Boosting Unsupervised Grammar Induction by Splitting Complex Sentences on Function Words

Jonathan Berant[1], Yaron Gross[1], Matan Mussel[1], Ben Sandbank[1], Eytan Ruppin,[1] and Shimon Edelman[2]
[1]Tel Aviv University and [2]Cornell University

1 Introduction

The statistical-structural algorithm for unsupervised language acquisition, ADIOS (for Automatic DIstillation Of Structure), developed by Solan et al. (2005), has been shown capable of learning precise and productive grammars from realistic, raw and unannotated corpus data, including transcribed children-directed speech from the CHILDES corpora, in languages as diverse as English and Mandarin. This algorithm, however, does not deal well with grammatically complex texts: the patterns it detects in complex sentences often combine parts of different clauses, negatively affecting performance. We address this problem by employing a two-stage learning technique. First, complex sentences are split into simple ones around function words, and the resulting corpus is used to train an ADIOS learner. Second, the original complex corpus is used to complete the training. We also show how the function words themselves can be learned from the corpus using unsupervised distributional clustering.

2 The ADIOS algorithm

ADIOS (Solan et al., 2005) is a statistical-structural algorithm for unsupervised language acquisition, which takes a corpus (a set of sentences) from some language as an input, and outputs an estimate for the grammar of that language.

2.1 The data structure

The first step in the ADIOS algorithm is to load the corpus into its data structure, which is a directed pseudograph. Every unique word in the corpus is represented in the graph by a *node*. Every sentence in the corpus is represented in the graph by a *path*, defined as an ordered set of graph edges. A path starts at a special node called the *begin node*, follows the graph edges to traverse the nodes corresponding to the words in the order of their appearance in the sentence, and finally reaches a second special node called the *end node* (see Figure 1). The graph is non-simple and may contain both loops and multiple edges between any pair of nodes.

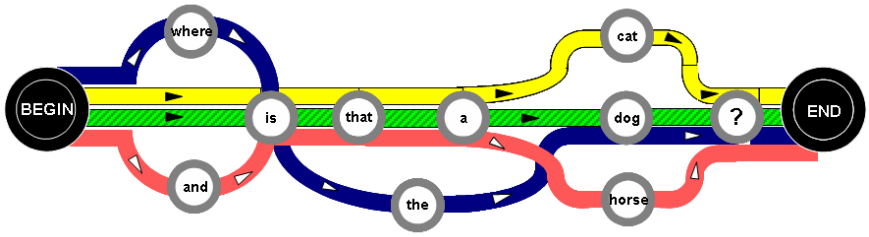

Figure 1: The ADIOS grammar learning algorithm (Solan et al., 2005) represents the training corpus as a pseudograph. The simple example depicted here represents the following four-sentence corpus: (1) is that a cat? (2) is that a dog? (3) where is the dog? (4) and is that a horse?

The loading procedure ensures that the number of paths in the graph is equal to the number of sentences in the corpus. We define the language that the algorithm has learned at any given moment as the set of sentences that can be produced by traversing any path and generating a string by accruing the nodes it passes through. At initialization, every path generates exactly one sentence, because no generalization has taken place yet.

Unsupervised grammar induction algorithms seek to infer hierarchical tree representations for the corpus sentences, making explicit the constituents that are interchangeable in various contexts. The ADIOS algorithm constructs the hierarchical representations in a bottom-up fashion, simultaneously carrying out sentence segmentation and constituent generalization.

2.2 Segmentation criterion

Segmentation of the input sentences is performed using a statistical segmentation criterion, MEX (Motif EXtraction), which scans the graph for *patterns*. Intuitively, a sequence of nodes is a pattern if there is a statistically significant bundle of paths that traverses it (for details see Solan et al., 2005). The advantage of the pseudograph data structure is that sentences are automatically aligned along nodes at which they overlap, and thus patterns are easily identified (see Figure 2).

The ADIOS algorithm iterates the MEX procedure along the paths of the graph, searching for patterns. For every path, the best pattern (if any) is chosen and rewired onto the graph: a new node P is created for the pattern, and for every instance of the pattern in the graph, the nodes of the pattern are replaced by the new node P. This leads to a reduction in the size of the graph. By performing this process on all of the paths iteratively, hierarchical structures are rapidly created.

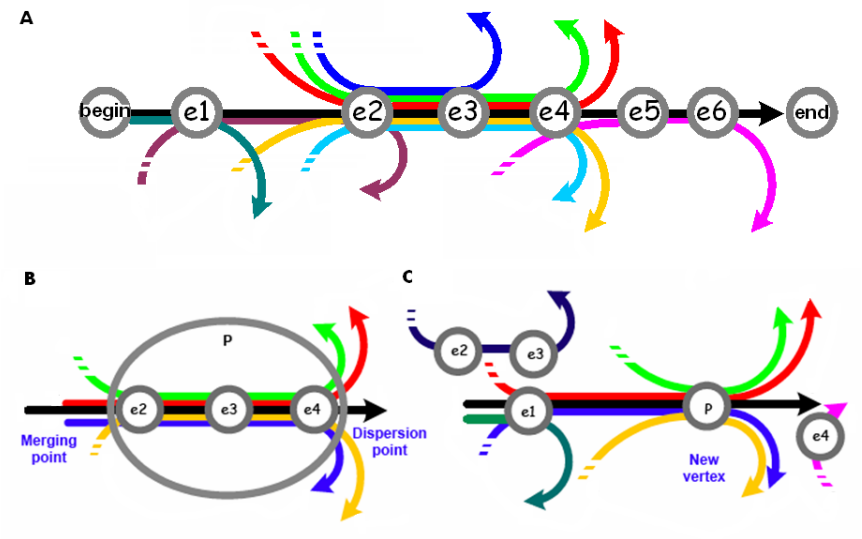

Figure 2: Pattern detection in the pseudograph: (A) The data structure allows for the easy identification of aligned sentences. (B) A set of nodes is identified as a pattern when a bundle of paths traverse it. (C) The pattern is rewired onto all of the paths that traverse the nodes of the pattern.

The algorithm stops when no new patterns are found in the graph.

2.3 Generalization

As mentioned, the segmentation procedure alone is capable of parsing the input sentences into a hierarchical structure. However, the size of the language remains unchanged by this procedure. To achieve generalization, ADIOS augments the MEX procedure with a search for elements that are in complementary distribution in the context of a given pattern.

The search for complementary-distribution elements is performed by sliding a context window of size L along a candidate path, seeking other paths that coincide with it within this window in all places but one. Suppose the paths diverge in the i'th place; the element in slot i in the candidate path is then temporarily replaced by an *Equivalence Class* (EC) that comprises all the nodes found in this slot in the partially aligned paths (see Figure 3), forming the generalized path. Thereafter, the usual MEX procedure is performed on the generalized path, which may lead to the creation of a pattern that contains an equivalence class. This is done for every possible position of the window on the path, and for every possible slot inside the window. The best pattern found is rewired into the graph. If a generalized pattern is thus rewired, the grammar of the learner becomes capable of generalization.

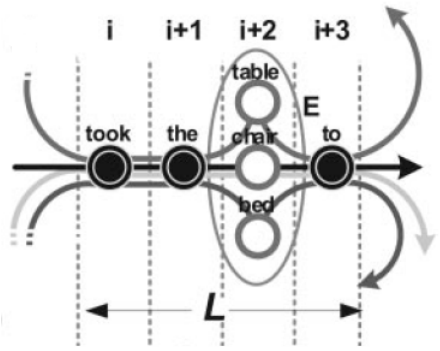

Figure 3: Generalization: looking at the paths that coincide with the search path along the context window, except for slot *i+2*, leads to the creation of a temporary *equivalence class* (EC) node at that slot. If a pattern that contains the generalized node is rewired onto the graph, the EC node becomes permanent.

ATIS version	Language model	Perplexity
2	ADIOS	11.5
2	Trigram Kneser-Ney Back-Off smoothing	14
2	PFA Inference (ALERGIA) + trigram	20
3	ADIOS	13.5
3	SLM-wsj + trigram 1.E+05 15.8 [10]	15.8
3	NLPwin + trigram	15.9
3	SLM-atis + trigram	15.9

Table 1: The lower perplexity achieved by ADIOS-derived language models comparing to standard benchmarks on the ATIS-2 and ATIS-3 corpora.

Applying this process on all of the paths iteratively leads to a rapid growth in the size of the language that has been learned. A sentence is in the language if there exists a path that can generate it (a pattern node generates the concatenation of the strings generated by its children; an EC node generates a string generated by one of its children).

The ADIOS algorithm has been subjected to extensive testing on a variety of corpora (Solan et al., 2005). Its performance generally exceeded that of other unsupervised grammar induction algorithms capable of dealing with raw corpus data, on various measures, such as precision and recall. In particular, in the language modeling task the perplexity levels achieved by ADIOS-derived models on the ATIS-2 and ATIS-3 corpora were lower than the standard benchmarks (Table 1).

Corpus	Word types	#sentences	Average sentence length
CHILDES	14,401	320,000	6 words
ATIS-N	1,153	12,700	10 words
Children's literature	52,180	41,129	52 words

Table 2: The average length of a sentence in simple corpora such as CHILDES and ATIS-N is much lower than in complex corpora such as children's literature

3 The problem and motivation for solution

Although ADIOS attains good performance when applied to corpora such as ATIS, CHILDES and artificial context-free grammars, it encounters difficulties when faced with more complex corpora such as texts from the Wall Street Journal or from children's literature. Specifically, executing the ADIOS algorithm on complex corpora results in a grammar with few patterns and poor generalization. The complexity of the WSJ and the literature corpora is manifested in two ways. First, the average length of a sentence in ATIS and CHILDES is much smaller than in more complex corpora (see Table 2). Second, ATIS and CHILDES are characterized by a relatively restricted number of grammatical structures: most of the sentences are simple questions or imperatives. In comparison, complex corpora contain multi-clause sentences that have a much more diverse and sophisticated grammatical structure.

Because complex sentences across languages are usually formed by joining together a few simple sentences according to some grammatical construction, a possible way of dealing with grammatical complexity may be to decompose the complex sentences into simple ones, and to allow ADIOS to learn on a simpler corpus, where one may expect better performance. This idea is related to what Elman (1990) called "the importance of starting small" (in that study, training a neural network progressively on a sequence of corpora of increasing complexity led to better learning).

Our method for dealing with complex sentences by decomposing them into simpler ones, which will be described in the next section, highlights the importance of the distinction between closed- and open-class words. Across languages, lexical items can be divided into two categories: content, or open-class words, and function, or closed-class words. Words in these two categories have distinct distributions in natural languages. Function words, which form a small, closed class (you cannot invent a new preposition, say, and expect to be understood) are very frequent in natural discourse. In comparison, content words form a much larger class, each of whose members, however, occurs much more rarely. A serious limitation of the ADIOS algorithm is its blindness to the distinction between closed- and open-class words. This results in a tendency for function words to appear at the edges of patterns early in the learning process (see Figure 4). This happens because many paths pass through a node of a frequent function word and

then branch out to different content words.

4 The Algorithm

Our solution uses a subset of function words, which tend to mark clause boundaries, as cues for decomposing complex sentences into simpler ones. Lexical items that are markers for the edge of a clause will be referred to as *conjunctions*. The algorithm consists of four steps (see Figure 5):

A Every sentence in the corpus is decomposed into a set of simple sentences by splitting on the conjunctions. The conjunctions themselves are omitted altogether from the sentences.

B The simpler corpus is loaded onto the graph and the ADIOS algorithm is executed.

C When learning concludes, the simple sentences are recomposed in their new generalized form.

D The ADIOS algorithm is applied again to the recomposed corpus.

The algorithm draws on the principles described in the previous section. Training is done in two phases. During the first phase, the learning procedure is restricted to deriving patterns from simple sentences only. When this stage concludes, the original complex sentences are made visible to the algorithm and learning continues. Conjunctions are used to split complex sentences to simpler ones. This

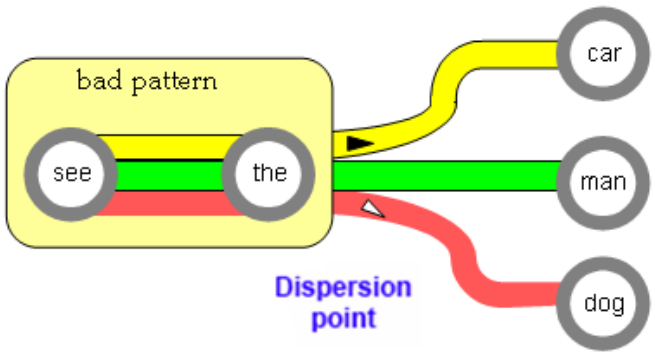

Figure 4: In the ADIOS procedure, function words are often a dispersion point for many paths, due to their peculiar statistical distribution. Because of this property, "bad" patterns that cross clause and phrase boundaries are created.

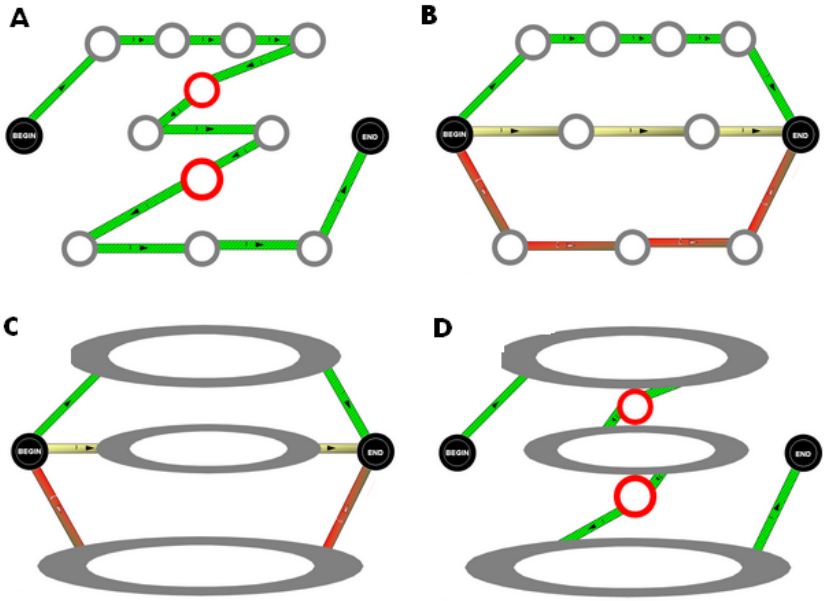

Figure 5: The algorithm for learning complex syntax. (A) One path in the graph is shown; conjunctions are highlighted. (B) The path is split on its conjunction nodes into several paths. (C) Learning is carried out on the simple paths, using the ADIOS algorithm. (D) Conjunctions are reinstated and training is repeated.

prevents the algorithm from creating patterns that cross clause boundaries early in the learning procedure.

5 Experiments and results

To test our algorithm, we constructed an artificial context-free grammar (jym1) that generates sentences of three types of complexity (Diessel, 2004):

(a) Complementation — "he believes that John runs."

(b) Relativization — "I like the books that you buy."

(c) Coordination — "After I run, I drink."

The grammar contains 155 lexical items and 240 derivation rules. The grammar divides the verbs in the lexicon into six classes according to their theta-grid and

S → Sentence | Sentence comma coor Sentence
Sentence → NPSubHumSing VPHumHumSing | NPSubObjPl VPObjPl
NPSubHumSing → DetSing NHumSing
VPHumHumSing → VHumHumSing NPHumSing
comma → ,
coor → after | before | because | although | when | where
DetSing → a | the
NHumSing → boy | man | fellow
VHumHumSing → kiss | touch | kick

Figure 6: A fragment from the jym1 context-free grammar, used to generate the corpus that tests the algorithm. The fragment demonstrates some of the complexities in the grammar, such as the distinctions made between verbs that take different types of arguments.

subcategorization frame. It distinguishes between human and non-human arguments, between external and internal arguments, and between noun phrases and sentences as internal arguments. Figure 6 presents a small fragment of the grammar. A corpus was generated from the jym1 grammar and divided into a training and testing parts. The training corpus consists of $8,500$ sentences and the testing corpus consists of 923 sentences. Our algorithm was trained and tested on these corpora.

To test the performance of ADIOS we used the measures of recall and precision. Recall was defined as the proportion of sentences from the test set that were accepted by the learner at the end of the training. This measures the coverage of the target language that the learner achieved. Precision was defined as the proportion of sentences generated by the learner that were accepted by the jym1 context-free grammar. This measures the accuracy of the learner.

Ten ADIOS learners were presented with the jym1 corpus (learners vary with respect to the order of sentences in the corpus). The mean recall and precision are shown in Figure 7. The splitting procedure resulted in a significantly improved recall: from 0.81 to 0.9 ($p < 0.01$); the precision did not change significantly ($p = 0.14$).

Although precision decreased from 0.2 to 0.14 (n.s.) when we split on conjunctions, a manual comparison between the corpora generated in the two different configurations gave the impression that the sentences generated when splitting on conjunctions were as good as in the normal setting. To test that we decided to measure our accuracy using *edit distance*. The edit distance between a sentence generated by a learner and the grammar is defined as the minimal number of elementary edit operations performed on the sentence that would turn it grammatical (Lyon, 1974). An edit operation is either the insertion of a lexical item to the sentence, the deletion of a lexical item from the sentence, or the substitution of a

lexical item in the sentence for another. The edit distance of a learner is defined as the average edit distance to a correct sentence across the entire corpus it generates. Edit distance is a much relaxed measure than precision, because it gives some credit to sentences that are only partially correct. Results on the ten learners show that the edit distance when splitting on conjunctions (2.15) is almost identical to the edit distance achieved in the normal setting, (2.09), supporting our claim that the accuracy of the algorithm was not harmed due to splitting ($p = 0.74$).

Next, we tried to split the sentences of the corpus not only on their conjunctions but also on their complementizers (e.g., "I like the books that you buy" would be split into "I like the books" and "you buy"). In this experiment, recall improved from 0.81 to 0.89 ($p < 0.01$), but precision also decreased significantly from 0.2 to 0.1 ($p < 0.05$). Thus, splitting only on conjunctions yielded better results. To confirm that the improvement stemmed from the linguistically relevant segmentation and not merely from the shorter sentences handed to the algorithm, we split the sentences to arbitrarily shorter phrases with the same mean length as in the first experiment. Again, ten learners were trained, but now recall significantly decreased from 0.81 to 0.53 while precision remained unchanged.

Even though splitting on conjunctions improved recall significantly, it was relatively high even in the baseline condition, when the regular version of ADIOS was used. As we mentioned before, when regular ADIOS is applied to complex corpora, the usual outcome is low recall. Thus, we wanted to see the effect of splitting the sentences when the initial recall is low. To that end, we constructed a second context free grammar (jym3) that differed from jym1 in two respects. First, the size of the lexicon was increased to 365 lexical items (compared to 155 lexical items in jym1). Given that the size of the corpus is fixed, this means that every lexical item was now much rarer, making generalization more difficult. Second, ambiguities were removed from the lexicon. In the jym1 corpus, ten lexical items belonged to two different categories each. Lexical items that belong to multiple categories may cause overgeneralization, and so all of the ambiguous lexical items were replaced by non-ambiguous ones.

From the jym3 context-free grammar, a training set of $9,000$ sentences and a test set of 1000 sentences was generated. Recall improved dramatically from 0.17 in the normal execution (regular ADIOS) to 0.59 when splitting the sentences. Precision on the other hand significantly decreased from 0.55 to 0.29. Given that we weigh recall and precision equally, the overall performance after the splitting was better: the F-score (defined as the geometric mean of precision and recall) improved significantly from 0.24 to 0.39. In general, splitting alone managed to boost ADIOS to a higher level of performance on a complex corpus.

6 Unsupervised identification of conjunctions

A lacuna in the approach described above is the need to provide the algorithm ahead of time with a list of conjunctions on which to split the complex sentences.

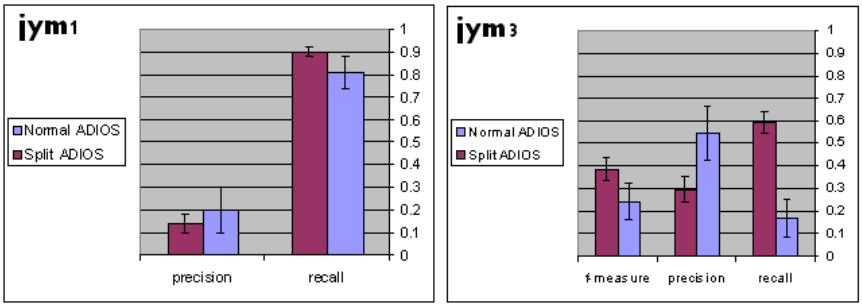

Figure 7: Results for the jym1 and jym3 corpora, showing the precision and recall measures, for the normal (baseline) version of ADIOS, compared to the version that splits complex sentences on conjunctions in the first phase of training.

Of course, no such list is available to a child acquiring a language. Children, however, are very good at recognizing function words very early on (Shi et al., 2006), using cues such as prosody. Thus, one may conjecture that children have access to various sources of information that help mark clause boundaries, and that are unavailable to an algorithm that only relies on text. Thus, it would be useful to show that clause boundaries and conjunctions are identifiable using only the statistical properties of their distribution in language.[1] If that can be demonstrated, the independently learned conjunctions can be passed on to ADIOS for the purpose of splitting. This would keep the entire learning process unsupervised.

The key to identifying conjunctions is to realize that the input is not just a finite sequence of words, but rather a set of sentences (much like natural language divides speech into utterances using prosody). Hence, it is possible to characterize what words appear at the beginnings of sentences, and what words appear at the endings of sentences. Intuitively, conjunctions are lexical items that coordinate two constituents which could be separate sentences in their own right. Thus, it is possible to characterize conjunctions as lexical items that are followed by words that usually appear at the beginning of a sentence, and are preceded by words that usually appear at the ending of a sentence. A natural way of formalizing this intuition is to use straightforward distributional clustering:

- Create a vector $V_{beginning}$ that counts for every possible bi-gram *b* in the language the frequency of *b* appearing at the beginning of a sentence. Create a similar vector V_{end} for the ending of the sentences.
- For every word *w* in the language, create a pair of vectors $W_{beginning}$ and W_{end}. $W_{beginning}$ counts for every possible bi-gram *b* in the language the frequency of *b* following *w*; W_{end} counts the frequency of *b* preceding *w*.

[1] Although in this work we only show a method for identifying conjunctions, it seems that it can be easily expanded to identifying clause boundaries.

Rank	Conj. in jym1	Conj. in children's literature
1	,	and
2	and	but
3	because	because
4	although	that
5	before	though
6	when	till
7	where	where
8	after	until
9		whether
10		which
11		when
12		for
13		thought
14		to

Table 3: The algorithm for unsupervised identification of conjunctions succeeds in ranking all of the conjunctions in the jym1 corpus at the top of the output list. When applied on the children's literature corpus, the top 25 lexical items found by the algorithm (with one exception) are all conjunctions, complementizers and prepositions.

- For every pair of vectors $W_{beginning}$ and W_{end}, calculate the angle α_1 between $W_{beginning}$ and $V_{beginning}$ and the angle α_2 between W_{end} and V_{end}.
- Let α be the mean of α_1 and α_2. Sort all the words in the lexicon according to α in increasing order and return the sorted list.

In the jym1 corpus, eight words are defined as conjunctions. Applying the algorithm to the jym1 corpus results in the eight conjunctions being ranked at the top of the output list. To test the conjunction detection method on natural data, we applied it to a fragment of the children's literature corpus ($55,000$ sentences and $6,300$ different word tokens[2]). The top 25 lexical items that were found by the algorithm (with one exception) were all conjunctions, complementizers and prepositions (see Table 3). To conclude, it seems that the method we developed succeeds in identifying words that are probable markers for clause boundaries.

[2]The entire corpus can be downloaded from Project Gutenberg (http://www.gutenberg.org) under the subject "children's literature." We took a random sample from the English corpus and divided it into sentences using simple preprocessing techniques.

7 Conclusion

The previously described statistical-structural algorithm for unsupervised language acquisition, ADIOS, detects in complex sentences patterns that combine parts of different clauses, negatively affecting performance. We addressed this problem by splitting complex sentences on function words, thus simplifying the complex sentences and preventing "across-clauses" patterns from occurring. To test this approach, we constructed an artificial CFG that generates sentences of three types of complexity: complementation, relativization and coordination. Splitting the complex sentences on function words indeed had a positive effect on the learning performance. Furthermore, we developed a method for an unsupervised identification of conjunctions, based solely on the statistical properties of these particles, and using distributional clustering. In conclusion, we presented a method for extending the capabilities of the ADIOS algorithm without detracting from its unsupervised nature.

Acknowledgments. Thanks to the US-Israel Binational Science Foundation and the Horowitz Center for Complexity Science for support.

References

Diessel, H. (2004). *The Acquisition of Complex Sentences*, volume 105 of *Cambridge Studies in Linguistics*. Cambridge University Press, Cambridge.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Lyon, G. (1974). Syntax-directed least-errors analysis for context-free languages: a practical approach. *Communications of the ACM*, 17:3–14.

Shi, R., Werker, J. F., and Cutler, A. (2006). Recognition and representation of function words in English-learning infants. *Infancy*, 10:187–198.

Solan, Z., Horn, D., Ruppin, E., and Edelman, S. (2005). Unsupervised learning of natural languages. *Proceedings of the National Academy of Science*, 102:11629–11634.