Compacting the Penn Treebank Grammar

Alexander Krotov and **Mark Hepple** and **Robert Gaizauskas** and **Yorick Wilks**
Department of Computer Science, Sheffield University
211 Portobello Street, Sheffield S1 4DP, UK
{alexk, hepple, robertg, yorick}@dcs.shef.ac.uk

Abstract

Treebanks, such as the Penn Treebank (PTB), offer a simple approach to obtaining a broad coverage grammar: one can simply read the grammar off the parse trees in the treebank. While such a grammar is easy to obtain, a square-root rate of growth of the rule set with corpus size suggests that the derived grammar is far from complete and that much more treebanked text would be required to obtain a complete grammar, if one exists at some limit. However, we offer an alternative explanation in terms of the underspecification of structures within the treebank. This hypothesis is explored by applying an algorithm to *compact* the derived grammar by eliminating redundant rules – rules whose right hand sides can be parsed by other rules. The size of the resulting compacted grammar, which is significantly less than that of the full treebank grammar, is shown to approach a limit. However, such a compacted grammar does not yield very good performance figures. A version of the compaction algorithm taking rule probabilities into account is proposed, which is argued to be more linguistically motivated. Combined with simple thresholding, this method can be used to give a 58% reduction in grammar size without significant change in parsing performance, and can produce a 69% reduction with some gain in recall, but a loss in precision.

1 Introduction

The Penn Treebank (PTB) (Marcus et al., 1994) has been used for a rather simple approach to deriving large grammars automatically: one where the grammar rules are simply 'read off' the parse trees in the corpus, with each local subtree providing the left and right hand sides of a rule. Charniak (Charniak, 1996) reports precision and recall figures of around 80% for a parser employing such a grammar. In this paper we show that the huge size of such a treebank grammar (see below) can be reduced in size without appreciable loss in performance, and, in fact, an improvement in recall can be achieved.

Our approach can be generalised in terms of Data-Oriented Parsing (DOP) methods (see (Bonnema et al., 1997)) with the tree depth of 1. However, the number of trees produced with a general DOP method is so large that Bonnema (Bonnema et al., 1997) has to resort to restricting the tree depth, using a very domain-specific corpus such as ATIS or OVIS, and parsing very short sentences of average length 4.74 words. Our compaction algorithm can be easily extended for the use within the DOP framework but, because of the huge size of the derived grammar (see below), we chose to use the simplest PCFG framework for our experiments.

We are concerned with the nature of the rule set extracted, and how it can be improved, with regard both to linguistic criteria and processing efficiency. In what follows, we report the worrying observation that the growth of the rule set continues at a square root rate throughout processing of the entire treebank (suggesting, perhaps that the rule set is far from complete). Our results are similar to those reported in (Krotov et al., 1994). [1] We discuss an alternative possible source of this rule growth phenomenon, *partial bracketting*, and suggest that it can be alleviated by *compaction*, where rules that are redundant (in a sense to be defined) are eliminated from the grammar.

Our experiments on compacting a PTB tree-

[1] For the complete investigation of the grammar extracted from the Penn Treebank II see (Gaizauskas, 1995)

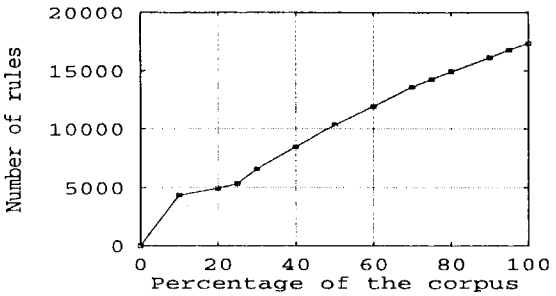


Figure 1: Rule Set Growth for Penn Treebank II

bank grammar resulted in two major findings: one, that the grammar can be compacted to about 7% of its original size, and the rule number growth of the compacted grammar stops at some point. The other is that a 58% reduction can be achieved with no loss in parsing performance, whereas a 69% reduction yields a gain in recall, but a loss in precision.

This, we believe, gives further support to the utility of treebank grammars and to the compaction method. For example, compaction methods can be applied within the DOP framework to reduce the number of trees. Also, by partially lexicalising the rule extraction process (i.e., by using some more frequent words as well as the part-of-speech tags), we may be able to achieve parsing performance similar to the best results in the field obtained in (Collins, 1996).

2 Growth of the Rule Set

One could investigate whether there is a finite grammar that should account for any text within a class of related texts (i.e. a domain oriented sub-grammar of English). If there is, the number of extracted rules will approach a limit as more sentences are processed, i.e. as the rule number approaches the size of such an underlying and finite grammar.

We had hoped that some approach to a limit would be seen using PTB II (Marcus et al., 1994), which larger and more consistent for bracketting than PTB I. As shown in Figure 1, however, the rule number growth continues unabated even after more than 1 million part-of-speech tokens have been processed.

3 Rule Growth and Partial Bracketting

Why should the set of rules continue to grow in this way? Putting aside the possibility that natural languages do not have finite rule sets, we can think of two possible answers. First, it may be that the full "underlying grammar" is much larger than the rule set that has so far been produced, requiring a much larger tree-banked corpus than is now available for its extraction. If this were true, then the outlook would be bleak for achieving near-complete grammars from treebanks, given the resource demands of producing hand-parsed text. However, the radical incompleteness of grammar that this alternative implies seems incompatible with the promising parsing results that Charniak reports (Charniak, 1996).

A second answer is suggested by the presence in the extracted grammar of rules such as (1).[2] This rule is suspicious from a linguistic point of view, and we would expect that the text from which it has been extracted should more properly have been analysed using rules (2,3), i.e. as a coordination of two simpler NPs.

$$NP \rightarrow DT\ NN\ CC\ DT\ NN \quad (1)$$

$$NP \rightarrow NP\ CC\ NP \quad (2)$$

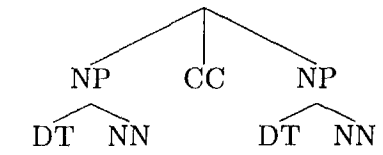
$$NP \rightarrow DT\ NN \quad (3)$$

Our suspicion is that this example reflects a widespread phenomenon of *partial bracketting* within the PTB. Such partial bracketting will arise during the hand-parsing of texts, with (human) parsers adding brackets where they are confident that some string forms a given constituent, but leaving out many brackets where they are less confident of the constituent structure of the text. This will mean that many rules extracted from the corpus will be 'flatter' than they should be, corresponding properly to what should be the result of using several grammar rules, showing only the top node and leaf nodes of some unspecified tree structure (where the 'leaf nodes' here are category symbols, which may be nonterminal). For the example above, a tree structure that should properly have been given as (4), has instead received

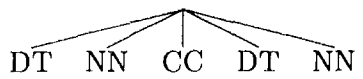
[2]PTB POS tags are used here, i.e. DT for determiner, CC for coordinating conjunction (e.g 'and'), NN for noun

only the partial analysis (5), from the flatter 'partial-structure' rule (1).

i. (4)



ii. (5)



4 Grammar Compaction

The idea of partiality of structure in treebanks and their grammars suggests a route by which treebank grammars may be reduced in size, or *compacted* as we shall call it, by the elimination of partial-structure rules. A rule that may be eliminable as a partial-structure rule is one that can be 'parsed' (in the familiar sense of context-free parsing) using *other* rules of the grammar. For example, the rule (1) can be parsed using the rules (2,3), as the structure (4) demonstrates. Note that, although a partial-structure rule should be parsable using other rules, it does not follow that every rule which is so parsable is a partial-structure rule that should be eliminated. There may be defensible rules which can be parsed. This is a topic to which we will return at the end of the paper (Sec. 6). For most of what follows, however, we take the simpler path of assuming that the parsability of a rule is not only necessary, but also sufficient, for its elimination.

Rules which can be parsed using other rules in the grammar are *redundant* in the sense that eliminating such a rule will *never* have the effect of making a sentence unparsable that could previously be parsed.[3]

The algorithm we use for compacting a grammar is straightforward. A loop is followed whereby each rule R in the grammar is addressed in turn. If R can be parsed using other rules (which have not already been eliminated) then R is deleted (and the grammar *without* R is used for parsing further rules). Otherwise R is kept in the grammar. The rules that remain when all rules have been checked constitute the compacted grammar.

An interesting question is whether the result of compaction is independent of the order in which the rules are addressed. In general, this is not the case, as is shown by the following rules, of which (8) and (9) can each be used to parse the other, so that whichever is addressed first will be eliminated, whilst the other will remain.

$$B \rightarrow C \quad (6)$$

$$C \rightarrow B \quad (7)$$

$$A \rightarrow B\ B \quad (8)$$

$$A \rightarrow C\ C \quad (9)$$

Order-independence can be shown to hold for grammars that contain no *unary* or *epsilon* ('empty') rules, i.e. rules whose righthand sides have one or zero elements. The grammar that we have extracted from PTB II, and which is used in the compaction experiments reported in the next section, is one that excludes such rules. For further discussion, and for the proof of the order independence see (Krotov, 1998). Unary and sister rules were collapsed with the sister nodes, e.g. the structure `(S (NP -NULL-) (VP VB (NP (QP ...))) .)` will produce the following rules: `S -> VP .`, `VP -> VB QP` and `QP -> ...`.[4]

5 Experiments

We conducted a number of compaction experiments: [5] first, the complete grammar was parsed as described in Section 4. Results exceeded our expectations: the set of 17,529 rules reduced to only 1,667 rules, a better than 90% reduction.

To investigate in more detail how the compacted grammar grows, we conducted a third experiment involving a *staged* compaction of the grammar. Firstly, the corpus was split into 10% chunks (by number of files) and the rule sets extracted from each. The staged compaction proceeded as follows: the rule set of the first 10% was compacted, and then the rules for the

[3] Thus, wherever a sentence has a parse P that employs the parsable rule R, it also has a further parse that is just like P except that any use of R is replaced by a more complex substructure, i.e. a parse of R.

[4] See (Gaizauskas, 1995) for discussion.

[5] For these experiments, we used two parsers: Stolcke's BOOGIE (Stolcke, 1995) and Sekine's Apple Pie Parser (Sekine and Grishman, 1995).

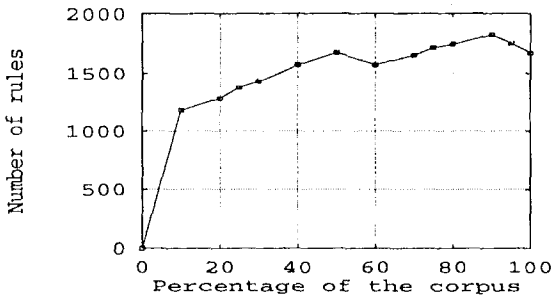


Figure 2: Compacted Grammar Size

next 10% added and the resulting set again compacted, and then the rules for the next 10% added, and so on. Results of this experiment are shown in Figure 2.

At 50% of the corpus processed the compacted grammar size actually exceeds the level it reaches at 100%, and then the overall grammar size starts to go down as well as up. This reflects the fact that new rules are either redundant, or make "old" rules redundant, so that the compacted grammar size seems to approach a limit.

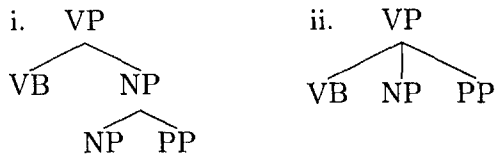
6 Retaining Linguistically Valid Rules

Even though parsable rules are redundant in the sense that has been defined above, it does not follow that they should always be removed. In particular, there are times where the flatter structure allowed by some rule may be more linguistically correct, rather than simple a case of partial bracketting. Consider, for example, the (linguistically plausible) rules (10,11,12). Rules (11) and (12) can be used to parse (10), but it should not be eliminated, as there are cases where the flatter structure it allows is more linguistically correct.

$$VP \rightarrow VB\ NP\ PP \quad (10)$$

$$VP \rightarrow VB\ NP \quad (11)$$

$$NP \rightarrow NP\ PP \quad (12)$$



(13)

We believe that a solution to this problem can be found by exploiting the date provided by the corpus. Frequency of occurrence data for rules which have been collected from the corpus and used to assign probabilities to rules, and hence to the structures they allow, so as to produce a *probabilistic* context-free grammar for the rules. Where a parsable rule is correct rather than merely partially bracketted, we then expect this fact to be reflected in rule and parse probabilities (reflecting the occurrence data of the corpus), which can be used to decide when a rule that *may* be eliminated *should* be eliminated. In particular, a rule should be eliminated only when the more complex structure allowed by other rules is more probable than the simpler structure that the rule itself allows.

We developed a linguistic compaction algorithm employing the ideas just described. However, we cannot present it here due to the space limitations. The preliminary results of our experiments are presented in Table 1. Simple thresholding (removing rules that only occur once) was also to achieve the maximum compaction ratio. For labelled as well as unlabelled evaluation of the resulting parse trees we used the `evalb` software by Satoshi Sekine. See (Krotov, 1998) for the complete presentation of our methodology and results.

As one can see, the fully compacted grammar yields poor recall and precision figures. This can be because collapsing of the rules often produces too much substructure (hence lower precision figures) and also because many longer rules in fact encode valid linguistic information. However, linguistic compaction combined with simple thresholding achieves a 58% reduction without any loss in performance, and 69% reduction even yields higher recall.

7 Conclusions

We see the principal results of our work to be the following:

- the result showing continued square-root growth in the rule set extracted from the PTB II;
- the analysis of the source of this continued growth in terms of *partial bracketting* and the justification this provides for compaction via rule-parsing;
- the result that the compacted rule set *does* approach a limit at some point dur-

	Full	Simply thresholded	Fully compacted	Linguistically compacted	
				Grammar 1	Grammar 2
Labelled evaluation					
Recall	70.55%	70.78%	30.93%	71.55%	70.76%
Precision	77.89%	77.66%	19.18%	72.19%	77.21%
Unlabelled evaluation					
Recall	73.49%	73.71%	43.61%	74.72%	73.67%
Precision	81.44%	80.87%	27.04%	75.39%	80.39%
Grammar size	15,421	7,278	1,122	4,820	6,417
reduction (as % of full)	0%	53%	93%	69%	58%

Table 1: Preliminary results of evaluating the grammar compaction method

ing staged rule extraction and compaction, after a sufficient amount of input has been processed;

- that, though the fully compacted grammar produces lower parsing performance than the extracted grammar, a 58% reduction (without loss) can still be achieved by using linguistic compaction, and 69% reduction yields a gain in recall, but a loss in precision.

The latter result in particular provides further support for the possible future utility of the compaction algorithm. Our method is similar to that used by Shirai (Shirai et al., 1995), but the principal differences are as follows. First, their algorithm does not employ full context-free parsing in determining the redundancy of rules, considering instead only direct composition of the rules (so that only parses of depth 2 are addressed). We proved that the result of compaction is independent of the order in which the rules in the grammar are parsed in those cases involving 'mutual parsability' (discussed in Section 4), but Shirai's algorithm will eliminate both rules so that coverage is lost. Secondly, it is not clear that compaction will work in the same way for English as it did for Japanese.

References

Remko Bonnema, Rens Bod, and Remko Scha. 1997. A DOP model for semantic interpretation. In *Proceedings of European Chapter of the ACL*, pages 159–167.

Eugene Charniak. 1996. Tree-bank grammars. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1031–1036. MIT Press, August.

Michael Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL.*

Robert Gaizauskas. 1995. Investigations into the grammar underlying the Penn Treebank II. Research Memorandum CS-95-25, University of Sheffield.

Alexander Krotov, Robert Gaizauskas, and Yorick Wilks. 1994. Acquiring a stochastic context-free grammar from the Penn Treebank. In *Proceedings of Third Conference on the Cognitive Science of Natural Language Processing*, pages 79–86, Dublin.

Alexander Krotov. 1998. Notes on compacting the Penn Treebank grammar. Technical Memo, Department of Computer Science, University of Sheffield.

M. Marcus, G. Kim, M.A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of ARPA Speech and Natural language workshop.*

Satoshi Sekine and Ralph Grishman. 1995. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings of Fourth International Workshop on Parsing Technologies.*

Kiyoaki Shirai, Takenobu Tokunaga, and Hozumi Tanaka. 1995. Automatic extraction of Japanese grammar from a bracketed corpus. In *Proceedings of Natural Language Processing Pacific Rim Symposium*, Korea, December.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.