Acquisition of Large Scale Categorial Grammar Lexicons

Stephen Watkinson and Suresh Manandhar
Department of Computer Science,
University of York,
York YO10 5DD, UK.
{stephen.watkinson, suresh.manandhar}@cs.york.ac.uk
Telephone: +44 1904 43{2781, 5268}
Fax: +44 1904 432767

Keywords: language learning, lexicon, Categorial Grammar

Abstract

A system is presented for inducing Categorial Grammar (CG) lexicons for natural language from either unannotated or minimally annotated corpora extracted from the Penn Treebank. A combination of symbolic and stochastic methods have been used to build a computationally effective and psychologically plausible system, which learns linguistically useful lexicons. There are a variety of parameters in the system, including the corpus annotation used, the knowledge given to the learner and the weight given to the symbolic and stochastic methods. We present results from a set of experiments that investigate these parameters. The results also show that the system performs well even when compared with systems used for simpler problems.

1 Introduction

In this paper we present a system, the Categorial Lexicon Learner (Cll), for learning wide-coverage probabilistic Categorial Grammar (CG) lexicons. It seems that humans learn language, including syntax, very effectively, so Cll has been designed focusing on both computational effectiveness and psychological plausibility, with the aim of learning linguistically plausible lexicons.

Cll can be used with a variety of settings, a number of which we present here. Two distinct settings of the knowledge available are investigated. The first allows no annotation on the examples presented to the learner (i.e. an unsupervised approach), however, Cll is provided with an initial lexicon (two different sizes have been investigated) of some closed-class words. The second allows no initial lexicon, but nouns and verbs in the corpus are annotated (i.e. a weakly supervised approach) and the possible CG categories for nouns and verbs are known.

Finally, the system uses both symbolic and stochastic constraints on the search space to direct the learner towards linguistically plausible hypotheses. This is achieved using a probabilistic version of CG. We investigate varying the strength of the probabilistic constraints to determine, which setting provides the most plausible linguistic results.

Experiments have been performed on examples from the Penn Treebank to investigate these parameters. They allow us to determine the best settings for Cll and also provide us with some indication of the knowledge and methods that can be used for a system to learn syntax in a way that is both computationally effective and psychologically plausible. The investigation of the combination of stochastic and symbolic constraints shows that these two approaches can be simply incorporated in a syntax learning setting and that simple symbolic and stochastic models provide satisfactory results. Finally, the experiments show that CLL performs respectably on a difficult learning problem, especially when compared to other systems used for less difficult problems.

The following section deals with the stochastic Categorial Grammar which is used by CLL. Section 3 discusses the use of compression in learning and how it is applied in CLL. Section 4 describes CLL in detail, including what linguistic knowledge can be made available to the learner and how the probabilistic and symbolic constraints are combined. Section 5 provides detailed information about the corpora that are used for the experiments. The experiments are described in Section 6 along with the results achieved. Similar work is discussed in Section 7. Finally, some conclusions about CLL and some extensions that need to be considered are presented in Section 8.

2 A Stochastic Model of Categorial Grammar

A large variety of formalisms have been developed to represent grammatical information. In this section we describe the Categorial Grammar formalism along with the stochastic extensions and explain why it has been used.

Categorial Grammar (CG) [22, 13] provides a functional approach to lexicalised grammar, and so, can be thought of as defining a syntactic *calculus*. Below we describe the basic (AB) CG.

There is a set of *atomic* categories in CG, which are usually nouns (n), noun phrases (np) and sentences (s). It is then possible to build up *complex* categories using the two slash operators "/" and "\". If A and B are categories then A/B and A\B are categories. With basic CG there are just two rules for combining categories: the forward (FA) and backward (BA) *functional application* rules. Following Steedman's notation [13] these are:

$$\begin{array}{lcll} X/Y\ Y & \Rightarrow & X & \quad (FA) \\ Y\ X\backslash Y & \Rightarrow & X & \quad (BA) \end{array}$$

We also include an extra rule for building compound noun phrases (which can cause problems in CG [22]).

$$\mathsf{np}\ \mathsf{np} \Rightarrow \mathsf{np} \qquad (NP)$$

In Figure 1 the parse derivation for "John ate the apple" is presented.

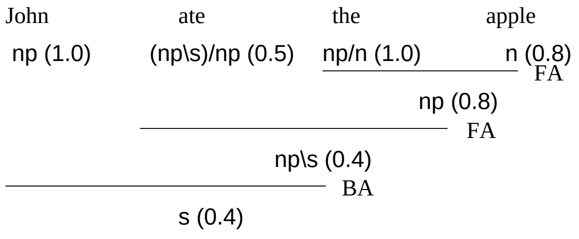

Figure 1: An example parse in CG

CG has at least the following advantages for our task.

- Learning the lexicon and the grammar is one task.
- The syntax directly corresponds to the semantics.

The first of these is vital for the work presented here. Because the syntactic structure is defined by the complex categories assigned to the words, it is not necessary to have separate learning procedures for the lexicon and for the grammar rules. Instead, it is just one procedure for learning the lexical assignments to words.

Secondly, the syntactic structure in CG parallels the semantic structure, which allows an elegant interaction between the two. While this feature of CG is not used in the current system, it could be used in the future to add semantic background knowledge to aid the learner, or for inducing semantic knowledge.

Stochastic models of grammar allow us to focus on the most likely of many possible analyses and can be used to provide a robust language model. We use a stochastic extension to Categorial Grammar, which is very similar to that given by Osborne and Briscoe [11].

Given a set of words W and a set of categories C, which those words can have assigned to them, then the lexicon consists of a set of triples (w, c, f) where $w \in W$, $c \in C$ and f is the frequency of this word-category pairing (based on the statistics from some corpus). The probability of a word being assigned a particular category is then the relative frequency of that word-category pairing with respect to the frequency of the word. That is, given a lexical entry (w, c, f) then the probability of the word w taking the category c is

$$\frac{f}{freq(w)}$$

where $freq(w)$ is the total frequency of the word w in the lexicon (i.e. the sum of the frequencies of all the lexical entries for w).

These probabilities can then be used to give likelihood assignments to parses as follows. All word-category pairs in a parse are assigned their probabilities. Then for each step of the parse, the resultant category is assigned the product of the probabilities of the combined categories. The application rules could now be written:

$$\begin{array}{rcll}(X/Y, P_i)\ (Y, P_j) & \Rightarrow & (X, P_i \times P_j) & (FA)\\ (Y, P_i)\ (X\backslash Y, P_j) & \Rightarrow & (X, P_i \times P_j) & (BA)\\ (\mathsf{np}, P_i)\ (\mathsf{np}, P_j) & \Rightarrow & (\mathsf{np}, P_i \times P_j) & (NP)\end{array}$$

where P_i and P_j are the probabilities of the categories. The probability of a particular analysis is the probability of the final category that is derived (usually an s). Alternatively, this value is the product of the probabilities of all the lexical entries used for the derivation, however, the intermediate category probabilities are used by CLL. Figure 1 includes an example of deriving the probability of a parse.

It is evident that we are calculating the probability of sequences of word-category pairs, not parses. As some of these sequences cannot be used to derive a parse, we have not actually defined a probability distribution over the set of possible parses. However, we suggest that the ranking of parses provided by this process will provide the necessary stochastic input to the learning process.

3 Compression as Learning

Inductive learning is commonly viewed as a process of compression, or optimisation [8]. In the context of natural language grammar learning, compression is vital. There are an infinite number of utterances that could need to be understood or generated, but there is only a finite space in which to represent the knowledge needed to perform the appropriate analysis. Wolff (e.g. [21]) has been one of the foremost proponents of considering natural language learning as compression from a psychological perspective, as well as computational perspective.

The results presented here use one type of compression. The aim is to build the smallest lexicons which cover the data. In this context we define "smallest" as the lexicon with the fewest number of lexical entries. Currently we are also investigating performing experiments using a minimum length encoding scheme similar to that of Osborne and Briscoe [11], which takes into account the frequency information stored in the lexicon and so can be considered to be compressing the corpus annotated with lexical categories.

4 The Learner

CLL is shown diagrammatically in Figure 2. In the following sections we explain the learning setting and the learning procedure respectively.

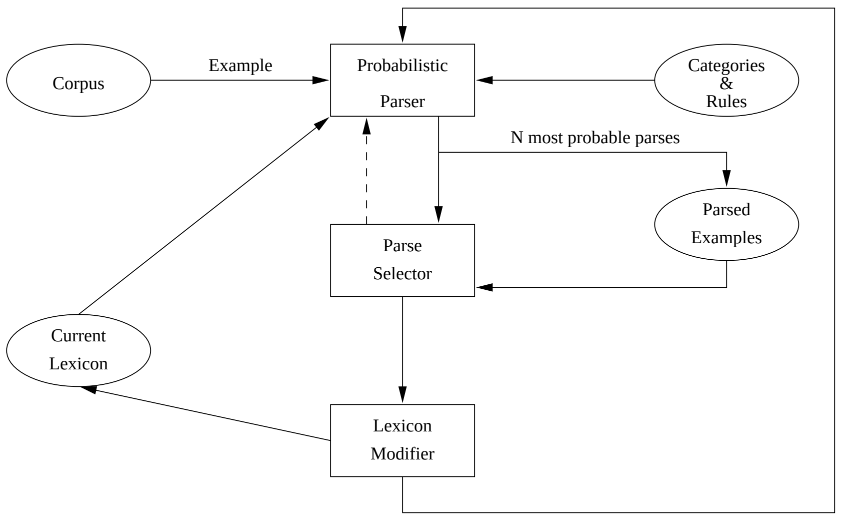

Figure 2: A diagram of the structure of the learner

4.1 The Examples

Two types of corpora have been used with two versions of CLL. Examples of the first type are simply Prolog facts containing a list of words e.g.

```
ex([mary,saw,a,man]).
```

Examples of the second type are similar, but have the nouns and verbs marked e.g.

```
ex([(N, mary),(V, saw),a,(N, man)]).
```

These types of examples have been investigated because of their psychological plausibility. It seems likely that humans do not receive examples with full structural annotation, but they may have learned some word groupings, e.g. noun and verbs [12, 19].

4.2 Initial Knowledge

The learner has four types of knowledge, which it may apply to the learning process. Again these have been chosen for their psychological plausibility [19].

Lexical Knowledge When the corpus used has no annotation, then a lexicon of closed-class words (determiners, prepositions and co-ordinating words) is supplied. Experiments have been performed with two different sizes of lexicon to determine how effective the initial lexicon is for constraining the search space. The small initial lexicon (SIL) contains 31 entries and the large initial lexicon (LIL) contains 348 entries. A lexical entry is implemented as a Prolog fact containing a word, a category and a frequency, i.e. the number of times which that word has appeared with that category, e.g.

```
lex(mary, np, 25).
```

The initial closed-class word lexicon is slightly different, as the lexical entries are marked as being closed-class and they are assigned a hand-built probability distribution prior to their use e.g.

```
lex(cc(the), np/n, 1.0).
```

The initial lexicons are used to bootstrap CLL and make it reasonably effective over a small set of examples. They are also reasonably psychologically plausible, as a child has learned some notion of simple word groupings and roles prior to learning complex syntax. However, it is perhaps more plausible that the child has learned verbs and nouns in this sense. Hence, we have the second setting where the noun and verbs are annotated in the corpus.

In this second setting, no initial lexicon is supplied, however, to maintain computational efficiency the sets of categories that nouns and verbs can take are given to Cll. They are used to reduce the ambiguity of nouns and verbs and so reduce the possible search space.

The Rules The probabilistic CG rules (see Section 2) are supplied to the learner.

The Categories The learner has a complete set of the categories that can be assigned to a word in the lexicon. Currently we are using a system with 30 possible categories, that allow most common declarative structures.

The Parser The system employs a probabilistic CKY chart parser (adapted from an algorithm given by Collins [3]), which calculates the N most probable parses, where N is the beam set by the user. The probability of a word being assigned a category is the relative frequency from the current lexicon (as described in Section 2). This probability is smoothed (for words that are not in the closed-class lexicon) to allow the possibility that the word may appear as other categories. For all categories for which the word has not appeared, it is given a frequency of one, except for the noun phrase category, which is given a frequency of ten if it has not occurred previously. This ensures an appropriate bias for the most common category for unknown words. This smoothing process is particularly useful for new words, as it ensures the category of a word is determined by its context. A more advanced smoothing approach could be included in future. However, this system allows unknown words to have their category defined by their context. It also allows unseen word-category pairs to be hypothesised.

Each non-lexical edge in the chart has a probability calculated by multiplying the probabilities of the two edges that are combined to form it (as described in Section 2). Edges between two vertices are not added if there are N (i.e. the value of the beam) edges labelled with the same category and a higher probability, (if one has a lower probability it is replaced).

It is important that the parser is efficient, as it is used on every example and each word in an example may be assigned any of a large number of categories. It is also used extensively in selecting the best parses (see below). Our probabilistic CKY parser meets this need, replacing earlier less efficient parsers.

The parser is also perhaps the most obvious place where the stochastic and symbolic constraints interact to suggest parses and hence hypothesise the changes to the lexicon. The larger the beam used, then the smaller the implications of the stochastic process, as more parses that would be available without the stochastic input are present. We use beam values of 1, 2 and 4 in the experiments to investigate the interaction between the stochastic and symbolic constraints on Cll.

4.3 The Learning Process

Having described the various components with which the learner is provided, we now describe how they are used in the learning process.

Parsing the Examples Examples are taken from the corpus one at a time and parsed. Each example is stored with the group of parses generated for it, so they can be efficiently accessed in future. The parse that is selected as the current correct parse (see below) is maintained at the head of this group. The head parse contributes information to the lexicon and annotates the corpus. The other parses are also used extensively for the efficiency of the parse selection module, as will be described below. When the parser fails to find an analysis of an example, either because it is ungrammatical, or because of the incompleteness of the coverage of the categories, the system skips to the next example.

The Parse Selector Once an example has been parsed, the N most probable parses are considered in turn to determine which can be used to make the most compressive lexicon by minimising the number of lexical entries.

Cll does not just look at what a parse would add to the lexicon. Changing the lexicon may change the results given by the parser for previous examples. Changes in the frequency of assignments may cause the probabilities of previous parses to change. This can correct mistakes made earlier when the evidence from the lexicon was too weak to assign the correct parse. Such correction is achieved by reparsing previous examples that may be affected by the changed lexicon (i.e. those examples that have some of the same words as the current example, but assigned to different categories). Not reparsing those examples that will not be affected, improves time efficiency greatly. In this way a new lexicon is built from the reparsed examples for each hypothesised parse of the current example. The parse leading to the most compressive of these is chosen. The amount of reparsing is also reduced by using stored parse information, selecting a parse with the different categories that has already been calculated and recalculating the probabilities only.

This may appear an expensive way of determining which parse to select, but it enables the system to calculate the most compressive lexicon and keep an up-to-date annotation for the corpus. Also, the parser is reasonably efficient and it is possible to do significant pruning, as outlined, so few sentences need to be reparsed each time.

Lexicon Modification The final stage takes the current lexicon and replaces it with the lexicon built using the selected parse. The whole process is repeated until all the examples have been parsed. The final lexicon is left after the final modification and the most probable annotation of the corpus is the set of top-most parses after the final parse selection.

5 Corpora

The system is applied to examples extracted from the Penn Treebank II [10, 9, 1] a corpus of over 4.5 million words of American English annotated with both part-of-speech and syntactic tree information.

The full Penn Treebank is not being used. The current corpora only contain sentences without null elements from the treebank (i.e sentences without any movement) and do not include any of the sentence fragments. They have been extracted from the Wall Street Journal Section of the Penn Treebank. The corpus, C1, contains 5000 sentences of 15 words or less with an average length of 9.56 words. To give an indication of the complexity of the corpus, the number of tokens, i.e. the total number of words including repetitions, is 47,782 and the total number of unique words, i.e. not including repetions, is 12,277. The average number of pairs of brackets in an example is 28.03.

Secondly, there is C2, a 1000 example corpus with 9467 tokens and 3731 words, which is used in the evaluation process. It also has a maximum sentence length of 15, with average length 9.47 and an average number of pairs of brackets of 27.74.

The examples in the corpora have some small modifications, such that adjacent nominals in the same subtree are combined to form a single nominal. This simplifies the learning process somewhat, as there are less compound nominals, which are difficult in CG [22].

C1 and C2 are the unannotated versions of the corpora. A version of each (AC1 and AC2 respectively) was also needed with the noun and verb annotation described above. C1 and AC1 are both used for training (i.e. building the lexicon and parsed corpus), while C2 and AC2 are used for testing (the lexicons extracted from C1 and AC1 respectively were used to parse them). Both of C1 and AC1 have been placed in example length order, to allow the learner to work on simple examples first. In future we want to carry out experiments on unordered examples.

Finally, we use two other annotated versions of the corpus. The first is a version with the bracketing of the Penn Treebank, which is used to compare the bracketing achieved in the learning process. The second, CGC, is a version of the Penn Treebank corpus where the syntactic annotation has been translated to a CG annotation (see Watkinson and Manandhar [20] for details). This is used as a gold standard for the lexical labelling, although it is not perfect for the task, as the translation is not entirely accurate and a number of the categories in CGC have variables in. It does, however, provide some guide as to the accuracy of lexical labelling for words in the examples, at least with respect to indicating an improvement of one setting of CLL over another.

6 Experiments

Lexicons have been built using a variety of settings for CLL, summarised in Table 1. Each of these experiments resulted in a broad coverage CG lexicon and a parsed corpus. The size of the lexicon is relevant, in particular to the size of the lexicon used in the CGC corpus, which is 15,136. Similarly, the average ambiguity, i.e. the average number of categories per word, gives an indication of the validity of a learned corpus and the size/complexity of the search space. CGC has an average ambiguity of 1.25 categories per word. The learned lexicon was then used to parse the respective test corpora – C2 for experiments 1 – 4 and AC2 for experiments 5 – 6. For the annotation learned the average crossing brackets measure [5] and the lexical accuracy (the percentage of correctly labelled words using CGC to define the correct labels) are also given.

Experiment Number	Training Corpus	Initial Lexicon	Parser Beam
1	C1	SIL	2
2	C1	LIL	2
3	C1	SIL	1
4	C1	SIL	4
5	AC1	-	2
6	AC1	-	1

Table 1: The experiments performed

Lexical accuracy gives us some understanding of the tagging accuracy of the lexicon when used with the parser, as well as an idea of the accuracy of the corpus that has been tagged. The crossing brackets rate gives a good idea of structural plausibility of the lexicons learned and the corpora built. Both have been used before with similar learning systems [6, 11].

Experiments on simple corpora of examples generated from hand-built grammars, using an earlier version of CLL are described by Watkinson and Manandhar [17, 18].

It seems that increasing the bootstrapping information provided by the initial lexicon is effective in improving the learning with respect to lexical accuracy – again this is not surprising, as the initial lexicon supplies information directly involved in lexical labelling. However, the fact that the crossing bracket rate increases when LIL is used instead of SIL, suggests that over-constraining the system with an initial lexicon may reduce the flexibility required to learn the best lexicons.

Currently it would appear that fairly strong probabilistic constraints are useful. There is not a great deal of difference between beams of 1 and 2, but 1 is consistently slightly better. Similarly, both are better than a beam of 4. This may suggest some future work that does not require the lexicon selection stage, as only the most likely parse need be considered. This could be used to eliminate the reparsing stage, although there would need to be some tech-

Experiment Number	Lexicon Size	Average Ambiguity	Lexical Accuracy		Average CBR	
			C1	C2	C1	C2
1	12,706	1.21	44.76	47.53	5.43	4.70
2	13,851	1.24	49.54	51.89	5.61	4.86
3	12,070	1.21	45.36	47.68	5.44	4.48
4	12,044	1.21	44.56	47.26	5.96	5.32
5	13,010	1.12	35.59	35.48	5.11	4.40
6	12,903	1.11	37.11	37.22	5.19	4.38

Table 2: The results of the experiments

nique for removing the effects of bad parses from the lexicon (perhaps one reparse at the end of the learning process).

Finally the results with the different corpus annotations are interesting. The experiments with the unannotated corpora achieve better results with respect to lexical accuracy. However, with respect to the crossing bracket rate, the experiments with noun and verb annotated corpora give better results. This seems to be due to the fact that using the initial lexicon ensures that a large number of words are accurately labelled, but it sometimes reduces the flexibility too much to allow the parser to build an accurate structure. This may in turn be due to the lack of flexibility in the categories that are available to the learner. The noun and verb annotated setting on the other hand provides the learner with a bias towards the correct structure of the examples, hence giving a good crossing bracket rate, but the initial knowledge is perhaps too weak to ensure a good labelling.

The lexical accuracy results may appear low, but considering that a base line accuracy of randomly assigning categories to words would return a value of 3.33%, they do suggest a significant level of learning.

7 Related Work

Wolff [21] using a similar (if rather more empiricist) setting also uses syntactic analysis and compression to build grammars. However, this syntactic analysis would appear to be very expensive and the system has not been applied to large scale problems. The compression metric is applied with respect to the compression of the corpus, rather than the compression of syntactic information extracted from the corpus. It seems unlikely that the simple induction algorithm would generate linguistically plausible grammars when presented with complex naturally occurring data.

Joshi and Srinivas [6] have developed a method called supertagging that similarly attaches complex syntactic tags (supertags) to words. Their most effective learning model was a combination of symbolic and stochastic techniques, like the approach presented here. However, the a full lexicon is supplied to the learner, so that the problem is reduced to one of disambiguating between the possible supertags. The learning appears to be somewhat supervised as it occurs over parts-of-speech rather than over the actual words. However, label accuracy is supplied and this can be compared with the accuracy of our system.

Osborne and Briscoe [11] present a somewhat supervised system for learning unusual stochastic CGs (the atomic categories are far more varied than standard CG) again using part-of-speech strings rather than words. While the problem solved is simpler, this system provides a suitable comparison for learning appropriate lexicons for parsing.

Neither Joshi and Srinivas [6] nor Osborne and Briscoe [11] are psychologically plausible, but they are computationally effective and they do give suitable results for comparison.

Osborne and Briscoe [11] achieve average crossing bracket rates of around 3 using a variety of Minimum Description Length and Maximum Likelihood approaches. Given that they are solving a simpler problem, the results we present here, with a best of 4.18 seem respectable.

Joshi and Srinivas [6] achieve a lexical accuracy of 78%. However, given that this is a much simpler disambiguation task, as the system is given a full lexicon, our results are respectable. As noted above, a random algorithm applied to our problem would, on average, return a lexical accuracy of 3.33%. Assuming the average ambiguity of the CGC corpus (i.e. 1.25) then if a full lexicon was supplied, a random algorithm should achieve a lexical accuracy of 80%. While the average ambiguity may be higher in general, this puts the results in perspective.

Recently there has been work on unsupervised learning of syntax. A variety of approaches have used the fact that constituents can be substituted for one another to identify constituents in groups of sentences and then extract rules [2, 7, 16, 15, 14]. The results of these systems, when given (Clark [2] gives a good summary), show better crossing bracket rates (0.82–2.12) than that obtained here, but the systems do not appear to provide complete bracketings (they have very low recall figures), whereas CLL provides a complete parse for each example.

8 Conclusions

We have presented CLL, an effective lexicon learning and corpus parsing system that works in both unsupervised and weakly supervised environments. We have also presented a set of experiments comparing a variety of parameters that need to be considered when building language learning systems. These have shown that the system is effective as a first step and builds useful lexicons and annotated corpora. CLL also shows the advantages of using stochastic models *alongside* symbolic models within language learning. Finally, we have begun to investigate what sort of plausible linguistic knowledge may actually be useful for CLL.

A large variety of further experiments could be performed using CLL. In particular we are investigating more complex forms of compression. It is also our intention to perform experiments on the full Penn Treebank, including examples with movement. This may require extending the CG we are currently using.

References

[1] Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. *Bracketing Guidlines for Treebank II Style Penn Treebank Project*, 1994.

[2] Alexander Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In Daelemans and Zajac [4], pages 105 – 112.

[3] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

[4] Walter Daelemans and Rémi Zajac, editors. *Proceedings of the Workshop Computational Natural Language Learning (CoNLL-2001)*, 2001.

[5] Joshua Goodman. Parsing algorithms and metrics. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 35 – 64. Association for Computational Linguistics, 1996.

[6] Aravind K. Joshi and B. Srinivas. Disambiguation of super parts of speech (or supertags): Almost parsing. In *Proceedings of the 15th Conference on Computational Linguistics (COLING'94)*, pages 154–160, 1994.

[7] Dan Klein and Christopher D. Manning. Distributional phrase structure induction. In Daelemans and Zajac [4], pages 113 – 120.

[8] M. Li and P.M.B. Vitányi. Theories of learning. In *Proceedings of the International Conference of Young Computer Scientistis*, 1993.

[9] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating predicate argument structure. In *The ARPA Human Language Technology Workshop*, 1994.

[10] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19, 1993.

[11] Miles Osborne and Ted Briscoe. Learning stochastic categorial grammars. In *Computational Natural Language Learning Workshop CoNLL'97*, pages 80–87, 1997.

[12] Steven Pinker. Language acquisition. In Daniel N. Oshershon and Howard Lasnik, editors, *An Invitation to Cognitive Science: Language*, volume 1, pages 199–241. The MIT Press, 1990.

[13] Mark Steedman. Categorial grammar. *Lingua*, 90:221 – 258, 1993.

[14] Menno van Zaanen. Abl: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics, COLING'00*, pages 961 – 967, 2000.

[15] Menno van Zaanen and Pieter Adriaans. Comparing two unsupervised grammar induction systems: Alignment based learning vs. emile. Technical Report 2001.05, School of Computing, University of Leeds, 2001.

[16] Marco Robert Vervoort. *Games, Walks and Grammars: Problems I've Worked On.* PhD thesis, Universiteit van Amsterdam, 2000.

[17] Stephen Watkinson and Suresh Manandhar. Unsupervised lexical learning with categorial grammars. In Andrew Kehler and Andreas Stolcke, editors, *Proceedings of the Workshop on Unsupervised Learning in Natural Language Processing*, pages 59–66, 1999.

[18] Stephen Watkinson and Suresh Manandhar. Unsupervised lexical learning with categorial grammars using the LLL corpus. In James Cussens and Sašo Džeroski, editors, *Learning Language in Logic*, volume 1925 of *Lecture Notes in Artificial Intelligence.* Springer, 2000.

[19] Stephen Watkinson and Suresh Manandhar. A psychologically plausible and computationally effective approach to learning syntax. In Daelemans and Zajac [4], pages 160 – 167.

[20] Stephen Watkinson and Suresh Manandhar. Translating treebank annotation for evaluation. In *Proceedings of the Workshop on Evaluation Methodologies for Language and Dialogue Systems, ACL/EACL 2001*, 2001.

[21] J.G. Wolff. Cognitive development as optimisation. In L. Bolc, editor, *Computational Models of Learning.* Springer Verlag, 1987.

[22] Mary McGee Wood. *Categorial Grammars.* Linguistic Theory Guides. Routledge, 1993. General Editor Richard Hudson.