004. Integrating Finite-state Technology with Deep LFG Grammars. In *Proceedings of the ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*, Nancy, France.

King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the EACL Workshop on Linguistically Interpreted Corpora (LINC '03)*, Budapest, Hungary.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Ninomiya, Takashi, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Efficacy of Beam Thresholding, Unification Filtering and Hybrid Parsing in Probabilistic HPSG Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 103–114, Vancouver, British Columbia, October. Association for Computational Linguistics.

Rohrer, Christian and Martin Forst. 2006. Improving Coverage and Parsing Quality of a Large-scale LFG for German. In *Proceedings of the Language Resources and Evaluation Conference (LREC-2006)*, Genoa, Italy.

Rosén, Victoria, Paul Meurer, and Koenraad de Smedt. 2006. Towards a Toolkit Linking Treebanking and Grammar Development. In Hajic, Jan and Joakim Nivre, editors, *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 55–66, December.

Zhang, Yi, Stephan Oepen, and John Carroll. 2007. Efficiency in Unification-Based N-Best Parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 48–59, Prague, Czech Republic, June. Association for Computational Linguistics.