Natural Language Engineering **5** (4): 377–394. Printed in the United Kingdom
© 1999 Cambridge University Press

Evaluating two methods for Treebank grammar compaction

ALEXANDER KROTOV, MARK HEPPLE,
ROBERT GAIZAUSKAS and YORICK WILKS
*Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK*
{alexk, hepple, robertg, yorick}@dcs.shef.ac.uk

(*Received 18 February 1998; revised 3 March 2000*)

Abstract

Treebanks, such as the Penn Treebank, provide a basis for the automatic creation of broad coverage grammars. In the simplest case, rules can simply be 'read off' the parse-annotations of the corpus, producing either a simple or probabilistic context-free grammar. Such grammars, however, can be very large, presenting problems for the subsequent computational costs of parsing under the grammar. In this paper, we explore ways by which a treebank grammar can be reduced in size or 'compacted', which involve the use of two kinds of technique: (i) *thresholding* of rules by their number of occurrences; and (ii) a method of *rule-parsing*, which has both probabilistic and non-probabilistic variants. Our results show that by a combined use of these two techniques, a probabilistic context-free grammar can be reduced in size by 62% without any loss in parsing performance, and by 71% to give a gain in recall, but some loss in precision.

1 Introduction

The past decade has seen the rise of corpus-based, 'empirical' methods within computational linguistics. Such methods standardly involve a training phase during which data is collected from a corpus for subsequent use in processing, and in some cases the volume of information extracted during training can be very large. This fact can have serious resource consequences subsequently, i.e. both in terms of the space required to store the data, and also the cost in time for mobilising a large amount of information during processing. In this context, the issue arises of how the collected data might filtered or pruned, or in any way reduced or compressed, as a route to easing the resource problems mentioned. This paper addresses this issue in relation to the area of parsing based on so-called treebank grammars, i.e. grammars derived from parse-annotated corpora or 'treebanks', such as the Penn Treebank (PTB) (Marcus *et al.*, 1993). Several approaches have been used for deriving grammars from treebanks, but in the simplest case, the rules of the grammar can simply be 'read off' the parse trees in the corpus, with each local subtree providing the left and right hand sides of a rule. Such an approach will yield a simple context-free grammar (CFG), or with some additional book-keeping, a probabilistic CFG (PCFG). Even for such simple grammatical models, the size of the grammar collected can be very large, e.g. more than 17 000 rules in our own experiments (Gaizauskas, 1995) on PTB II (the second release of the PTB), so clearly the possibility that the collected grammar can be pruned or compacted to any significant extent is very attractive, provided that any loss in parsing performance is within acceptable bounds. Even more intriguing is the possibility that suitable pruning of the grammar might produce an *increase* in parsing performance, alongside any other computational benefits.

In what follows, we will note some previous work within the treebank parsing area, before going on to describe our own experiments on extracting grammars from the PTB. Here we report the worrying observation that the rate of acquisition of ‘new rules’ continues with very little reduction throughout processing of the entire treebank, suggesting perhaps that the resulting rule set is far from complete. We suggest a possible explanation of this rule growth phenomenon in terms of *partial* (i.e. incomplete) bracket assignment during annotation. We will then discuss the task of compacting a treebank grammar, and report the results of our own experiments, which are based on the use of two techniques: (i) *thresholding* of rules by the number of times they have occurred; and (ii) a method of *rule-parsing*, which has both probabilistic and non-probabilistic variants.[1] The extent of compaction that can be achieved depends upon our requirements for what should *not* be lost during compaction. For example, if we require only the preservation of a CFG’s string set (i.e. so that initial and compacted grammars are weakly-equivalent CFGs) then a treebank grammar can be reduced to a small fraction of its initial size – $\sim$10% in our experiments. If, however, we want to compact a PCFG so that initial and compacted grammars return equally probable results for the most-probable parse of any sentence, then the extent of compaction will be much less – around a 62% reduction in grammar size in our experiments. By weakening this ‘equal probability’ requirement, we have been able to produce greater extents of compaction, without substantial loss of parsing performance (as measured by precision and recall).

These results, we believe, weigh in favour of the potential practical utility of treebank grammars. Although we have chosen to work with a comparatively simple grammar formalism – nonlexicalised PCFG – for our explorations of treebank grammar compaction, we are hopeful that our general approach will adapt to some of the other approaches that have been used in treebank parsing, such as lexicalised PCFG and other formalisms.

2 Treebank grammars and parsing

The previous work on treebank grammars that bears most immediate comparison to our own is that of Charniak (1996). Charniak extracted a PCFG having almost

[1] The results of our work on grammar compaction at an earlier stage were presented in Krotov, Hepple, Gaizauskas and Wilks (1997, 1998). This paper extends these previously reported results and provides a more complete explication of our methods.

16 000 rules from the PTB II.[2] Rules are assigned probabilities according to the equation in (1) (where $\mid r \mid$ is the occurrence count for rule r in the trees of the training corpus, and $\lambda(r)$ returns the non-terminal that r expands):

(1) $$p(r) = \frac{\mid r \mid}{\Sigma_{r' \in \{r'' \mid \lambda(r'')=\lambda(r)\}} \mid r' \mid}$$

The probability of any parse under such a PCFG is simply the product of the rule probabilities for each rule occurrence in the parse. This grammar was evaluated by computing the most-probable parse of the sentences (all of length ≤ 40) in a test set, and comparing this parse (the 'response') to their treebank parse (the 'key'). Standard metrics for evaluating parses are *precision*: the proportion (%) of response constituents that are also present in the key (and are hence 'correct'), and *recall*: the proportion of constituents in the key that are also present in the response (and hence were found). These metrics both have *labelled* and *unlabelled* variants, where the former requires constituents in response and key to have the same syntactic category to be considered the same, whereas the latter does not. Charniak (1996) reports *unlabelled* precision and recall figures of around 80% for his grammar. Charniak (1997a) discusses a number of statistical treebank grammar models, including a simple PCFG one such as that above, for which somewhat lower *labelled* precision and recall figures are reported (around 71% and 75%, respectively).

These results are by no means the best in treebank parsing. Much better figures, i.e. labelled precision/recall of around 87–88%, are reported for *lexicalised* statistical approaches, which employ statistics on the behaviour of individual words (such as Magerman (1995), Collins (1996) and Charniak (1997a)). See Charniak (1997b) for a review of work in this area. Other *non*-lexicalised approaches include Thompson, Mooney and Tang (1997), in which very domain-specific text is addressed, and Schabes, Roth and Osborne (1993), which considers only binary-branching rules. Shirai, Tokunaga and Tanaka (1995) extract a grammar for Japanese from the EDR corpus, and report figures for unlabelled precision/recall of 75/85%. Addressing the issue of how phrase structure should be represented for Korean, Lee, Kim, Han and Kim (1997) extract a grammar from a corpus of 10 000 manually parse-annotated Korean sentences. Johnson (1998) addresses the impact of different tree representations upon the performance of a treebank derived PCFG, and describes a simple node relabelling transformation that improves the labelled precision/recall figures for a PTB II derived PCFG by around 8%.

Another approach within treebank parsing is Data-Oriented Parsing (DOP) (Bod 1992, 1993; Bonnema, Bod and Scha 1997), which collects statistics on the occurrence frequency of all tree fragments within a corpus, derives any sentence by assembling such fragments, and scores any parse in terms of the sum of the probabilities of all

[2] The trees of the training corpus were subject to some limited manipulation before their rules were extracted, i.e. empty categories were ignored, two additional tags for auxiliaries were used to distinguish them from other verbs.

of its derivations (and hence this model presents significant problems in terms of computational overhead[3]).

For our investigations, we have chosen to work with a simple nonlexicalised PCFG, after the fashion of Charniak (1996), rather than some of the other approaches described above, to allow us to focus on the basic idea of treebank grammar compaction. However, we are hopeful that our general approach will adapt to some of the other approaches, and so provide a basis for more efficient practical treebank parsing in general.

3 Grammar extraction and rule set growth

The method we used for extracting the grammar from the corpus is very much as for Charniak (1996), i.e. after some limited, automated, preparation of the trees in the corpus, the rules are simply read of the trees, with rule probabilities assigned as by the equation given in the previous section. There were, however, some minor differences to Charniak's method as regards the prior manipulation of corpus trees. Firstly, unlike Charniak, we have not introduced additional lexical tags to distinguish auxiliaries from other verbs. Secondly, we have eliminated unary projections within trees where both mother and daughter node bear nonlexical categories, moving the daughter tree up to occupy the position of the mother node. Our tree manipulation method is precisely stated as follows (see Gaizauskas (1995) for further exposition):

1. Eliminate all hyphen/equals-attached suffix tags (used to specify grammatical function, semantic role, co-indexation, etc.).
2. Delete all null elements (signaled by label `-NONE-`) and also any nonterminal nodes that are caused to have no children by such deletion.
3. Any node with a single child that bears a non-lexical tag (whether it is so due to deletion of null elements or not) is deleted, with the child taking its place in the next higher level constituent.

For example, the tree fragment in (2a) would become (2b) by deletion of suffixes, and then (2c) by deletion of null structure, and finally (2d) by deletion of unary structure. The resulting tree yields the rules shown in (2e).

(2) a.
```
(PP (IN without)
  (S-NOM
    (NP-SBJ (-NONE- *-1) )
    (VP (VBG missing)
      (NP (DT a) (NN beat) ))))
```

b.
```
(PP (IN without)
  (S
    (NP (-NONE- *) )
    (VP (VBG missing)
      (NP (DT a) (NN beat) ))))
```

c.
```
(PP (IN without)
  (S
    (VP (VBG missing)
      (NP (DT a) (NN beat) ))))
```

d.
```
(PP (IN without)
    (VP (VBG missing)
      (NP (DT a) (NN beat) )))
```

[3] In Bonnema *et al.* (1997), for example, a very domain-specific corpus such as ATIS is used, tree depth is restricted, and the sentences parsed are very short (average length 4.74 words).

e. $$\begin{aligned} PP &\rightarrow IN\ VP \\ VP &\rightarrow VBG\ NP \\ NP &\rightarrow DT\ NN \end{aligned}$$

Perhaps the most contentious aspect of the tree manipulation process described above is the elimination of unary structure. This move was initially made with a view to avoiding problems that unary rules can present for parsing, but we believe that an examination of the unary rules that would otherwise be admitted provides adequate justification for the manoeuvre. We find that the move avoids the inclusion of 66 unary rules, of which just two rules ($NP \rightarrow QP$ and $NP \rightarrow NP$) account for $\sim$73% of the occurrences. Of the 23 categories that can rewrite via unary rules to other non-lexical categories, 12 can rewrite to ≥ 20 others, and 10 can rewrite to themselves (i.e. they participate in cycles), the latter set including all major categories in terms of both number of rules and number of rule occurrences (S, NP, VP, PP, ADJP, SBAR), with the sole exception of ADVP. The presence of categories which mutually rewrite suggests a grammar that is descriptively flawed, since anything derivable from one is derivable from the other, i.e. they serve no discriminating function with respect to each other. It is perhaps not surprising that such a problem arises, given that the formalism of the extracted grammar, with its atomic-categoried context-free rules, fails to encode non-local dependencies, such as that between the bottom of a unary rewrite sequence and the upper-level context in which it appears. The method used in removing unary structure (i.e. replacing a parent node with its child) attempts to partially compensate for this limitation by generating trees in which the bottom structure of a unary rewrite is placed directly into the upper-level context, i.e. directly anchoring the consequences of the rewrite sequence to the higher level context which licensed it.

This grammar extraction method was applied to the *Wall Street Journal* portion of the PTB II, which comprises 2312 files, containing 49 208 sentences, consisting of 1 253 013 tokens. The resulting grammar contains 17 534 rules.[4] An immediate question to ask of such a grammar is how close it is to being complete, at least for this domain, i.e. does it contain all, or nearly all, the rules needed to analyse *Wall Street Journal* English? Given the size of the corpus, it seems unlikely that every last rule has been discovered, but we might hope that most of them have. To investigate this question, we examined the 'accession rate' for new grammar rules, i.e. the rate at which new rules are discovered as new texts are processed. One might expect that as more texts are processed, the number of new rules added per text will be smaller, i.e. as some asymptotic limit is approached. However, in the results we obtained, plotted in Figure 1, rule accession proceeds at a healthy

[4] See Gaizauskas (1995) for a detailed analysis of characteristics of the corpus (e.g. the distribution of sentences by length, and of tree occurrences by depth) and of the extracted rule set (e.g. the distribution of rules and rule occurrences by left-hand side category, or by length of right-hand side).

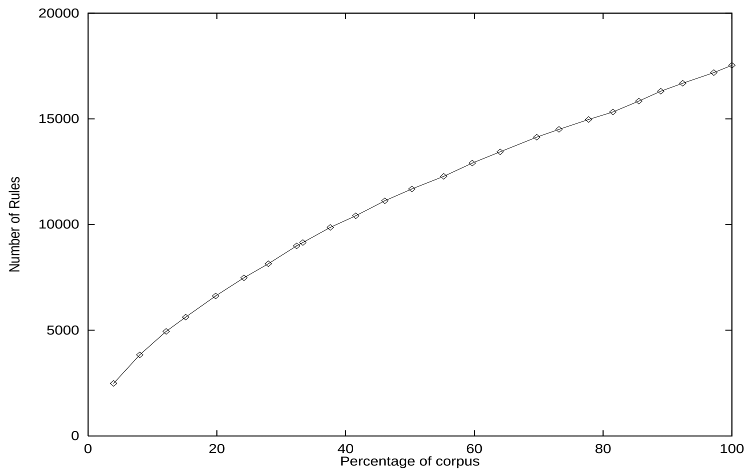


Fig. 1. Rule set growth for Penn Treebank II.

rate throughout processing of the entire corpus, with no suggestion of a limit being approached.[5]

Why should the set of rules continue to grow in this way? One possibility is that natural languages do not have finite rule sets. Another is that the full grammar for this domain is finite but much larger than the rule set so far produced, requiring a much larger tree-banked corpus than is now available for its extraction. In either of these cases, the outlook would be bleak for achieving near-complete grammars from treebanks, particularly given the resource demands of producing parse-annotated text.

A third possibility is suggested by the presence in the extracted grammar of rules such as (3) (where *CC* tags coordinating conjunctions), which is suspicious from a linguistic point of view. We would expect that the text from which it was extracted should more properly have been analysed using rules (4,5), i.e. as a coordination of two simpler NPs.

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

(4) $$NP \rightarrow NP\ CC\ NP$$

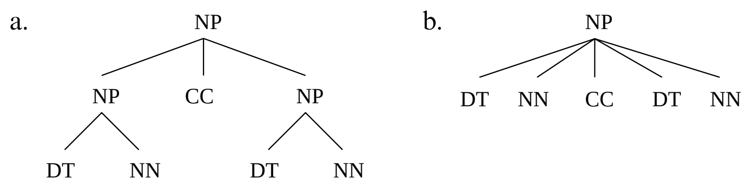
(5) $$NP \rightarrow DT\ NN$$

It is possible that this example reflects a widespread phenomenon of *partial bracketting* within the PTB. Such incomplete structures might arise during parse-annotation of texts, with annotators adding brackets where they are confident that some string forms a given constituent, but leaving others out where they are less confident of the constituent structure. This would result in the extracted rule set containing many rules that are 'flatter' than they should be, corresponding to what should properly

[5] As we reported in Krotov, Gaizauskas and Wilks (1994), a similar result was observed for PTB I. In that case, we extracted 2700+ rules from 45 000+ part-of-speech tokens of input. Again, there was no sign of a limit being approached for rule accession.

be the result of using several grammar rules, but instead showing only the top node and leaf nodes of some unspecified tree structure. For the example above, a tree structure that should properly have been annotated as (6a) has instead received only the partial analysis (6b), yielding the flatter 'partial-structure' rule (3).

(6)



Even assuming some reasonable limit on the length of rule righthand sides, the number of partial-structure rules that could be derived from even a relatively small 'genuine' underlying grammar is potentially enormous. Hence it could be that the continuing rule set growth shown in Figure 1 is more a matter of the continued accumulation of unnecessary 'partial-structure' rules rather than of 'genuine' rules. This idea of 'partial-structure' is immediately suggestive of a route by which such unwanted rules could be identified: *rule-parsing*. For the example above, the rule (3) can be parsed using the rules (4,5), as the structure (6a) demonstrates.

Whether or not it is correct that partial bracketting is widespread with the PTB, and plays a significant role in relation to the continued rule growth, the idea of rule-parsing can be justified and applied to the task of rule elimination in a number of different ways, as we shall discuss in the next section.

4 Approaches to grammar compaction

We shall explore two approaches by which treebank grammars may be reduced in size, or *compacted* as we call it, by the elimination of rules: *thresholding*, and *rule-parsing*.

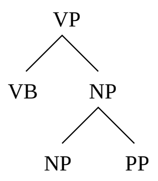
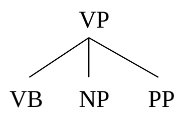
The method of thresholding simply involves discarding rules that have occurred less than some threshold number of times. This method was used by Charniak (1996), who tried discarding all the rules that had appeared only once in the training corpus. For the grammar we have extracted from PTB II, we have tested the result of applying a range of different threshold levels, as reported in the next section.

The method of rule-parsing involves determining whether a rule should be discarded or not on the basis of the parses of the rule itself that can be constructed using the *other* rules of the grammar. This approach is suggested by the idea of there being partial bracketting within the treebank, as discussed in the previous section. Unfortunately, although a partial-structure rule should be parsable using other 'genuine' rules of the grammar (assuming they are present), it does not follow that every rule which is so parsable is a partial-structure rule that should be eliminated. This point is easily seen in relation to the example of the following three (linguistically plausible) rules. The first rule can be parsed using the other two, i.e. under the structure in (10a), but it is not a 'partial-structure' rule, i.e. there are cases where the flatter structure (10b) it allows is linguistically correct.

(7) $$VP \rightarrow VB\ NP\ PP$$

(8) $$VP \rightarrow VB\ NP$$

(9) $$NP \rightarrow NP\ PP$$

(10) a.  b. 

Furthermore, although the phenomenon of partial bracketting clearly is present within the PTB, the *extent* to which it is present, and hence the extent of its impact on the extracted grammar, is something that we are not in a position to quantify. Fortunately, the use of rule-parsing does not rely on the correctness of our hypothesis regarding partial bracketting, but can be justified in other ways.

A useful case to consider is that of *data compression* techniques, which divide into *lossless* vs. *lossy* methods. For lossless methods, the output at decompression is identical to the initial input, i.e. no information is lost. Lossy methods, however, achieve a reduction in the amount of data to be stored by discarding information, so that the output is only an approximation of the input. Such methods can be viewed as employing a model which determines the information that is discarded or preserved in compression. Another characteristic of lossy methods is that they allow a choice of compression ratio, i.e. so that the user can vary a setting on a 'dial' to specify a desired balance between the volume of data to be stored and the fidelity of the output as an approximation of the input.

Rule-parsing could be used in various ways in identifying rules to be eliminated from an initial treebank grammar. To determine precisely *how* rule-parsing should be used requires a decision as to what is to be preserved under compaction (c.f. the 'model' of a lossy compression method). For example, if we decided that compaction should preserve the set of all CFG parse trees that can be assigned to any sentence, then, in fact, no rules could be removed from the initial grammar. (The fact that each rule is drawn from at least one tree in the training corpus demonstrates that it is associated with a non-empty tree set.) If instead, however, we required only that compaction should preserve the set of strings (or part-of-speech sequences) that can be parsed (i.e. assigned at least one parse tree), then we should be able to eliminate any rule that can be parsed using other rules of the grammar. For example, consider a sentence having a parse P which employs a rule R, which can itself be assigned a parse T using only other rules ($\neq R$) of the grammar. Substituting every occurrence of R in P with the tree T yields a parse of the initial sentence not employing R at all, and so eliminating R from the grammar does not result in any loss of coverage. This approach to compaction by rule-parsing, what we call the 'naive' method, is explored in section 6.

An alternative 'model' for compaction, and one which is far more promising for practical utility than the naive method, arises for PCFG treebank grammars. If the intended use of such a PCFG in practical parsing is in generating just the most-probable parse of a sentence (or where there are several equally probable parses, one of them), then a natural requirement to make of compaction is that it should preserve maximal parse probability, i.e. so that for any sentence, the 'best' (most-probable) parse returned under the compacted grammar is of equal probability to (if perhaps not identical with) the best parse returned for the sentence under the uncompacted grammar. In this case, a rule R can be eliminated from the grammar just in case its best parse T using other rules of the grammar has probability equal to or greater than the rule's own probability, since replacing a use of R with T in some larger parse will never reduce the latter's probability. The consequence of eliminating rules in this way will be that the most-probable parses possible for any sentence under the compacted grammar will be a non-empty subset of the most-probable parses allowed by the uncompacted grammar. This approach to compaction by rule-parsing is explored in section 7.

5 Compaction by thresholding

The method of thresholding involves discarding rules under some criterion which is based on the number of times that any rule occurred in the training corpus. Perhaps the simplest model of thresholding is to eliminate any rule that occurred no more than n times, for some $n > 0$. However, other models are possible. For example, Gaizauskas (1995) suggests an approach whereby the most infrequently occurring rules within any category are eliminated until the remaining rules account for not fewer than n% of the rule occurrences for that category, e.g. so that the infrequently used NP rules would be removed until those that remain accounted for, say, 95% of NP occurrences. Gaizauskas applied this method to an initial grammar of around 17 500 rules (extracted very much as for the grammar discussed in section 3), and reports that a 95% threshold reduced the initial grammar to 2144 rules, whilst thresholds of 90%, 80% and 70% resulted in grammars of 872, 240 and 112 rules, respectively.[6] However, no parsing performance figures are available for these grammars.

In this section, we focus on the use of the simple model of thresholding as an approach to treebank grammar compaction, i.e. the method whereby any rule occurring not more than some threshold number of times is removed. This method was previously tried by Charniak (1996), who compared the performance of his full treebank grammar (discussed in section 2) with the subset grammar consisting of only those rules that had appeared more than once in his training corpus, of which there were around 6800. He found the effects of the reduction to be small, i.e. no change in unlabelled recall, and a small reduction in unlabelled precision (from 82% to 81.6%).

[6] It should be noted that the method also involved the removal of rules that became 'unreachable' as a consequence of other rules being eliminated, i.e. all the rules under a given category C would be removed if no other rules remained having C on the right-hand side.

We were interested in determining the effect of using a range of different threshold levels. For these experiments, we used training and test corpora similar to the ones used by Collins (1996). The training corpus contains text from sections 02–21 of the *Wall Street Journal* portion of the PTB II. The test sample consists of the sentences of section 23 that are of length ≤ 40, which number 2245 in total. From the training corpus, a PCFG comprising 15420 rules was extracted. Various reduced grammars were produced by using different thresholds, applied in the obvious way (e.g. applying a threshold of 2 means that rules occurring two times or less were discarded). Table 1 shows the thresholds that were used, the size of the resulting grammars, and the parsing performance results for each. Evaluation was made on the basis of the most-probable parse with top category `S` returned for each sentence,[7] and scored using the `evalb` program developed by Satoshi Sekine and Michael Collins. The PTB trees used for keys in evaluation were also subjected to the tree manipulation process described in section 3,[8] so that the structures used in scoring were of a comparable character to those from which the grammar was extracted. In addition to unlabelled and labelled precision and recall, the results include scores for some of the other metrics that have been widely used in parsing evaluation, which are concerned with so-called *crossing brackets*, where bracketted sequences (irrespective of label) in response and key overlap but neither is contained in the other (so each dominates some terminal elements that the other does not). These metrics are: *crossing brackets* (CB: number of response constituents that cross any key constituents), *zero crossing brackets* (0CB: proportion of sentences whose parse (response) crosses no brackets in the key), *two crossing brackets* (2CB: proportion of sentences whose parse crosses no more than two brackets in the key).

The results in Table 1 show a quite surprising degree of resilience for the parsing performance on the sentences parsed in testing in the face of increasingly severe compaction under thresholding. Even with a threshold of 100, producing a grammar size reduction of 97%, the figure for labelled recall falls by very little, less than 1% as compared to the initial grammar, whilst labelled precision suffers somewhat more, falling by nearly 7% but remaining above the 70% mark. The damaging effect of this compaction is shown, however, by the loss of coverage, i.e. by the test sentences that *fail* to receive a parse, which for the 100 threshold grammar account for nearly one in three of the test set. Looking at the less severe cases of thresholding, we find that a threshold of 1 produces an effect roughly in line with that reported

[7] The obvious alternative here would be to use the most-probable parse returned by the parser, irrespective of the category appearing at the top of the parse tree, in effect allowing the PCFG itself to decide whether the input should be considered a sentence or noun phrase or whatever. We found that selecting the most-probable parse with category `S` appearing as its top node for evaluation gave significantly better parsing performance results. An improvement of around 3% on average in both labelled and unlabelled precision and recall scores is seen pretty much across the board for the grammars discussed in the paper.

[8] Recall that the tree manipulation process has the effect of partially flattening trees, by elimination of unary projections. The alternative approach of scoring directly against the unmodified PTB trees would result in the grammar being penalised for failing to reproduce structure that was not there in the trees from which the grammar was derived. We have found that scoring against unmodified keys makes little difference for precision, but results in recall scores that are somewhat lower, by around 4%.

Table 1. *Compaction by thresholding and parsing performance*

Threshold (N)	Size (N)	Reduction (%)	UR (%)	UP (%)	LR (%)	LP (%)	CB (N)	0CB (%)	2CB (%)	No-parse (%)
0	15 420	0	77.19	80.67	74.08	77.43	2.13	37.97	66.40	0.04
1	6517	58	77.52	80.20	74.42	76.99	2.18	37.71	65.77	0.2
2	4508	71	77.43	79.62	74.27	76.36	2.24	37.63	65.71	0.2
3	3675	76	77.18	79.11	74.02	75.87	2.31	36.86	64.26	0.7
5	2736	82	77.05	78.45	73.94	75.29	2.39	36.17	62.97	1.1
10	1776	88	76.95	77.39	73.60	74.02	2.49	34.00	61.84	2.4
15	1418	91	76.93	76.89	73.55	73.51	2.55	34.08	61.06	3.9
20	1209	92	76.41	75.61	72.82	72.06	2.68	33.80	60.54	4.7
30	933	94	76.07	74.87	72.70	71.55	2.84	32.89	58.03	8.7
50	660	96	76.62	74.35	73.40	71.22	2.84	32.69	57.12	16.5
70	540	96	76.83	74.54	73.86	71.66	2.78	32.85	58.02	25.8
100	443	97	76.82	73.53	73.63	70.67	2.92	31.44	56.21	29.3

by Charniak (1996), i.e. a substantial reduction in grammar size (58%) with very little change in parsing performance or loss of coverage. For subsequent thresholds, greater loss of coverage is observed, although increasing quite gradually at first. Even at threshold 5, which produces a grammar size reduction of 82%, the loss of coverage is just over 1%. At this threshold, we observe only a small decrease in labelled recall (0.1%), and a somewhat larger but still reasonably modest decrease in labelled precision (2.1%). These results suggest that even the simple technique of thresholding can play a role that is greater than previously expected in the task of deriving practically useful grammars from treebanks.

6 Compaction by rule-parsing: naive method

If we require compaction to preserve only the set of strings, or part-of-speech sequences, that can be parsed, then we can eliminate any rule that can be parsed using other rules of the grammar, as justified in section 4. The algorithm we use for such 'naive' compaction is as follows:

```
Input: A context-free grammar $G$

Let $G_C := G$
For Each phrase structure rule $R$ in $G$
    If $R$ can be parsed using $G_C - \{R\}$ Then $G_C := G_C - \{R\}$

Output: The compacted grammar $G_C$
```

Thus, a loop is followed whereby each rule R of the grammar is addressed in turn. If R can be parsed using the other rules (which have not already been eliminated), then R is deleted (and the grammar *without* R used subsequently), and otherwise R is kept in the grammar. The rules that remain when all rules have been checked constitute the compacted grammar.

A variant of this kind of approach to compaction was previously tried by Shirai *et al.* (1995), in deriving a treebank grammar for Japanese. Although the grammar they derive is a PCFG, the rule probabilities are not used by their compaction algorithm. The most notable difference of their method to our own is that their algorithm does not employ full context-free parsing in determining the redundancy of rules, but instead considers only direct composition of rules (so that, in effect, only parse trees of depth 2 are addressed).

An interesting question to ask of our naive compaction method is whether the result of compaction is independent of the order in which the rules are addressed. In general, the result is order dependent, as is shown by the following rules, of which (13) and (14) can each be used to parse the other, so that whichever is addressed first will be eliminated, whilst the other will remain.

(11) $B \rightarrow C$

(12) $C \rightarrow B$

(13) $A \rightarrow B\ B$

(14) $A \rightarrow C\ C$

Order-independence can be shown to hold for grammars that contain no *epsilon* ('empty') rules and no *unary* rules of the form *nonterminal* → *nonterminal*. The tree manipulation process used as part of our grammar extraction method, which was described in section 3, ensures that the treebank grammars produced satisfy this requirement, and so order dependence is not an issue for the results reported in this paper. We will return to the question of order-independence, both for the grammars we have extracted and more generally, at the end of this section.

We applied the naive compaction method to the set of rules extracted from the entire *Wall Street Journal* portion of the PTB II (as described in section 3). The results are striking: the initial set of 17 534 rules reduce to only 1667 rules, a greater than 90% reduction. To investigate the relation between rule set growth and compaction, we conducted an experiment involving a *staged* compaction of the grammar. The corpus was split into 10% chunks (by number of files) and the rule sets extracted from each. To begin the staged compaction, the rule set of the first 10% was compacted. Then the rules for the next 10% were added and the resulting set again compacted, and so on again for each further portion. The results of this experiment are shown in Figure 2. After 50% of the corpus has been processed, the compacted grammar size starts to go down as well as up, ending up smaller at 100% of the corpus than it was at 50%. This suggests that 'new rules' being added during staged compaction either are immediately eliminated or make possible the elimination of rules already present.

Although it is interesting to observe the extent of grammar compaction that is possible under the naive algorithm, the question remains as to the utility of the grammar that results. The only straightforward way to make this evaluation is on the usual 'most-probable parse' basis, i.e. by exploiting the rule probabilities collected during grammar extraction, even though the compaction algorithm treats

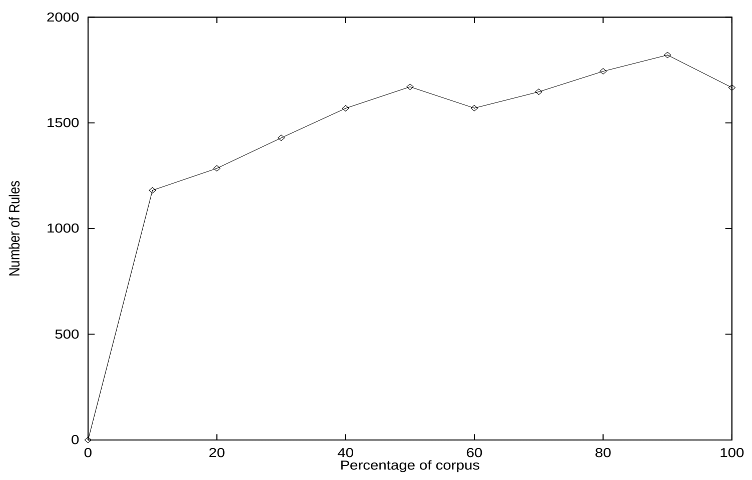


Fig. 2. Compacted grammar size under staged compaction.

Table 2. *Naive compaction and parsing performance*

Grammar	Size (*N*)	Reduction (%)	UR (%)	UP (%)	LR (%)	LP (%)	CB (*N*)	0CB (%)	2CB (%)
Initial	6517	0	77.52	80.20	74.42	76.99	2.18	37.71	65.77
Compacted	1218	81	44.84	26.73	32.44	19.34	9.05	7.05	15.22

the grammar as a CFG. Our results on this question involve not the 17 534 rules discussed immediately above, but rather the thresholded grammar of 6517 rules discussed in the previous section, which was derived from sections 02–21 of the *Wall Street Journal* portion of the PTB II, with rules appearing only once removed.[9] This grammar reduces to just 1218 rules under naive compaction. Parsing performance figures for the two grammars, tested as before on section 23 of the corpus, are given in Table 2, under the usual barrage of metrics. It is clear that naive compaction has had a very damaging effect upon parsing performance. This is really not surprising, given that the compaction method pays no attention to the probabilities that play a crucial role in evaluation. In the next section, we will discuss rule-parsing based compaction that *does* exploit rule probabilities.

We return now to the issue of order-independence for the rule-parsing based compaction algorithm described above. For the specific grammar that we have

[9] The full grammar of 17 534 rules would be unsuitable for the immediate purpose of parsing evaluation as it is derived from the entire *Wall Street Journal* portion of the PTB II, and so does not provide for a separation of training and testing data. The unthresholded grammar of 15 420 rules extracted as described in section 5 would be free of this problem. However, the thresholded grammar of 6517 rules was chosen in preference to the unthresholded grammar because it fairly nearly preserves the latter's parsing performance and coverage, but its smaller size is such as to considerably reduce the computational expensive of the experiments performed.

extracted, order-independence of compaction does hold, as we shall demonstrate in the next paragraph. In the general case, however, order-independence will not hold, i.e. different orders of selecting rules under the compaction algorithm are liable to result in different final compacted rule sets, as was illustrated by the example involving rules (11–14) above. This fact in no way precludes the use of rule-parsing based compaction, but in this general case, we might expect the compaction algorithm to be used in conjunction with some predecided criterion for determining the order of rule selection. A number of alternatives immediately present themselves, i.e. rules might be ordered in terms of increasing, or decreasing, frequency of occurrence in the training data, or in terms of the length of their right-hand sides. It remains an open question as to whether any such ordering method will in general result in better parsing performance than any of its alternatives.

We claimed above that order-independence of compaction under the rule-parsing algorithm *does* hold for the treebank grammars that we have extracted, as a consequence of the tree manipulation process that is part of our extraction method. We shall now briefly present a proof of this claim. The crucial fact that makes this proof possible is that, due to the tree manipulation process, the grammars we derive contain no *epsilon* rules (i.e. rules with 'empty' right-hand sides) and no *unary* rules of the form *nonterminal* → *nonterminal*. The absence of such rules in a grammar G allows us to establish a lemma that any parse under G of a rule $R \in G$ cannot use R itself, unless the parse is *trivial*, i.e. consists *only* of R. There are two cases to be addressed for this lemma: (i) a parse consisting of a use of R plus some unary rules cannot be a parse of R, since the latter all take the form *nonterminal* → *terminal*, (ii) any *other* non-trivial parse involving R additionally includes a use of a branching rule (i.e. with right-hand side length >1), so the parse must have more leaf nodes than R has right-hand side categories, and hence cannot be a parse of R. This lemma allows a straightforward demonstration of order-independence in compaction. Recall that the elimination of a rule during compaction leaves the coverage unchanged. Consider two grammars G_1 and G_2 which are alternative intermediate stages (i.e. under different ordering of rule selection) in the course of compacting a grammar G that contains no unary or epsilon rules. Assume both grammars contain a rule R, which can be eliminated from G_1 but not from G_2, i.e. R has a parse under $G_1 - \{R\}$, but not $G_2 - \{R\}$. Since G_1 and G_2 have the same coverage (the same as G), the rules of $G_1 - \{R\}$ used to parse R must themselves be parsable (trivially *or* non-trivially) under G_2. It follows (by combining those parses together) that R has a parse under G_2 which (by the lemma) cannot contain R, and so is a parse under $G_2 - \{R\}$, i.e. R *can* be eliminated from G_2, contradicting our assumption. Hence, R can be eliminated either from both G_1 and G_2, or from neither, and so the order in which rules are addressed in compaction is irrelevant to the ultimate eliminability of any given rule, and hence also to the overall result of compaction.

7 Compaction by rule-parsing: probabilistic method

When our treebank grammar is a PCFG, a natural requirement to make of compaction is that it should preserve maximal parse probability, i.e. so that the 'best' parses of any string under initial and compacted grammars are of equal probability. Given this requirement, a rule R can be eliminated from a grammar G just in case its best parse under $G - \{R\}$ is at least as probable as R itself.

We noted in section 4 that lossy methods of data compression allow for a choice of compression ratio, i.e. specifying a balance between data reduction and the degree of approximation. A compaction approach exploiting rule probabilities is straightforwardly adapted to allow a similar degree of freedom, by basing our criterion for rule elimination on the *ratio* of the probabilities of a rule's best parse and the rule itself (i.e. dividing the former by the latter[10]). In these terms, a compaction method that preserves maximal parse probability is implemented by requiring that the ratio's value must be ≥ 1 for a rule to be eliminated. By lowering this threshold ratio to values that are increasingly below 1 (but > 0), we can progressively increase the extent of compaction (i.e. since more rules will be eliminated), but at the expense of the goal of preserving maximal parse probability, which is progressively less well approximated.

A slightly different way of implementing what is essentially the same idea is to compute the best-parse/rule probability ratio for each rule, and to use this ratio to rank the rules, with lower-valued ratios giving higher rank. Different 'compression ratios' can then be achieved by eliminating a greater or lesser proportion of the lower ranked rules. In terms of the basic criterion of preserving the PCFG's maximal parse probability behaviour, rules whose best-parse/rule probability ratio is ≥ 1 can always be eliminated. Rules for which the best-parse/rule probability ratio is 0 are ones that have no parse under the rest of the grammar, so let us assume that these rules will always be retained. Given these two bounding cases of what is always retained and what is always eliminated, we can specify a 'setting' for the compression ratio in terms of the proportion of the intervening rules that are retained, i.e. the rules whose best-parse/rule probability ratio r is such that $0 < r < 1$. Thus, with a 'ranked compaction' setting of 0% ('0%RC'), all of these intervening rules (i.e. with $0 < r < 1$) are discarded, so only the rules that have no parse under the other rules of the grammar are retained. Hence this setting is equivalent to naive compaction. On the other hand, a ranked compaction setting of 100% ('100%RC') will mean that all of the intervening rules are retained, and so the criterion of preserving the PCFG's maximal parse probability behaviour is not compromised.

For our experiments on probabilistic compaction, we used the same initial grammar and testing method as we did for our experiments on naive compaction, i.e. the thresholded grammar of 6517 rules derived from sections 02–21 of the *Wall Street Journal* portion of the PTB II, with rules appearing only once removed, and with testing again on section 23 of the corpus.[11] Table 3 shows the results of these exper-

[10] Where a rule cannot be parsed using the other rules of the grammar, a maximal parse probability of zero is assigned. Hence, dividing the parse probability by the rule probability, rather than *vice versa*, avoids the problem of zero valued denominators.

[11] As before, this thresholded grammar was chosen in preference to the unthresholded grammar to ease the computational expense of the experiments made. The unspecified No-Parse scores for each of the compacted grammars in Table 3 are all 0.2%, as for the initial thresholded grammar, since probabilistic compaction, like naive compaction, preserves coverage.

Table 3. *Probabilistic compaction and parsing performance*

Grammar (%-ranked)	Size (*N*)	Reduction (%)	UR (%)	UP (%)	LR (%)	LP (%)	CB (*N*)	0CB (%)	2CB (%)
Initial	6517	0	77.52	80.20	74.42	76.99	2.18	37.71	65.77
100	5917	9	77.65	80.23	74.54	77.02	2.18	38.06	65.68
90	5447	16	78.38	78.77	75.19	75.56	2.34	36.99	63.41
80	4977	24	79.03	76.72	75.72	73.51	2.52	35.65	60.37
70	4507	31	78.38	74.74	75.07	71.58	2.70	34.63	59.04
60	4037	38	77.01	71.30	73.51	68.06	3.01	31.91	54.84
40	3098	52	72.44	60.90	68.73	57.79	3.91	26.64	46.32
20	2158	67	65.11	49.60	59.81	45.57	5.46	19.86	35.07
0	1218	81	44.84	26.73	32.44	19.34	9.05	7.05	15.22

iments, including the extent of grammar reduction for different 'ranked compaction' settings, and the parsing performance figures for the resulting rule sets. For a ranked compaction setting of 100%, which achieves a grammar size reduction of 9%, the parsing performance is essentially unchanged from that of the initial grammar, as we would expect.[12] Looking particularly at the figures for labelled precision and recall, we observe that decreasing the ranked compaction setting in 10% steps produces a relatively slow decline in precision, but produces at first an *increase* in recall. Only at 60%RC does the recall fall below its value for the 100%RC grammar. This observation of increasing recall scores is somewhat surprising. The explanation for this change is presumably that the elimination of rules by rule-parsing will tend to produce test parses with more structure, i.e. which are less flat. It might then be that compaction by rule-parsing actually improves the grammar by elimination of partial-structure rules. However, it could instead just be that, in the usual trade off between recall and precision, rule elimination slightly tips the balance in favour of recall. The 70%RC grammar, which is the smallest grammar whose recall is not less than that of the 100%RC grammar, achieves a grammar size reduction of 31%. Since the simple thresholding used in generating the initial rule set used here produced a 58% reduction, the overall combined reduction rates for the 100%RC and 70%RC grammars are 62% and 71%, respectively.

We conducted a further set of experiments concerned with investigating the effect on a treebank grammar, and its compacted variants, of the presence in the corpus of the 'categories' `X` and `FRAG`. According to the PTB annotator's guide (Bies *et al.*, 1995), these labels are used where the annotator is uncertain of the structure that should be assigned, so it is not clear that it makes sense for them to be included in a treebank grammar, if the grammar is to be viewed as a representation of grammatical knowledge (rather than of grammatical ignorance). The label `X` is used

[12] Johnson (1998), citing Krotov *et al.* (1997), applies the idea of eliminating rules whose best-parse under the rest of the grammar is at least as probable as the rule itself to a PCFG derived from the PTB. This method corresponds to the '100% ranked compaction' case in the present scheme. The method achieves a size reduction of just under 9% for his grammar.

Table 4. *Effect of removing* X *and* FRAG

Grammar (%-ranked)	Size (*N*)	Reduction (%)	UR (%)	UP (%)	LR (%)	LP (%)	CB (*N*)	0CB (%)	2CB (%)
Initial	6430	0	77.72	80.43	74.74	77.34	2.17	37.85	66.07
100	5831	9	77.84	80.45	74.86	77.37	2.16	37.76	66.07
90	5365	17	78.49	78.89	75.42	75.81	2.33	36.66	63.38
80	4899	24	79.43	77.12	76.23	74.01	2.47	35.80	60.56
70	4432	31	78.40	74.75	75.19	71.70	2.72	34.06	58.50
60	3966	38	77.00	71.29	73.63	68.17	3.04	31.46	54.49
0	1169	82	44.84	26.76	33.18	19.80	9.17	6.84	14.50

to mark 'unknown, uncertain, or unbracketable' material. The label FRAG is used for 'clause-like' fragments (from which, in fact, so much may be absent that only a unit of some non-clause category remains, such as NP or ADJP). For these experiments, we eliminated all rules containing the two labels from the 6517 rule grammar, leaving a rule set of size 6430. We also ignored any test corpus parses that included the labels, which reduced the sample size from 2245 to 2197 sentences. The results for this grammar and test set are presented in Table 4, as well as for the grammar's variants under ranked compaction. The results indicate a slight improvement in parsing performance for the initial grammar, which is, for the most part, maintained under compaction.

8 Conclusion

The automatic extraction of grammar rules from parse-annotated corpora, such as the Penn Treebank, provides an attractive route to the creation of broad-coverage grammars. Such grammars can be very large, however, presenting obvious problems for their subsequent practical use in parsing. This concern has lead us to investigate ways in which such treebank grammars can be reduced in size, or 'compacted'. The experiments on grammar compaction reported in this paper have addressed the use of methods involving rule-parsing and thresholding. Our results indicate that these methods can achieve a substantial reduction in grammar size with little or no loss in parsing performance. Although we have chosen to work with the formalism of nonlexicalised PCFG, we are hopeful that our general approach will adapt to other approaches that have been used in treebank parsing, such as lexicalised PCFG. Any significant improvement of the efficiency of parsing using treebank grammars, achieved by compaction or otherwise, will contribute to the likelihood that they will find real use in practical application.

Acknowledgements

We thank two anonymous referees for their valuable comments on the paper.

References

Bies, A., Ferguson, M., Katz, K. and MacIntyre, R. (1995) *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. Available at: `ftp://ftp.cis.upenn.edu/pub/treebank/doc/manual`.

Bod, R. (1992) A computational model of language performance: Data Oriented Parsing. *Proceedings of COLING'92*, pp. 855–859. Nantes, France.

Bod, R. (1993) Using an annotated corpus as a stochastic grammar. *Proceedings of European Chapter of the Association for Computational Linguistics '93*, Utrecht, The Netherlands.

Bonnema, R., Bod, R. and Scha, R. (1997) A DOP model for semantic interpretation. *Proceedings of European Chapter of the Association for Computational Linguistics*, pp. 159–167.

Charniak, E. (1996) Tree-bank grammars. *Proceedings 13th National Conference on Artificial Intelligence* (*AAAI-96*), pp. 1031–1036. MIT Press.

Charniak, E. (1997a) Statistical parsing with a context-free grammar and word statistics. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. MIT Press.

Charniak, E. (1997b) Statistical techniques for natural language parsing. *AI Magazine*. **18**(4): 33–44.

Collins, M. (1996) A new statistical parser based on bigram lexical dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 184–191.

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

Johnson, M. (1998) PCFG models of linguistic tree representations. *Computational Linguistics*, **24**(4): 613–632.

Krotov, A., Gaizauskas, R. and Wilks, Y. (1994) Acquiring a stochastic context-free grammar from the Penn Treebank. *Proceedings of Third Conference on the Cognitive Science of Natural Language Processing*, pp. 79–86. Dublin, Ireland.

Krotov, A., Hepple, M., Gaizauskas, R. and Wilks, Y. (1997) Compacting the Penn Treebank grammar. *Technical Report CS-97-04*, Department of Computer Science, University of Sheffield.

Krotov, A., Hepple, M., Gaizauskas, R. and Wilks, Y. (1998) Compacting the Penn Treebank grammar. *Proceedings 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp. 699–703.

Lee, K. J., Kim, J.-H., Han, Y. S. and Kim, G. C. (1997) Restricted representation of phrase structure grammar for building a tree-annotated corpus of Korean. *Natural Language Engineering* **3**: 215–230.

Magerman, D. (1995) Statistical decision-tree models for parsing. *Proceedings 33rd Annual Meeting of the Association for Computational Linguistics*, pp. 276–283.

Marcus, M., Santorini, B. and Marcinkiewicz, M. A. (1993) Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19**(2): 313–330.

Schabes, Y., Roth, M. and Osborne, R. (1993) Parsing the Wall Street Journal with the inside-outside algorithm. *Proceedings Sixth Conference of the European Association for Computational Linguistics*, pp. 341–347.

Shirai, K., Tokunaga, T. and Tanaka, H. (1995) Automatic extraction of Japanese grammar from a bracketed corpus. *Proceedings of Natural Language Processing Pacific Rim Symposium*, pp. 211–216. Korea.

Thompson, C. A., Mooney, R. J. and Tang, L. R. (1997) Learning to parse natural language database queries into logical form. *Proceedings of the ML-97 workshop on Automata Induction, Grammatical Inference and Language Acquisition.*