Experiments in Structure Preserving Grammar Compaction

Mark Hepple[°] **and Josef van Genabith**[*]
[°] Sheffield University — [*]Dublin City University
Dept. of Computer Science — Computer Applications
m.hepple@dcs.shef.ac.uk — josef@compapp.dcu.ie

Abstract

Structure preserving grammar compaction (SPC) is a simple CFG compaction technique originally described in (van Genabith *et al.*, 1999a, 1999b). It works by generalising category labels and in so doing plugs holes in the grammar. To date the method has been tested on small corpra only. In the present research we apply SPC to a large grammar extracted from the Penn Treebank and examine its effects on rule treebank grammar size and on rule accession rates (as an indicator of grammar completeness).

1 Introduction

Tree banks and resources compiled from treebanks are potentially very useful in NLP. Grammars extracted from treebanks — so called treebank grammars (Charniak, 1996) — can form the basis of large coverage NLP systems. Such treebank grammars, however, can suffer from several shortcomings: they commonly feature a large number of flat, highly specific rules that may be rarely used, with ensuing costs for processing (load) under the grammar. Furthermore, the observation that the rate of acquisition of 'new rules' continues with little reduction as the treebank is processed suggests that the derived grammar is far from complete.

Some methods aimed at reducing the size of, or 'compacting', treebank grammars, without undermining parsing performance, have been reported, which involve either *thresholding* or *rule parsing*. In a thresholding approach (henceforth THC), rules which occur less than a set number of times are dropped from the grammar (Charniak, 1996). In the rule parsing approach (henceforth RPC), rules whose RHSs can be parsed by other rules in the grammar yielding the same LHS are discarded (Krotov *et al.*, 1998, 1999) . Combined approaches employ both thresholding and rule parsing techniques (Krotov *et al.*, 1998, 1999). Where a treebank grammar is a *probabilistic* CFG, both the initial grammar and its compacted variants can be evaluated by comparing trees from a test component of the corpus to (probabilistically) best-parse results for the same sentences computed using the PCFG. While the above compaction methods can produce substantial compaction rates (with either negligible or only limited loss in performance), they can be criticised for not being structure preserving: they eliminate structural possibilities and in doing so may exclude the linguistically 'correct' structure for some given input.

Recently, van Genabith *et al.* (1999a, 1999b) presented a structure preserving grammar compaction method (henceforth SPC) which is claimed to respect linguistically motivated structure. The basic idea is simple: the method generalises category labels (i.e. collapses categories into supercategories) and this can have the effect of collapsing together rules that were previously distinct. Collapsed rules are isomorphic to the original rules up to node relabelling. Such collapsing of rules can serve to plug 'holes' in the original treebank grammar, in a way that is explained below. To date SPC has been tested only on small rule sets (< 1000). An open question has been what effect SPC has on treebank grammar rule accession rates.

In the present paper we apply SPC to a large reference corpus — the second release of the Penn Treebank (PTB II), (Marcus et al., 1999) — in order to (i) compare SPC with THC, RPC and combined approaches and (ii) answer the question of what effect SPC has on rule accession rates. The paper is structured as follows, Firstly, we explain and contrast RPC and SPC in some detail. Secondly, we report on a number of possible SPC

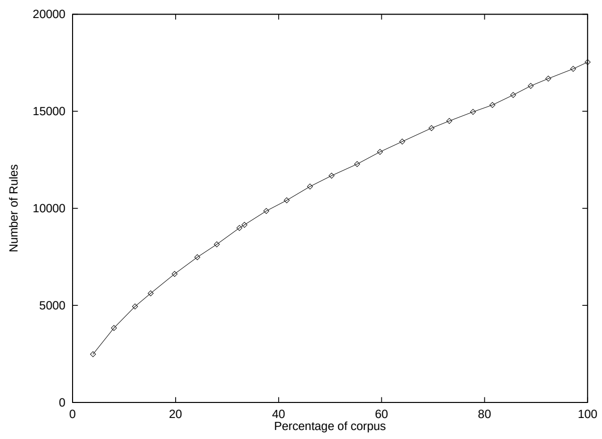


Figure 1: Rule Set Growth for Penn Treebank II

experiments with different degrees of compaction. Finally, we conclude and outline further work.

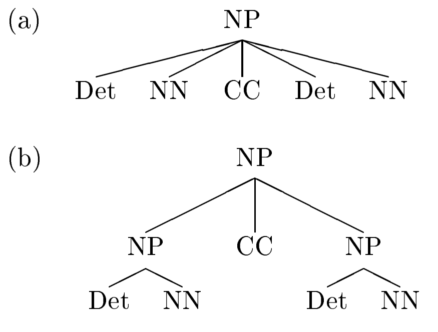
2 Grammar Compaction Techniques

2.1 Rule Accession During Treebank Grammar Extraction

Krotov *et al.* (1998) examined rule 'accession rate' during treebank grammar extraction, i.e. the rate at which new rules are discovered as new corpus texts are processed, for which results are plotted in Figure 1. One might expect that the number of new rules added per text would decrease as the corpus was processed, i.e. as some asymptotic limit is approached. The fact that rule accession does not decrease in this way suggests that the derived grammar is far from complete.

2.2 Rule Parsing Grammar Compaction (RPC)

Krotov *et al.* (1998) suggest a possible explanation of this rule growth phenomenon in terms of *partial* (i.e. incomplete) bracket assignment in corpus trees, giving trees such as tree (a) following, which appears should have instead been analysed as in (b).

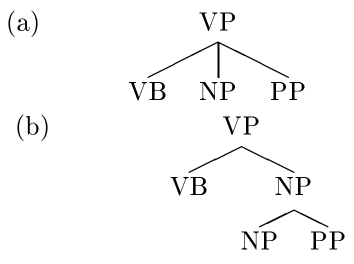


Such partial brackettings will give rise to overly flat 'partial structure' rules in the treebank grammar, such as rule (a) in the following rules derived from the above trees:

```
NP -> Det NN CC Det NN      (a)
NP -> NP CC NP              (b)
NP -> Det NN                (c)
```

Krotov *et al.* (1998,1999) suggest that such 'partial structure' rules could be eliminated by a method of rule parsing, e.g. so that rule (a) above could be recognised as a partial structure rule by virtue of the fact that it can be parsed using rules (b) and (c), and hence discarded.

The problem with this idea is that rule parsing, naively applied, can result in the elimination of linguistically correct rules. For example, for the two linguistically plausible structures below, the rules (b,c) derived from tree (b) can be used to parse the rule (a) derived from tree (a).



```
VP -> VB NP PP     (a)
VP -> VB NP        (b)
NP -> NP PP        (c)
```

The deletion of rule (a) here eliminates a linguistically motivated attachment possibility. It is in this sense that RPC is not structure preserving, or, to put it another way, is structure eliminating. Krotov *et al.* (1999) show how the negative consequences of RPC for parsing performance can be avoided by taking rule probablitities into account during compaction, but the use of rule probabilities to decide which rules are retained or discarded is clearly something other than a criterion of linguistic correctness.

2.3 Structure Preserving Grammar Compaction (SPC)

SPC was first presented in (van Genabith *et al.*, 1999a, 1999b) in the context of semi-automatically annotating treebanks with higher level feature structure (in their case Lexical-Functional Grammar f-structure [Kaplan and Bresnan,83]) information:[1] to do this (van Genabith *et al.*, 1999a, 1999b) extracted a grammar

[1]And further to develop stand-alone unification grammar resources based on treebank resources.

(approx. 500 rules) from the publicly available subset of the AP treebank [REFERENCE], manually annotated those rules with feature structure equations, for each tag class provided macros associating lexical items in the tag class with corresponding feature structures and then used this annotated grammar to automatically "reparse" the original treebank entries (not the strings) simply following the CFG annotations provided by the original annotators. A constraint solver resolves the feature structure annotations encountered in the process and in doing so generates a feature structure. The method makes the CFG part of treebank feature structure annotation deterministic (in fact the whole process is deterministic if the feature structure annotations are) and guarantees that the feature structure constructed is induced by the "best" CFG analysis available (namely the one provided by the human annotators).

The AP treebank employs a large number of highly discriminating lexical (approx. 180) and phrasal tags (approx. 50). In contrast to "standard" LFG CFG grammar components, the grammar extracted from the AP fragment features flat rules, many of which, owing to the large number of lexical and phrasal categories. differ only minimally. van Genabith *et al.* (1999a, 1999b) observed that in many respects the treebank grammar they extracted looks like a unification grammar compiled out into atomic categories. In order to make the grammar more like a standard LFG CFG, they tried generalising the grammar by providing lexical macros and modifying grammar rule annotations so as to move information originally encoded in the fine-grained tagset into the feature structure. This makes many tag label distinctions superfluous and such tags can be collapsed into supertags. Generalising tag labels (and phrasal categories) in this way has the following effect on the CFG:

Assume a general CFG rule of the form `U -> X Y Z`. Assume further that this rule is not part of the grammar extracted from a treebank. The treebank tagset doesn't even have the tags/categories `X`, `Y` and `Z`. Instead, the treebank features fine-grained variants of `X`, `Y`, `Z`. Lets assume there are `l` types of `X` subcategories `X_1 .. X_l`, `m` types of `Y` categories and `n` types of `Z` subcategories. Furthermore, from some particular treebank fragment we extract the following rules expanding `U`:

```
U -> X_3 Y_1 Z_5
U -> X_1 Y_1 Z_2
U -> X_3 Y_2 Z_2
```

Clearly, the theoretical - not necessarily the linguistic - space of possibilities for rules expanding U of this type is:

```
U -> X_1 Y_1 Z_1
U -> X_1 Y_1 Z_2
   ..      ...
U -> X_l Y_m Z_n
```

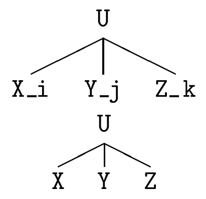
that is `l*m*n` possibilities. The rule set extracted from the treebank clearly misses some of these possibilities. It is likely that any given treebank fragment is going to feature only a subset of the theoretically possible rules. The hypothesis is now that it is in this sense that treebank grammars have holes, i.e. are incomplete. Or at least that this kind of incompleteness plays an important aspect in rule accession rates. Now consider the following: what happens if the fine-grained category sets are generalised into supercategories? For the example at hand, lets assume the following collapsing:

```
{X_1, ... ,X_l} => X
{Y_1, ... ,Y_m} => Y
{Z_1, ... ,Z_n} => Z
```

This has the effect that (i) the above grammar rules extracted from the treebank collapse into the single general rule `U -> X Y Z`, (ii) any possible grammar rule of the form `U -> X_i Y_j Z_k` collapses into `U -> X Y Z` and (iii) thereby effectively the whole space `l*m*n` of possible rules collapses into the single general rule. It is in this sense that generalising tags/categories as described above plugs holes in grammars. Note further that the method is structure preserving in the following sense: for each local subtree of the form (a) below the method will generate form (b), where the latter is isomorphic to the first tree up to node relabelling. SPC can thus be seen as a simple node relabelling process.

(a)

(b)



2.4 Comparative Analysis of SPC and RPC

In this section and in the rest of the paper we study SPC as a pure CFG compaction method, i.e. we abstract away from the fact that in the original approach SPC was partially motivated by shifting or repackaging information from the CFG to the feature structure level. This enables us to compare SPC with RPC in terms of their general characteristics, and to study the impact of SPC on treebank grammar size and its effect on rule accesssion rates.

RPC and SPC show a number of interesting similarities and differences. Both reduce the number of rules. Both can be viewed *lossy* or *lossless* methods,[2] depending on how one choses to look at them: They are lossless in that neither shrinks the string set defined by the original grammar. RPC is lossy in that it eliminates structural analyses available under the original grammar. For SPC matters are more complicated. It is lossless in the sense that for each structure provided by the original grammar it will provide a structure isomorphic up to node relabelling. However, each structure allowed by the SPC grammar may correspond to multiple structures under the original category set, only some of which may actually have occurred in the original rule set extracted from the treebank. The crucial question is whether these additional structures allowed for (and their associated strings) are linguistically correct or not.

2.4.1 Structure

RPC clearly changes (better: eliminates) structure. It does so by removing analysis possibilities. SPC is structure preserving up to node relabelling.

2.4.2 Ambiguity

RPC reduces ambiguity. It reduces the number of rules available and never changes the rest of the rules. The net effect is that of removing analysis choices.

With SPC the matter is more complicated: it can both reduce and induce ambiguity. Consider the following example. Assume that certain types of prepositional phrases `PP_n` exclusively go with noun phrases `NP` while another type of prepositional phrase `PP_v` always attaches to `VP`. In grammar `G1` we find

```
G1: VP -> V NP PP_v VP -> V NP  NP -> NP PP_n
G2: VP -> V NP  PP  VP -> V NP  NP -> NP PP
```

with `G2` a collapsed version of `G1`. `G1` is clearly deterministic for both `V NP PP_v` and `V NP PP_n` sequences while `G2` associates `V NP PP` with two different analyses.

SPC can also reduce ambiguity. However it can only do this for simple lexical ambiguity. Structural ambiguities in the original grammar can never be conflated in SPC because SPC guarantees structural isomorphism up to node relabelling:

```
G1: tag1(word) tag2(word)
G2: tag(word)
```

where `G2` is a collapsed version of `G1`. Since structural ambiguities in the original grammar can never be conflated, the effect of SPC can only ever be a net increase in ambiguity.

2.4.3 Recursion

Clearly, RPC can only affect recursion in the grammar by the elimination of recursive rules. By contrast SPC can have the effect of introducing more recursion into the grammar. Let `VP_i, VP_j` be VP types that are collapsed to a single category `VP`. This has the effect of converting the following non-recursive rule in `G1` to a recursive one in `G2`.

```
G1:   VP_i -> V  VP_j
G2:   VP -> V  VP
```

2.4.4 Weak Generative Capacity

Both RPC and SPC change strong generative capacity. RPC eliminates flat structures, SPC relabels nodes. That much is clear. RPC never changes weak generative capacity, since for any rule that is eliminated, the other rules used to parse it can serve in its place in any structures in which it appeared. SPC, however, can affect weak generative capapcity.

For example, assume we have three noun tags `n1`, `n2`, `n3`. And we find the following PS rules in our treebank:

```
np -> det n1
np -> det n2
np -> det n3
np -> n1
np -> n2
```

It seems plausible to collapse the nouns together, i.e. `n1,n2,n3 => n0`, which reduces the rule set to just:

```
np -> det n0
np -> n0
```

This compacted grammar covers a case that was not covered by the initial grammar, for which an additional rule `np -> n3` would have been required. If this additional case is linguistically correct, then the grammar has not only been compacted but also improved by correct generalisation. However, the additional rule might be absent from the data (corpus) because it is linguistically wrong, in which case the compacted grammar overgenerates (e.g. `n1` might be mass noun, `n2` plural count noun and `n3` singular count noun).

[2]This terminology is drawn from the area of *data compression*. For lossless methods, the output at decompression is identical to the initial input. Lossy methods, however, reduce the data to be stored by discarding information, so that the output is only an approximation of the input. Such methods involve a model which determines the information that is discarded or preserved.

Mapping #1	Mapping #2	Mapping #3
`{ NN NNP }`	`{ NN NNP NNPS NNS }`	mapping #1 *plus*:
`{ NNPS NNS }`	`{ PRP PRP$ }`	`{ NP WHNP }`
`{ PRP PRP$ }`	`{ DT PDT }`	`{ PP WHPP }`
`{ JJR JJS }`	`{ JJ JJR JJS }`	`{ S SBAR SBARQ SINV SQ }`
`{ WDT WP WP$ }`	`{ WDT WP WP$ }`	`{ ADJP WHADJP }`
`{ VB VBP VBZ VBD }`	`{ VB VBP VBZ VBD VBG VBN }`	`{ ADVP WHADVP }`
`{ RBR RBS }`	`{ RB RBR RBS WRB }`	
`{ ‘‘ ’’ }`	`{ ‘‘ ’’ }`	
`{ . : , -- }`	`{ . : , -- }`	

Figure 2: SPC mappings used in experiments

The alternative mapping `n1,n2 => n0` produces a smaller reduction in the rule set below, but does so without allowing bare singular count noun `NPs`.

```
np -> det n0
np -> det n3
np -> n0
```

Hence, many *prima facie* plausible mappings may turn out to be damaging and the degree of collapsing has to be guided by linguistic knowledge to keep overgeneration at bay.

3 Experiments

For our experiments, we used Wall Street Journal portion of the PTB II. Before rules were extracted, treebank annotations were preprocessed, with tag-suffixes and ‘empty’ nodes being deleted, and with null and (non-prelexical) unary structure being collapsed (cf. Krotov *et al.*, 1998,1999). The set of tags extracted from this fragment contains 46 lexical and 28 phrasal tag types.

We conducted three experiments with different degrees of collapsing. The mappings used collapsed categories as shown in Figure 2. Thus, the first line `{ NNPS NNS }` of Mapping #1 indicates that categories `NNS` and `NNPS` (for plural common and plural proper nouns, respectively) were collapsed to a single category. The first two mappings involve only lexical tags, with the second being more ‘severe’ (i.e. more collapsing) than the former (e.g. the second collapses all nouns, both singular and plural, to a single category). The third mapping extends the second mapping with some additional collapsing of phrasal categories.

The results for ‘rule accession’ under SPC using the three mappings are plotted in Figure 3, alongside those for rule accession without compaction. (Note that the data points along the x-axis here correspond to the different sections (00 throught 23) of the the PTB II/WSJ corpus, e.g. the first point being for rules derived from section 00 alone, the next for section 00 and 01, the next for sections 00 through 02, and so on.)

The different degrees of severity of the two lexical mappings is clearly shown in the results, with Mapping #1 producing approximately half the compaction effect achieved by Mapping #2. The latter mapping achieves a reduction of the grammar to about two-thirds of its initial size, this effect being roughly constant throughout processing of the corpus. The addition in Mapping #3 of phrasal category collapsing produces surprisingly little additional compaction effect over that achieved by Mapping #2.

It is clear from these results that SPC does not provide a complete answer to the problem of continued rule acquisition as discussed earlier, i.e. even with SPC, the rule set continues to grow with little reduction as the treebank is processed, although growth rate is lower than without SPC. Even so, the results suggest that the method may prove valuable in producing treebank grammars of manageable size.

4 Evaluating Alternative SPC Mappings

As noted, some plausible SPC mappings on tags may turn out to be damaging, i.e. might introduce linguistically invalid generalisation into the compacted grammar. Ultimately, the mappings employed will need to be created on the basis of informed linguistic intuitions, and even well-formed intuitions allow for the possibility of different viewpoints. The question arises as to whether anything other than purely subjective criteria can be used in choosing between alternative plausible mappings.

A promising answer to this question arises in the context of Probabilistic CFG (PCFG) treebank grammars. The SPC method readily generalises from CFG to PCFG. The only complication

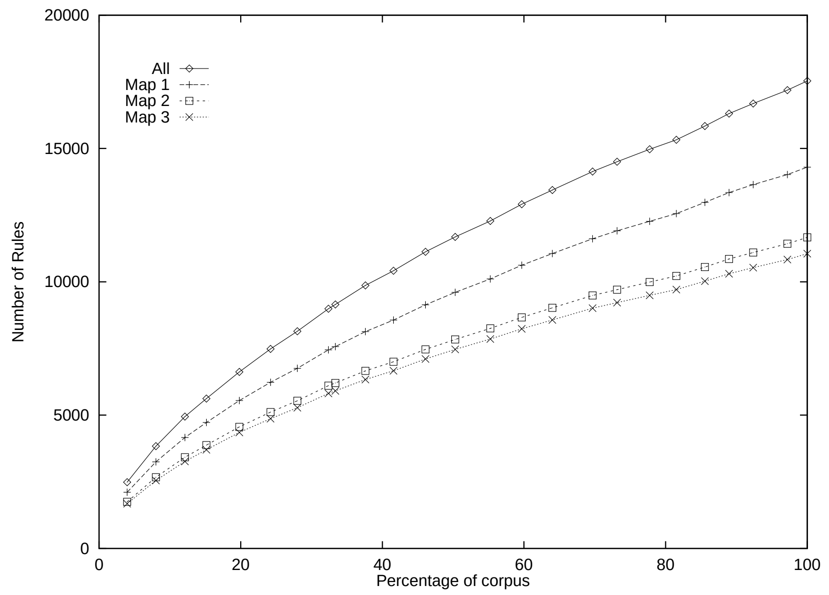


Figure 3: Results of SP Compaction Experiments

is how to provide an occurrence count for rules in the compacted grammar, as needed to provide the rule probabilities of the PCFG. This, however, is straightforward: we only need to sum the occurrence counts for the multiple rules that collapse together to give each rule of the compacted grammar. It could be expected that this move has benefits with respect to the sparse data problem as it arises for PCFGs. This expectation is however not fulfilled as some of our current experiments with collapsed PCFGs (not reported here in detail) show. The compacted PCFGs are evaluated by computing best parses from sentences from a test component of the treebank and comparing them to the "gold standard" parse provided for them in the treebank itself, using standard evaluation metrics such a bracket precision and recall, crossing brackets etc. Where *labelled* precision/recall metrics are required, which are sensitive to category labels, the gold standard parse trees in the treebank are relabelled under the SPC mapping prior to comparison. A more detailed desciption of this part of our work may be provided for the final version of the paper.

5 Conclusion

In the present paper we have analysed (some of) the formal properties of SPC, contrasted it with those of RPC and conducted a number of compaction experiments on a large grammar (# rules > 17000) extracted from the Penn II treebank. The results are the following:

In contrast to RPC and THC, SPC is structure preserving in the following sense: for each tree defined by (or rule in) the original grammar it generates a tree (rule) isomorphic up to node relabelling. Both RPC and SPC change strong generative capacity. RPC maintains week generative capacity while SPC can extend week generative capacity in the compacted grammar. Linguistic expertise is required so that the grammar is not undermined in this way. RPC can only reduce ambiguity, SPC can both reduce and increase ambiguity in the compacted grammar. SPC can conflate structures differing only in node labeling. It cannot conflate configurationally distinct structures.

SPC provides but a partial answer to rule accession rates, as shown in our experiments. Plugging holes in a treebank grammar contributes to reducing rule growth rates but is one factor among many.

In summary: SPC is a simple and intuitive grammar compaction technique. It is both a stricter and looser form of compaction than RPC. It is stricter in that it preserves linguistically motivated structure and looser in that it can extend weak generative capacity. It can be a valuable technique for treebanks with large and discriminating tag and category sets (such as the AP and Suzanne tree banks).

References

E. Charniak. Tree-bank grammars. In AAAI-96, Proceedings of the Thirteenth National Conference on Artificial Intelligence, MIT Press, pp.1031-1036, 1996.

R.M. Kaplan and J. Bresnan. Lexical Functional Grammar. In Bresnan, J., editor 1982, *The mental representation of grammatical relations.* MIT Press, Cambridge Mass. 173–281, 1982.

A. Krotov, M. Hepple, R. Gaizauskas and Y. Wilks. Compacting the Penn Treebank Grammar. In COLING-ACL'98, Montreal, Canada, pp.699-703, 1998.

A. Krotov, M. Hepple, R. Gaizauskas and Y. Wilks. 1999. Evaluating two Methods for Treebank Compaction. *Submitted to Journal of Natural Language Engineering*

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

J. van Genabith, L. Sadler and A. Way, Structure Preserving cf-psg compaction, lfg and treebanks. In *Proceedings ATALA Workshop - Treebanks*, Journees ATALA, Corpus annotes pour la syntaxe Universite Paris 7, France, 18-19 Juin 1999, pp. 107-114

J. van Genabith, A. Way and L. Sadler. Semi-Automatic Generation of F-Structures from Tree Banks. LFG99, The fourth International Conference on LFG. Manchester, U.K., 19-21 July 1999.