Optimisation of corpus-derived probabilistic grammars

Anja Belz
CCSRC, SRI International
23 Millers Yard, Mill Lane
Cambridge CB2 1RQ, UK

1 Overview

This paper examines the usefulness of corpus-derived probabilistic grammars as a basis for the automatic construction of grammars optimised for a given parsing task. Initially, a probabilistic context-free grammar (PCFG) is derived by a straightforward derivation technique from the Wall Street Journal (WSJ) Corpus, and a baseline is established by testing the resulting grammar on four different parsing tasks. In the first optimisation step, different kinds of local structural context (LSC) are incorporated into the basic PCFG. Improved parsing results demonstrate the usefulness of the added structural context information. In the second optimisation step, LSC-PCFGs are optimised in terms of grammar size and performance for a given parsing task. Tests show that significant improvements can be achieved by the method proposed.

The structure of this paper is as follows. Section 2 discusses the practical and theoretical questions and issues addressed by the research presented in this paper, and cites existing research and results in the same and related areas. Section 3 describes how LSC-grammars are derived from corpora, defines the four parsing tasks on which grammars are tested, describes data and evaluation methods used, and presents a baseline technique and baseline results. Section 4 discusses and describes different types of LSC and demonstrates their effect on rule probabilities. Methods for deriving four different LSC-grammars from the corpus are described, and results for the four parsing tasks are presented. It is shown that all four types of LSC investigated improve results, but that some lead to overspecialisation of grammars. Section 5 shows that LSC-grammars can be optimised for grammar size by a generalisation technique that at the same time seeks to optimise parsing performance for a given parsing task. An automatic search method is described that carries out a search for optimal generalisations of the given grammar in the space of partitions of nonterminal sets. First results are presented for the automatic search method that show that it can be used to reduce grammar size and improve parsing performance.

Parent node information is shown to be a particularly useful type of LSC, and the results for the complete parsing task achieved with the corresponding grammar are better than any previously published results for comparable unlexicalised grammars. Preliminary tests for LSC grammar optimisation show that it can drastically reduce grammar size and significantly improve parsing performance. In one set of experiments, a partition was found that increased the labelled F-Score for the complete parsing task from 72.31 to 74.61, while decreasing grammar size from 21,995 rules and 1,104 nonterminals to 11,254 rules and 224 nonterminals. Results for grammar optimisation by automatic search of the partition space show that improvements in grammar size and parsing performance can be achieved in this way, but do not come close to the big improvements achieved in preliminary tests. It is concluded that more sophisticated search techniques are required to achieve this.

2 Background and related research

The research reported in this paper covers a range of issues: (i) corpus-derived grammars; (ii) the usefulness of structural context information in making parsing decisions; (iii) automatic construction methods for specialised grammars that take corpus-derived grammars as a starting point; (iv) the (in)adequacy of PCFGs as a grammar formalism; and (v) the question of whether parsing strategies that do without lexical information can come closer to the performance of lexicalised systems. Each of these issues will be discussed in more detail over the following sections.

Corpus-derived grammars. Over the last five years, a range of research projects — e.g. Charniak (1996), Cardie & Pierce (1998), Johnson (1998, 2000), Krotov et al. (2000) — have looked at probabilistic grammars that have been directly derived from bracketted corpora (or treebanks, hence the term

"treebank grammar" coined by Charniak, 1996). The basic idea in grammar derivation from corpora is simple. For each distinct bracketting found in a corpus, a grammar rule is added to the grammar and the rule's probability is derived in some way (often by maximum-likelihood estimation with some smoothing method) from the frequency of occurrence of the bracketting in the corpus. For instance, the bracketting `(NP (DT the) (NN tree))` would yield the production rule `NP` $\rightarrow$ `DT NN`.

However, because the number of rules in grammars derived in this entirely straightforward manner is infeasibly large at least in the case of the WSJ Corpus, and because their parsing performance moreover tends to be poor, some techniques are usually applied to reduce grammar size and to improve performance. All approaches edit the corpus in some way, e.g. eliminating single child rules, empty category rules, functional tags, co-indexation tags, and punctuation marks. Different compaction methods (such as eliminating rules with frequency less than some n) have been investigated that reduce the size of grammars without too much loss of performance (in particular by Charniak and Krotov et al.). To improve parsing performance, e.g. Charniak relabels auxiliary verbs with a separate POS-tag and incorporates a "right-branching correction" into the parser to make it prefer right-branching structures.

As a result of such techniques, the final grammars for which performance results are reported tend to have little in common with the rule set underlying the corpus from which they were derived.

Several other grammar building and training methods are similar to treebank grammar construction: Bod & Scha's DOP[1] method which extracts tree fragments rather than rules from corpora, MBL[2] methods (Daelemans et al.) for building parsing systems from corpora, and — more generally — any method that estimates the likelihood of brackettings (or of brackettings converted into taggings) from a corpus, since such methods directly utilise both the brackettings and their frequencies as found in the corpus.

The existing results for corpus-derived grammars that do not undergo significant further development demonstrate their limitations: they cannot compete with state-of-the-art parsing results (see Section 2). It will be argued in this paper that grammars directly extracted from corpora do, however, provide a useful starting point for further automatic grammar construction methods.

Context-free grammars that incorporate structural context. It is frequently observed (e.g. Manning & Schütze (1999, p. 416ff)) that PCFGs are inadequate as a grammar formalism because of the very strong independence assumptions inherent in them, reflecting on the one hand a complete lack of lexicalisation, and on the other a lack of structure dependence.

It is true that in conventional PCFGs the probability of, say, a given NP bracketting is independent of the identity of the head noun as well as its structural context (e.g. whether the NP is in subject or object position). However, this independence is not due to the formal characteristics of PCFGs, but rather to the way they tend to be used. If the set of nonterminals of a PCFG does not distinguish between, say, NPs in subject position and NPs in object position, then the probabilities of any rules containing the nonterminal NP are necessarily independent of the subject/object distinction.

However, it is straightforward to introduce such a dependence into a PCFG by splitting the category NP into two categories NP-SBJ and NP-OBJ. Similarly, categories (nonterminals) can be divided on the basis of lexemes, lexical categories or semantic classes.

PCFGs may not be able to accommodate lexical and structural information in the most elegant fashion, but the point here is not about representational elegance and efficiency. Rather, the fact that PCFGs encode languages that make up the formal class of context-free languages is entirely separate from their ability to reflect the dependence of rule probabilities on lexical and structural context.

Examining different kinds of structural context within the PCFG framework (as done in this paper) has two advantages: firstly, there are polynomial-time algorithms for finding most likely parses, and secondly, there is a simple measure of the complexity added to a grammar by the introduction of a piece of structural information such as the subject/object distinction, namely the resulting increase in the number of rules in the grammar.

Automatic grammar construction. It is sometimes observed that deriving probabilistic grammars from corpora in the way described above is not an automatic grammar learning method because all that is done is to extract the PCFG that underlies the corpus and is encoded in its sentences, brackettings and occurrence frequencies. As was pointed out above, creating a grammar in this way is simply one of many ways to utilise the brackettings and frequencies of corpora, a feature shared with many computational learning approaches to automatic grammar construction. However, as previously mentioned, the limitations of grammars directly extracted from corpora indicate that using them as starting points for further grammar development is the more useful approach.

[1] Data-Oriented Parsing.
[2] Memory-Based Learning.

Grammar/Parser	Grammar Size	performance (WSJ unseen)				
		LR	UR	LP	UP	CB
Fully lexicalised:						
Collins (2000)	–	90.1	–	90.4	–	0.73
Charniak (2000)	–	90.1	–	90.1	–	0.74
Collins (1999)	–	88.5	–	88.7	–	0.92
Collins (1997)	–	88.1	–	88.6	–	0.91
Charniak (1997)	–	87.5	–	87.4	–	1.00
Magerman (1995) SPATTER	–	84.6	–	84.9	–	1.26
Nonlexicalised:						
Charniak (1996)	10,605	–	78.8	–	80.4	–
without frequency 1 rules	3,943	–	78.2	–	80.7	–
Krotov et al. (2000)	15,420	74.1	77.1	77.4	80.6	2.13
without frequency 1 rules	6,514	74.4	77.5	76.9	80.2	2.18
WSJ 15–18 treebank PCFG	6,135	69.1	–	71.4	–	2.67

Table 1: Performance of comparable lexicalised and nonlexicalised grammars on full parsing.

Reference	Method	LP	LR	F-Score
Lexicalised:				
Tjong Kim Sang et al. (2000)	System combination	94.2	93.6	93.9
Muñoz et al. (1999)	SNoW	92.4	93.1	92.8
XTAG Research Group (1998)	XTAG + Supertagging	91.8	93.0	92.4
Ramshaw & Marcus (1995)	Transformation Based Learning	91.8	92.3	92.0
Veenstra (1998)	MBL	89.0	94.3	91.6
Nonlexicalised:				
Argamon et al. (1999)	MBL	91.6	91.6	91.6
Cardie & Pierce (1998)	Error-Driven Grammar Pruning	90.7	91.1	90.9
	WSJ 15–18 treebank PCFG	89.2	87.6	88.4

Table 2: Performance of comparable lexicalised and nonlexicalised grammars on NP-chunking.

Creating a starting point for grammar learning in this way is particularly useful because context-free grammars cannot be learnt from scratch from data. At the very least, an upper bound must be placed on the number of nonterminals allowed. Even when that is done, there is no likelihood that the grammars resulting from an otherwise unsupervised method will look anything like a linguistic grammar whose parses can provide a basis for semantic analysis[3].

Parsing with(out) lexical information. Corpus-derived grammars tend to be nonlexicalised PCFGs, hence the existing research cited above can be seen as investigations into the results that can be achieved in parsing without taking into account lexical information.

In syntactic parsing tasks, nonlexicalised methods are generally outperformed by lexicalised approaches. In the case of complete (non-shallow) parsing, nonlexicalised methods are outperformed by large margins. Table 1 shows an overview of state-of-the-art nonlexicalised and lexicalised results for statistical parsing systems (U/LR = Un-/Labelled Recall, U/LP = Un-/Labelled Precision, see Section 3). For comparison, the last row of the table shows this paper's baseline result for the complete parsing task (see Section 3.4).

In NP-chunking, a shallow syntactic parsing task that has become a popular research topic over the last decade (for details see Section 3.2 below), nonlexicalised systems also tend to lag behind lexicalised ones, although by much smaller margins. Table 2 shows a range of results for the baseNP chunking task and data set given by Ramshaw & Marcus (1995). Again, the corresponding baseline result from this paper is included in the last row. It is clear from this overview that the difference between lexicalised and unlexicalised systems is far smaller for this parsing task than for complete parsing.

There are several reasons for investigating how well parsers can do without lexicalisation. Apart

[3]Any linguistic CFG can be converted into a normal form that encodes the same set of sentences, but whose derivations and substructures are not semantically meaningful.

from the theoretical interest, optimising grammars before adding lexicalisation may improve their overall performance, as lexicalised systems often perform worse than comparable nonlexicalised systems when the lexical component is taken out. E.g. Collins (1996) includes results for the system with lexical information removed, which reduces LR from 85.0 to 76.1 and LP from 85.1 to 76.6 in one test – worse than the comparable results reported below in Section 4.3 (78.78 and 77.16). Furthermore, the results shown in Tables 1, 2 and 4 indicate that shallow parsing tasks require lexical information to a far lesser extent than nonshallow ones, so that the added expense of lexicalisation might be avoidable in the case of such tasks.

3 Grammars, parsing tasks, data and evaluation

3.1 Grammars from corpora

The basic procedure used for deriving PCFGs from WSJ Sections 15–18 can be summarised as follows[4]:

1. In the first step, the corpus is iteratively edited by deleting (i) brackets and labels that correspond to empty category expansions; (ii) brackets and labels containing a single constitutent that is not labelled with a POS-tag; (iii) cross-indexation tags; (iv) brackets that become empty through a deletion; and (v) functional tags.
2. In the second step, each remaining bracketting in the corpus is converted into a production rule. The rules are divided into nonlexical ones (those that form the grammar), and lexical ones (those that form the lexicon).
3. In the final step, a complete PCFG is created. The set of lexical rules is converted into a lexicon with POS-tag frequency information. The set of nonterminals is collected from the set of rules. Each set is sorted, the number of times each item occurs is determined, and duplicates are removed. Probabilities P for rules $N \rightarrow \zeta$ are calculated from the rule frequencies C by Maximum Likelihood Estimation (MLE): $P_{MLE}(N \rightarrow \zeta) = \frac{C(N \rightarrow \zeta)}{\sum_i C(N \rightarrow \zeta^i)}$

3.2 Four parsing tasks

Results are given in the following and subsequent sections for four different parsing tasks:

1. *Full parsing:* The task is to assign a complete parse to the input sentence. A full parse is considered 100% correct if it is identical to the corresponding parse given in the WSJ Corpus.
2. *Noun phrase identification:* The task is to identify in the input sentence all noun phrases, nested and otherwise, that are given in the corresponding WSJ Corpus parse.
3. *Complete text chunking:* This task was first defined in Tjong Kim Sang & Buchholz (2000), and involves dividing a sentence into flat chunks of 11 different types. The target parses are derived from WSJ parses by a deterministic conversion procedure.
4. *Base noun phrase identification:* First defined by Abney (1991), this task involves the recognition of non-recursive noun phrase chunks (so-called baseNPs). Target parses are derived from WSJ parses by a simple conversion procedure.

3.3 Data and evaluation

Sections 15–18 of the Wall Street Journal (WSJ) corpus were used for grammar derivation, and Section 01 from the same corpus was used for testing parsing performance. Parsing performance was tested with the commonly used `evalb` program by Sekine & Collins[5]. The program evaluates parses in terms of the standard PARSEVAL evaluation metrics *Precision, Recall* and *crossing brackets*. For a parse P and a corresponding target parse T, Precision measures the percentage of brackets in P that match the target brackettings in T. Recall is the percentage of brackettings from T that are in P. Crossing brackets gives the average number of constituents in one parse tree that cross over constituent boundaries in the other tree. (See e.g. Manning & Schütze (1999), p. 432–434.)

For Precision and Recall there are *unlabelled* and *labelled* variants. In the latter, both the pair of brackets and the constituent label on the bracket pair have to be correct for the bracketting to be correct,

[4]Throughout this paper, WSJ refers to the PENN II Treebank version.
[5]Available from `http://cs.nyu.edu/cs/projects/proteus/evalb/`.

whereas in the unlabelled variant only the brackets have to be correct. In this paper, unless otherwise stated, Precision and Recall always mean Labelled Precision and Recall, in particular, all new results presented are the labelled variants. Precision and Recall are commonly combined into a single measure, called F-Score, given by $(\beta^2+1)\times Precision \times Recall/\beta^2(Precision + Recall)$. In this paper, $\beta = 1$ throughout.

All grammars tested are nonlexicalised, therefore input sentences are sequences of POS-tags not words. In the tests, sentences of a length above 40 words (consistently close to 7.5% of all sentences in a corpus section) were left out. All grammars are formally probabilistic context-free grammars (PCFGS). The parsing package LoPar (Schmid (2000)) was used to obtain Viterbi parses for data sets and grammars. If LoPar failed to find a complete parse for a sentence, a simple grammar extension method was used to obtain partial parses instead.

3.4 Baseline

A baseline grammar "*BARE*" was extracted from WSJ Sections 15–18 by the method described in Section 3.1, applied to the four parsing tasks defined in Section 3.2, and tested and evaluated as desribed in the preceding section. This yielded the following set of results which forms the baseline for the purpose of this paper. (Results include 9 partial parses.)

Full parsing			NP identification			BaseNP chunking			Complete text chunking		
LR	LP	F	LR	LP	F	LR	LP	F	LR	LP	F
69.08	71.43	70.24	74.97	81.62	78.15	87.6	89.21	88.4	89.63	88.99	89.31

4 Introducing structural context into PCFGs

4.1 Different types of structural context

In this section, the effects of introducing three different types of structural context (SC) into PCFG *BARE* are examined: (i) the grammatical function of phrases, (ii) their depth in the parse tree, and (iii) the category of the parent phrase. All three types of structural context are local to the immediate neighbourhood of the phrase node for which they provide the expansion probability conditions. Other local SC types that could be considered include position among the children of the parent node, and identity of immediate sibling nodes. Useful nonlocal SC types might be the identity of more distant ancestors than the parent node and of more distant sibling nodes.

Grammatical function. As mentioned above, the WSJ corpus subdivides standard phrase categories such as `NP` by attaching functional tags to them that reflect the grammatical function of the category, e.g. `NP-SBJ` and `NP-OBJ`. However, the corpus is not consistently annotated in this fashion (the same type of phrase may have zero, one or more functional tags). Parsing results for grammar *FTAGS* might be better if the grammar is derived from a more consistently annotated corpus.

The rule that expands a noun phrase to a personal pronoun is a strong example of the extent to which grammatical function can affect expansion probabilities. In the WSJ, 13.7% of all `NP`s expand to `PRP` as subject, compared to only 2.1% as object. Of all object `NP`s, 13.4% expand to `PRP` as first object, compared to 0.9% as second object (source: Manning & Schuetze, 1999. p. 420).

Depth of embedding. The depth of embedding of a phrase is determined as follows. The outermost bracketting (corresponding to the top of the parse tree) is at depth 1, its immediate constituents are at depth 2, and so on. In the parsed sentence `(S (NP (DT the) (NN cat)) (VP (VBD sat) (PP (IN on) (NP (DT the) (NN mat)))))`, `S` is at depth 1, the first occurrences of `NP` and `VP` are at depth 2, the first occurrences of `DT` and `NN` as well as `VBD` and `PP` at depth 3, `IN` and the second `NP` at depth 4, and the second occurrences of `DT` and `NN` are at depth 5.

It is not obvious that the depth of embedding of a phrase captures linguistically meaningful parts of its local structural context. However, different phrases of the same category do occur at certain depths with higher frequency than at others. This is most intuitively clear in the case of NPs, where subject NPs occur at depth 2, whereas object NPs occur at lower depths.

More surprisingly, VPs too have preferences for occurring at certain levels. Table 3 (previously shown in Belz (2000, p. 49)) provides clear evidence of this. The first column shows the six most frequent WSJ VP expansion rules, the second column shows their canonical probabilities (calculated over all WSJ VP rules). The remaining columns show how these probabilities change if they are made conditional on depths of embedding 2–7. For each depth, the highest rule probability is highlighted in boldface font,

		Depth of Embedding					
		2	3	4	5	6	7
p(VP → TO VP)	**0.089**	0.004	**0.067**	**0.136**	**0.127**	**0.135**	**0.130**
p(VP → MD VP)	*0.056*	*0.075*	0.043	*0.055*	0.062	0.050	0.047
p(VP → VB NP)	0.054	0.001	0.036	0.052	*0.073*	*0.088*	*0.096*
p(VP → VBN PP)	0.039	0.004	*0.049*	0.047	0.042	0.044	0.055
p(VP → VBZ VP)	0.038	0.069	0.034	0.037	0.025	0.023	0.021
p(VP → VBD S)	0.026	**0.090**	0.016	0.005	0.005	0.004	0.003

Table 3: Rule probabilities at different depths of embedding for 6 common VP rules.

and the second highest in italics. At depth 2, for instance, the most likely rule is the one with the fourth highest canonical probability, and at depth 5, the second most likely rule is the one with the third highest canonical probability. In fact, there is only one depth (4) at which rule probablities appear in their canonical order, which shows how strongly even VP rules are affected by depth of embedding.

Parent node. The parent node of a phrase is the category of the phrase that immediately contains it. In `(S (NP (DT the) (NN cat)) (VP (VBD sat) (PP (IN on) (NP (DT the) (NN mat))))) S` is the parent of `NP` and `VP`, `VP` is the parent of `PP`, which is the parent of `NP`. Thus, distinguishing between `NP-S` (an `NP` with `S` as its parent) and `NP-PP` captures part of the subject/object distinction.

The advantage of using parent node information was previously noted by Johnson[6] (1998).

4.2 Four LSC-Grammars

Grammars incorporating local structural context — or LSC grammars — were extracted from the corpus by the same procedure as described in Section 3.1 above, except that during Step 2, each bracket label that is not a POS tag was annotated with a tag representing the required type of LSC.

Four different grammars were derived in this way, PCFGs *FTAGS*, *DOE*, *PN* and *DOEPN*. All four grammars incorporate the functional tags present in the WSJ Corpus. Additionally, for grammar *DOE*, each nonterminal was annotated with a tag representing the depth of embedding at which it was found, for grammar *PN*, nonterminals were annotated with tags encoding their parent node, and for grammar *DOEPN*, nonterminals were given both depth and parent node tags. The resulting grammars are significantly larger than the baseline grammar *BARE*. Grammar sizes and numbers of nonterminals (excluding POS tags) are as follows:

Grammar Type	*BARE*	*FTAGS*	*DOE*	*PN*	*DOEPN*
Size (n rules)	6,135	10,118	21,995	16,480	33,101
Nonterminals	26	147	1,104	970	4,015

4.3 Performance on parsing tasks

In calculating labelled bracketting Recall and Precision for the LSC-grammar results, all labels starting with the same category prefix, e.g. NP, are considered equivalent (standard in `evalb`). The idea is that the additional information encoded in the LSC-tags attached to category labels helps select the correct parse, not that it should be retained in the annotation for further analysis. Table 4 shows parsing results for the unseen data in WSJ Section 01 (the results for baseline grammar *BARE* are also included for comparison). Best F-Scores are highlighted in boldface font, and second-best F-Scores in italics.

The best results in Table 4 are better than those reported by Charniak (1996) and Krotov et al. (2000), even though the previous results were obtained after using ca. 10/11 of the WSJ corpus as a training set (compared to 3/25 used here):

	UF	LF
Krotov et al. (2000)	79.12	76.09
Charniak (1996)	79.59	–
PN-Grammar	80.51	77.96

[6] Johnson calls it grandparent node, but means the same thing.

Grammar Type	*BARE*	*FTAGS*	*DOE*	*PN*	*DOEPN*
Partial parses	9	9	25	20	62
Full parsing:					
LR	69.08	71.41	72.72	78.78	74.33
LP	71.43	73.06	71.9	77.16	70.61
F-Score	70.24	72.23	72.31	**77.96**	*72.42*
Crossing brackets	2.76	2.51	2.53	1.91	2.59
% 0 CBs	32.34	35.43	35.75	44.40	37.0
NP identification:					
LR	74.97	77.22	78.2	83.86	81.02
LP	81.62	81.02	77.56	81.22	74.30
F-Score	78.15	*79.07*	77.88	**82.52**	77.51
BaseNP chunking:					
LR	87.6	87.35	87.02	90.27	87.05
LP	89.21	88.68	87.03	89.52	84.11
F-Score	*88.4*	88.01	87.02	**89.89**	85.55
Complete text chunking:					
LR	89.63	89.49	89.17	90.84	89.24
LP	88.99	88.64	87.28	89.46	85.85
F-Score	*89.31*	89.06	88.21	**90.14**	87.51

Table 4: Parsing results for the four LSC-grammars and WSJ Section 01.

Incorporating different types of LSC affects results for the four parsing tasks in different ways. It is clear from the results in Table 4 that some kinds of contextual information are useful for some tasks but not for others. For example, adding parent phrase information improved results (from grammar *BARE* to grammar *PN*) by almost 8 points (F-Score 70.24 to 77.96) for the complete parsing task, by about 4.5 points (F-Score 78.15 to 82.52) for NP identification, by 1.5 points (F-Score 88.4 to 89.89) for baseNP chunking, and by just under one point (F-Score 89.31 to 90.14) for complete text chunking.

It is likely that adding depth of embedding information indiscriminately (as in grammars *DOE* and *DOEPN*) results in overspecialisation. Looking at results for seen data (part of the training corpus) confirms this. Table 5 shows results for the baseline grammar and the four LSC grammars on WSJ Section 15, i.e. one of the sections used during grammar derivation. On seen data, grammar *DOEPN* performs best on all parsing tasks. Tables 4 and 5 together imply that adding depth of embedding information for all depths to all rules simply overfits the training data and results in undergeneralisation.

Similarly, it is likely that not all the information added in the four LSC grammars is useful for all parsing tasks. Distinguishing 27 depths of embedding is probably too much for all parsing tasks, e.g. distinguishing depths above 20 is generally unlikely to be useful, as the occurrence of rules at such depths is rare. Techniques for eliminating the information that makes no useful contribution for a given parsing task are discussed in the following section.

5 Automatic optimisation of LSC-Grammars

5.1 Initial assumptions

If it is true that some of the LSC information added to the grammars tested so far makes little or no contribution to a grammar's performance on a given parsing task, then it should be possible to reduce grammar size without loss of parsing performance by selectively taking out some of the added information. At the same time, if it is true that some of the LSC-grammars are overspecialised (overfit the data), then it should be possible to improve the grammar's performance by selectively generalising them.

As pointed out above in Section 4.3, it is clear from the LSC results that adding different kinds of LSC information to a grammar has different effects on the results for different parsing tasks. It should therefore be possible to optimise a grammar for a given parsing task by selectively taking out the information that is not useful for the given task. The idea behind the experiments reported in the following section was to see to what extent the LSC grammars can be optimised in terms of size and parsing performance by grammar partitioning for each of the parsing tasks.

Grammar Type	*BARE*	*FTAGS*	*DOE*	*PN*	*DOEPN*
Partial parses	0	0	0	0	0
Full Parsing:					
LR	71.48	75.15	82.81	84.64	90.39
LP	75.03	78.64	84.86	85.94	91.43
F-Score	73.21	76.86	83.82	*85.29*	**90.91**
Crossing brackets	2.57	2.15	1.37	1.31	0.75
% 0 CBs	34.48	41.85	56.31	57.33	73.46
NP identification:					
LR	76.54	79.26	84.51	87.46	91.17
LP	84.89	85.61	88.79	88.75	92.61
F-Score	80.5	82.31	86.6	*88.1*	**91.88**
BaseNP chunking:					
LR	90.21	90.28	92.68	94.40	95.99
LP	92.59	92.70	94.54	95.66	97.19
F-Score	91.38	91.47	93.60	*95.03*	**96.59**
Complete text chunking:					
LR	91.68	91.67	93.59	94.25	96.45
LP	92.46	92.56	94.19	95.02	96.84
F-Score	92.07	92.11	93.89	*94.63*	**96.64**

Table 5: Parsing results for the LSC-grammars and WSJ Section 15 (seen data).

5.2 Preliminary definitions

The addition of structural context as described in previous sections can be viewed in terms of split operations on nonterminals, e.g. in the *FTAGS* grammar, the nonterminal `NP` is split into `NP-SUBJ` and `NP-OBJ` (among others). This results in grammar specialisation, i.e. the new grammar parses a subset of the set of sentences parsed by the original one. The reverse, replacing `NP-SUBJ` and `NP-OBJ` with a single nonterminal `NP`, can be seen as a merge operation, and results in grammar generalisation, i.e. the new grammar parses a superset of the sentences parsed by the original one.

An arbitrary number of such merge operations can be represented by a partition on the set of nonterminals of a grammar. A partition is defined as follows.

Definition 1 Partition

A partition of a nonempty set A is a subset Π of 2^A such that $\emptyset$ is not an element of Π and each element of A is in one and only one set in Π.

PCFGs can be defined as follows.

Definition 2 Probabilistic Context-Free Grammar (PCFG)

A PCFGs is a 4-tuple (W, N, N^S, R), where W is a set of terminal symbols $\{w_1, \ldots w_u\}$, N is a set of nonterminal symbols $\{n_1, \ldots n_v\}$, $N^S \subseteq N$ is a set of start symbols $\{n_1^s, \ldots n_v^s\}$, und R is a set of rules with associated probabilities $\{(r_1, p(r_1)), \ldots (r_1, p(r_x))\}$. Each rule r is of the form $n \rightarrow \alpha$, where α is a sequence of terminals and nonterminals. For each nonterminal n, the values of all $p(n \rightarrow \alpha_i)$ sum to one.

Given a PCFG $G = (W, N, N^S, R)$ and a partition $\Pi_N = \{N_1, \ldots N_v\}$ of the set of nonterminals N, the partitioned PCFG $G' = (W, N', N^{S'}, R')$ is derived by the following procedure:

1. Assign a new nonterminal name to each of the non-singleton elements of Π_N.
2. For each rule r_i in R, and for each nonterminal n_j in r_i, if n_j is in a non-singleton element of Π_N, replace it with the corresponding new nonterminal.
3. Find all rules in R of which there are multiple occurrences as a result of the substitutions in Step 2, sum their frequencies and recalculate the rule probabilities.

If start symbols are permitted to be merged with non-start symbols, then there are two ways of determining the probability of a rule expanding the nonterminal resulting from such a merge: either its frequency is the sum of the frequencies of all nonterminals in the merge set, or it is the sum of just the frequencies of the start symbols in the merge set. The latter option was chosen in the tests reported below.

5.3 "Proof of concept"

The discussion and results in this section provide preliminary confirmation of the prediction made in Section 4.3 that for the different LSC grammars there exist (non-trivial) partitions that outperform the original base grammar. More formally, the "proof of concept" provided below shows the following for most of the grammar/task combinations:

> Given a base grammar $G = (W, N, N^S, R)$ and a parsing task P, a partition of the set of nonterminals N can be found such that the derived grammar $G' = (W, N', N^{S'}, R')$
>
> 1. is smaller than G (i.e. $|R'| < |R|$), and
> 2. performs better than G on P.

Some of the five LSC-PCFGs can be derived by partition from one of the others. For example, *BARE* can be derived from all others, *FTAGS* can be derived from *DOE*, *PN* and *DOEPN*, and *DOE* and *PN* can both be derived from *DOEPN*. This means that for some of the grammars, the results given in Section 4.3 in themselves show that there exists at least one (non-trivial) partition that is smaller than and outperforms the original grammar. E.g. for the baseNP chunking task, the partition that derives *PN* from *DOEPN* achieves nearly a 3 point improvement (F-Score 87.63 to 90.23), while reducing grammar size from 33,101 rules to 16,480, and the number of non-terminals from 4,015 to 970.

In the remainder of this section it is shown that there are other partitions of the *DOE* grammar that improves its performance and reduces its size.

Grammar type	Depth bands	Grammar size	Nonterminals
DOE	*1, 2, ... 27*	21,995	1,104
	1, 2, 3, rest	12,933	312
	1, 2, rest	11,254	224
	1, rest	10,165	170
FTAGS	–	10,118	147
BARE	–	6,135	26

Table 6: Sizes and depth bands of *DOE* grammar and 5 of its partitions.

From the parsing results for the *DOE* grammar it appears that indiscriminately adding depth of embedding information does not help improve parsing performance for shallow parsing tasks on unseen data: while there is a significant improvement for the complete parsing task (F-Score 70.24 to 72.31), the F-Scores for the other three parsing tasks are worse. That there is any improvement shows that some useful information is added. It is likely that distinguishing all depths simply leads to overspecialisation of the grammar, resulting in a large increase in parse failures on the one hand, and the selection of bad, previously unlikely, parses on the other. If this is so then partitioning the *DOE* grammar in a way equivalent to distinguishing broader depth bands rather than each individual depth will improve results.

To test this hypothesis, three different partitions of the *DOE* grammar were created. The partitions (too large to be shown in their entirety) correspond to distinguishing between the different depths shown in the second column of Table 6, e.g. in the case of the fourth row, all nonterminals NT-*n* with a depth tag n greater than 1 are merged into a single nonterminal NT-*rest*. The last two columns show the number of rules and nonterminals in each grammar. The last two rows show the corresponding numbers for the *BARE* and *FTAGS* grammars (*DOE*-type grammars all incorporate functional tags).

The partitioned *DOE* grammars all improve results (compared to grammars *BARE*, *FTAGS*, and *DOE*) for the full parsing task, with the *DOE*-{1, 2, rest} grammar performing the best. For the NP identification task, grammar *DOE* achieved a worse F-Score than grammar *BARE*, yet all the partitioned *DOE* grammars achieve a better F-Score than grammar *BARE*, with the *DOE*-{1, 2, rest} grammar again performing the best. On the baseNP chunking task and the complete text chunking task, grammar *BARE* performs the best, but all the derived *DOE* grammars outperform the nonpartitioned *DOE* grammar. On

Grammar Type	*BARE*	*FTAGS*	*DOE*	*DOE*-{1,r}	*DOE*-{1,2,r}	*DOE*-{1, 2, 3, r}
Full Parsing:						
LR	69.08	71.41	72.72	73.35	74.13	74.14
LP	71.43	73.06	71.9	74.48	75.1	74.73
F-Score	70.24	72.23	72.31	73.91	**74.61**	*74.43*
Crossing brackets	2.76	2.51	2.53	2.32	2.19	2.21
% 0 CBs	32.34	35.43	35.75	38.83	39.1	38.56
NP identification:						
LR	74.97	77.22	78.2	77.39	77.95	78.29
LP	81.62	81.02	77.56	81.20	81.31	80.85
F-Score	78.15	79.07	77.88	79.25	**79.59**	*79.55*
BaseNP chunking:						
LR	87.6	87.35	87.02	87.82	87.73	87.74
LP	89.21	88.68	87.03	88.93	88.59	88.11
F-Score	**88.4**	88.01	87.02	*88.37*	88.16	87.93
Complete text chunking:						
LR	89.63	89.49	89.17	89.58	89.71	89.70
LP	88.99	88.64	87.28	88.54	88.52	88.29
F-Score	**89.31**	89.06	88.21	89.06	*89.11*	88.99

Table 7: Parsing results of *DOE* grammar and 5 of its partitions.

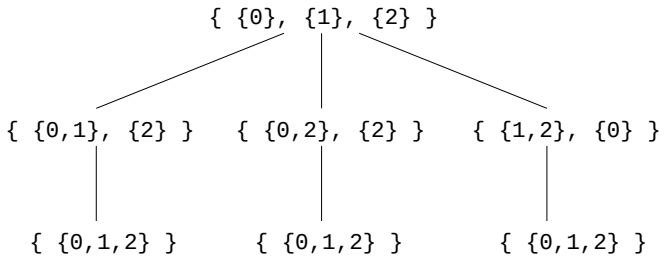

Figure 1: Partition tree for a set with three elements.

the baseNP chunking task, the *BARE* grammar's F-Score is closely matched by the *DOE*-{1, rest} grammar. These results show that partitions can be found that not only drastically reduce grammar size but also significantly improve parsing performance on a given parsing task.

5.4 Search for optimal partition of LSC-Grammars

Given. A PCFG $G = (W, N, N^S, R)$, a data set D, and a set of target parses D^T for D.

Search space. The search space is defined as the partition tree for the set of nonterminals N in the given grammar G. Each node in the tree is one of the partitions of N, such that each node's partition has fewer elements than all of its ancestors, and the partition at each node can be derived from its parent by merging two elements of the parent's partition.

The single node at the top of the tree is the trivial partition corresponding to N itself. Each node is the parent of $\frac{1}{2}(n^2 - n)$ child nodes, where n is the number of elements in the parent partition. Each level reduces the number of states by one. The complete partition tree for a set with three elements looks as shown in Figure 1.

Search method. The partition tree is searched top-down by a variant of beam search. A list of the n current best candidate partitions is maintained (initialised to the trivial partition). For each of the n current best partitions a subset of size b of its children in the partition tree is generated and evaluated (b thus defines the width of the beam). From the set of current best partitions and the newly generated candidate partitions, the n best elements are selected and form the new current best set. This process is iterated until either no new partitions can be generated that are better than their parents, or the lowest level of the partition tree is reached.

In the current version of the evaluation function, only the F-Score achieved by candidate solutions on the test data is taken into account. Search stops if in any iteration (depth of the partition tree) no

solution is found that outperforms the current best solutions. That is, size is not explicitly evaluated at all. Candidate solutions are evaluated on a subset of the test data, because evaluating each candidate solution on all 1,993 sentences of WSJ Section 01 makes the cost of the search procedure prohibitive.

There are three variable parameters in the partition tree search procedure: (i) the number n partitions (nodes in the tree) that are further explored, (ii) the size x of the subset of the test data that candidate solutions are evaluated on, and (ii) the width b of the beam.

5.5 Results for LSC-Grammar optimisation by search of partition tree

Table 8 shows some results for automatic optimisation experiments carried out for grammar *PN* and the baseNP chunking and complete text chunking tasks. The first three columns show the variable parameters b (beam width), n (size of list of best solutions maintained), and x (size of data subset used in evaluation). The fourth column shows the number of runs results are averaged over, and the fifth and sixth columns show the number of iterations and evaluations carried out before search stopped. Column 7 gives the average number of nonterminals the best solution grammars had, and column 8 their average evaluation score. The last two columns show the overall change in F-Score (calculated on all of WSJ Section 01) and grammar size for the given grammar and parsing task.

Var. Parameters			*Runs*	*Iter.*	*Eval.*	*Nonterms*	*F-Score (sub)*	*F-Score +/-*	*Size +/-*
b	n	x							
Grammar: *PN*; Grammar Size: 16,480/970 Task: BaseNP chunking; F-Score: 89.89									
100	2	50	4	4	45	968.25	95.93	+0.032 (89.92)	-0.25
100	10	50	4	6.75	341.5	967.25	97.25	+0.048 (89.94)	-2
500	1	50	4	5.25	499	967.5	97.49	+0.06 (89.95)	-2.25
Grammar: PN; Grammar Size: 16,480/970 Task: Complete Text Chunking; F-Score: 90.14									
1,000	1	10	4	5	523.75	967	100.00	+0.06 (90.2)	-0.75

Table 8: Results for automatic optimisation tests.

Current results show insensitivity to the precise values of parameters b and n. What appears to matter is just the total number of evaluations, results being better the more candidate solutions are evaluated. Results indicate a greater sensitivity to the value of x: a data subset size of 10 is clearly too small, as search quickly finds solutions with an F-Score of 100 and then stops (last row of Table 8).

Overall, results are not nearly as good as might have been expected after the preliminary tests described above. Only small numbers of nonterminals were merged, and small improvements achieved, before search stopped. However, the fact that every single run achieved an F-Score improvement and almsot all runs resulted in a decrease in grammar size even for small numbers of merged nonterminals indicates that the basic approach is right, but that some way has to be found of overcoming the local optima on which search in the reported experiments stopped, by widening the width of the beam, changing the evaluation function, or by using a more sophisticated search method.

6 Conclusions and further research

The first part of this paper looked at the effect of adding three different kinds of local structural context — grammatical function, parent node and depth of embedding — to a basic PCFG derived from the Wall Street Journal Corpus. Grammars were tested on four different parsing task differing in complexity and shallowness. Results showed that all three types of context improve performance on the complete parsing task, but that only parent node information improves performance on all parsing tasks. The PCFG with parent node information was particularly successful and achieved better results on the complete parsing task than the best previously published results for nonlexicalised grammars and WSJ corpus data.

In the second part of the paper, a new method for optimising PCFGs was introduced that has the effect of overcoming overspecialisation by generalising grammars. It was shown that partitions can be found that drastically reduce grammar size and significantly improve parsing performance. First results were reported for applying an automatic search method to a PCFG that incorporates parent node information,

and the tasks of baseNP chunking and complete text chunking. Results are promising, but indicate that in order to achieve radical improvements in parsing performance and grammar size, a different evaluation function and/or more sophisticated search methods may be required.

7 Acknowledgements

The research reported in this paper was carried out as part of the *Learning Computational Grammars* Project funded under the European Union's TMR programme (Grant No. ERBFMRXCT980237).

References

Krotov A, Hepple M, Gaizauskas R, Wilks Y. 2000. Evaluating two methods for treebank grammar compaction. *Natural Language Engineering*, 5(4):377–394.

Belz A. 2000. *Computational Learning of Finite State Models for Natural Language Processing*. Ph.D. thesis, COGS, University of Sussex.

Cardie C, Pierce D. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *Proceedings of COLING-ACL '98*, pp 218–224.

Manning C, Schütze H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Tjong Kim Sang E, Buchholz S. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pp 127–132.

Charniak E. 1996. Tree-bank grammars. Technical Report CS-96-02, Department of Computer Science, Brown University.

Charniak E. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of NCAI-1997*, pp 598–603.

Charniak E. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*, pp 132–139.

Schmid H. 2000. LoPar: Design and implementation. Bericht des Sonderforschungsbereiches "Sprachtheoretische Grundlagen für die Computerlinguistik" 149, Institute for Computational Linguistics, University of Stuttgart.

Ramshaw L, Mitchell M. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pp 82–94. Association for Computational Linguistics.

Collins M. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL and EACL '97*, pp 16–23.

Johnson M. 1998. The effect of alternative tree representations on tree bank grammars. In *Proceedings of the Joint Conference on New methods in Language Processing and Computational Natural Language Learning (NeMLaP3/CoNLL'98)*, pp 39–48.

Collins M. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Collins M. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML 2000*.

Abney S. 1991. Parsing by chunks. In Berwick R, Abney S, Tenny C (eds), *Principle-Based Parsing*, pp 257–278. Kluwer.