Unsupervised Efficient Learning and Representation of Language Structure

Zach Solan, David Horn, Eytan Ruppin
Faculty of Exact Sciences
Tel Aviv University
Tel Aviv, Israel 69978
{*rsolan,horn,ruppin*}*@post.tau.ac.il*

Shimon Edelman
Department of Psychology
Cornell University
Ithaca, NY 14853, USA
se37@cornell.edu

Abstract

We describe a linguistic pattern acquisition algorithm that learns, in an unsupervised fashion, a streamlined representation of corpus data. This is achieved by compactly coding recursively structured constituent patterns, and by placing strings that have an identical backbone and similar context structure into the same equivalence class. The resulting representations constitute an efficient encoding of linguistic knowledge and support systematic generalization to unseen sentences.

Motivation

Considerations of representational parsimony dictate that the explanation for the pattern of acceptable sentences in a language be as concise as possible. A reduced representation of linguistic knowledge need not, however, take the form of a meta-language such as a prescriptive rule-set or grammar [1]. Instead, syntax may constitute an abstraction, emerging from a corpus of language [2], yet coexisting within the same representational mechanism that embodies the data. The process of abstraction can be guided by principles such as complementarity of distributions: tokens that function similarly in some sense (phonological, morphological, syntactic or semantic) but represent systematic rather than free variation will form complementary distributions or classes (e.g., [1, 3]).

In thinking about emergent regularities [2], or syntactic-semantic constructions [4], we adopt Langacker's vision:

> "...particular statements (specific forms) coexist with general statements (rules accounting for those forms) in a speaker's representation of linguistic convention, which incorporates a huge inventory of specific forms learned as units (conventional expressions). Out of this sea of particularity speakers extract whatever generalizations they can. Most of these are of limited scope, and some forms cannot be assimilated to any general patterns at all. Fully general rules are not the expected case in this perspective, but rather a special, limiting case along a continuum that also embraces totally idiosyncratic forms and patterns of all intermediate degrees of generality." [5], p.43.

Langacker's conception of grammar as an inventory of linguistic units, which is structured in the sense that some units function as components of others, is well-suited to serve as a basis for a psychologically motivated theory of language learning, due to its clear parallels with the notion of *unitization* that arises in cognitive psychology [6]. Recent developments in probability and information theory and in computational learning have rendered distributional [1] methods of linguistic unitization both more tractable and more readily relatable to grammar-based formalisms [7]. The present paper describes a project that aims to transform the idea of the emergence of distributional syntactic and semantic knowledge into a concrete computational model, constrained by psycholinguistic data and striving for biological plausibility.

The ADIOS model

The ADIOS (Automatic DIstillation Of Structure) model constructs syntactic representations of a sample of language from raw, unlabeled corpus data. The model has two components: (1) a Representational Data Structure (RDS) graph, and (2) a Pattern Acquisition (PA) algorithm that learns the RDS in an unsupervised fashion. The PA algorithm aims to detect *patterns* — repetitive sequences of "significant" *paths* (strings) of primitives occurring in the corpus. In that, it is related to prior work on alignment-based learning [8] and regular expression ("local grammar") extraction [9] from corpora. We stress, however, that our algorithm requires no prejudging either of the scope of the primitives or of their classification, say, into syntactic categories: all the information needed for its operation is extracted from the corpus in an unsupervised fashion.

In the initial phase of the algorithm, the text is segmented into the smallest possible morphological constituents (e.g., ed is split off both walked and bed; the algorithm later discovers that bed should be left whole, on statistical grounds).[1] This initial set of unique constituents is the vertex set of the newly formed RDS (multi-)graph. A directed edge is inserted between two vertices whenever the corresponding transition exists in the corpus (Figure 1(a)); the edge is labeled by the sentence number and by its within-sentence index. Thus, corpus sentences and sub-sentences initially correspond

[1] We remark that the algorithm can work in any language, with any set of tokens, including individual characters – or phonemes, if applied to speech.

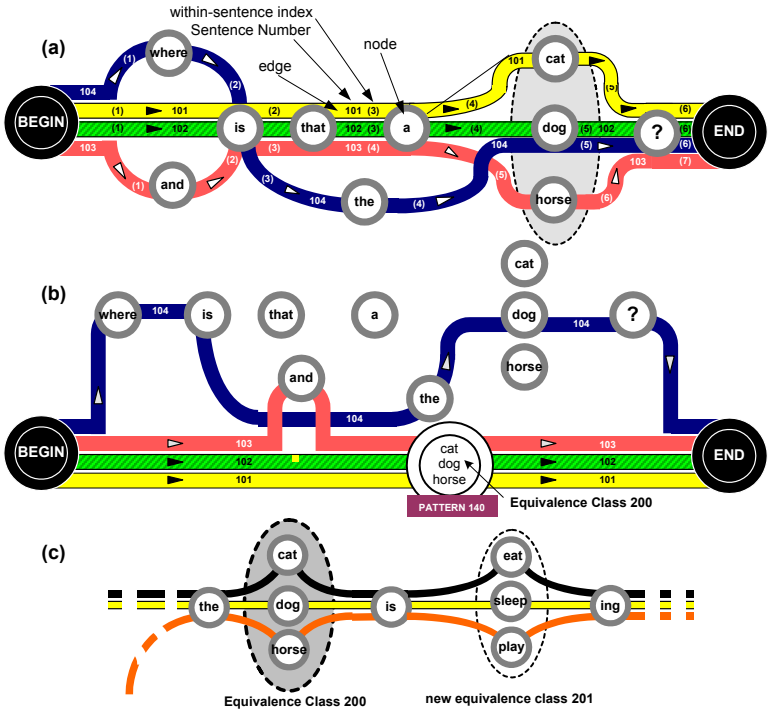

Figure 1: (a) A small portion of the RDS directed multi-graph for a simple corpus containing sentences #101 (is that a cat?) #102 (is that a dog?) #103 (and is that a horse?) #104 (where is the dog?). Each sentence is depicted by a solid colored line; edge direction is marked by arrows and is labeled by the sentence number and within-sentence index. The sentences in this example join a pattern is that a {dog, cat, horse} ?. (b). The abstracted pattern and the equivalence class associated with it are highlighted (edges that belong to sequences not subsumed by this pattern, e.g., #104, are untouched). (c) The identification of new significant patterns is done using the acquired equivalence classes (e.g., #200). In this manner, the system "bootstraps" itself, recursively distilling more and more complex patterns. This kind of abstraction also supports generalization: the original three sentences (shaded paths) form a pattern with two equivalence classes, which can then potentially generate six new sentences (e.g., the cat is play-ing and the horse is eat-ing).

to paths in the graph $path_i = \{c_{i1} \rightarrow \ldots, c_{ik}\}$, a path being a sequence of edges that share the same sentence number. Paths that correspond to entire sentences start at the 'BEGIN' node and end at the 'END' node.

In the second phase, the algorithm repeatedly scans the RDS graph for Significant Patterns (SP), which are then used to modify the graph. Each SP consists of a non-empty prefix (a sequence of graph edges), an equivalence class of vertices, and a non-empty suffix (another sequence of edges; cf. Figure 1(a)). For each path $path_i$, the algorithm constructs a set $s_j = \{p_1, \ldots, p_m\}$ of paths of the same length as $path_i$ that together could form a pattern. Of the many possible candidate sets, the algorithm analyzes all those whose size exceeds a certain fixed threshold. Each candidate set is assigned a score S that assesses its likelihood of capturing a significant regularity rather than a random fluctuation in the data. The definition of the score S (equation 1) combines a syntagmatic consideration (preferring longer paths) with a paradigmatic one (preferring informationally significant equivalence classes).

In equations 1-3, L is the length of the typical pattern the system is expected to acquire, k is the actual length of the candidate pattern, $P^{(k)}$ is the probability of the k^{th} order statistics (k-gram) over the path $\{c_1 \rightarrow \ldots \rightarrow c_k\}$ that is embedded in the graph, and $P^{(2)}$ is the probability of the second order statistics (bi-gram) of the same path in the graph.[2] Thus, $P^{(2)}$ corresponds to the probability of a random walker on the graph to have traversed the path $c_1 \rightarrow c_2, \ldots, \rightarrow c_k$, while $P^{(k)}$ corresponds to the probability of a walker with an unlimited memory span to complete the same path. To calculate the significance of the entire set of candidate paths, $P^{(k)}$ and $P^{(2)}$ should be summed over all the candidate paths in the set.

[2] $P^{(1)}(path_i)$ corresponds to the "first order" probability of choosing the set of nodes $c_1, \ldots, c_k$ without taking into account their sequential order along the path. Thus, $P^{(1)}(path_i) = P(c_1)P(c_2)P(c_3)\ldots P(c_k)$. $P^{(2)}$ is a better candidate for identifying significant *strings* (as opposed to mere sets of nodes), because it takes into account the sequence of nodes along the path.

$$S(path_i) = e^{-(L/k)^2} P^{(k)}(path_i) \log\left(P^{(k)}(path_i)/P^{(2)}(path_i)\right), \quad path_i \doteq c_1 \to c_2 \to \ldots \to c_k \quad (1)$$

$$P^{(k)}(path_i) = P(c_1)P(c_2|c_1)P(c_3|c_1 \to c_2)\ldots P(c_k|c_1 \to c_2 \to \ldots \to c_{k-1}) \quad (2)$$

$$P^{(2)}(path_i) = P(c_1)P(c_2|c_1)P(c_3|c_2)\ldots P(c_k|c_{k-1}) \quad (3)$$

These probabilities can be estimated from frequencies that are immediately available in the graph. Given the path $path_i = c_1 \to c_2 \to c_3 \to \ldots \to c_k$, its probability $P^{(k)}(path_i)$ is the product along the path of the probabilities $p(c_j|c_1 \to c_2 \to \ldots c_{j-1})$, $1 < j \leq k$, each of which is equal to the number of edges connecting the path $c_1 \to c_2 \to \ldots c_j$, divided by the number of edges connecting the path $c_1 \to c_2 \to \ldots c_{j-1}$ (see Figure 2). $P^{(2)}$ is the product along the path of the probabilities $p(c_j|c_{j-1})$, $1 < j \leq k$, each of which is equal to the number of edges connecting $c_{j-1} \to c_j$, divided by the total number of out-edges at node c_{j-1}.

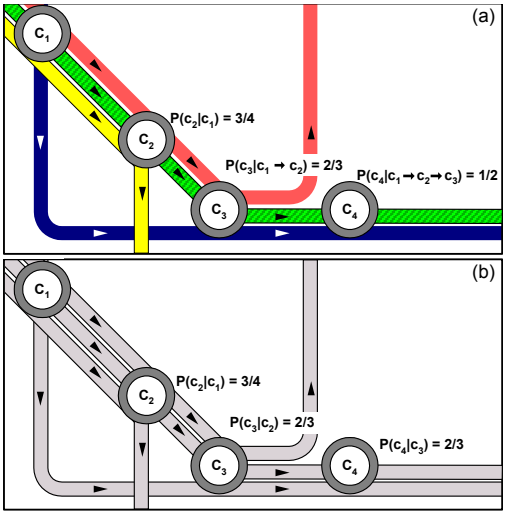

Figure 2: (a) The k-gram model used to calculate $P^{(k)}$, the probability of a walker starting at node c_1 to reach the node c_4 via the path $c_1 \to c_2 \to c_3 \to c_4$. The k^{th} order probability of a path of length k is the product of k conditional probabilities along the graph, each of which is equal to the number of paths that coincide along the exact sequence of nodes ending at c_j, $1 < j \leq k$, divided by the total number of paths that reach c_{j-1}. In this example $k = 4$, and $P^{(4)}(c_1 \to c_2 \to c_3 \to c_4) = \frac{1}{4}$. (b) The k-gram model can be compared to the bi-gram model, in which there is no information about the history of the walker moving along the graph (this is illustrated by using the same color for all the paths in the graph); $P^{(k)}$ is then calculated using equation 3. Here, $P^{(2)}(c_1 \to c_2 \to c_3 \to c_4) = \frac{1}{3}$.

The most significant set of candidate paths is now tagged as a Significant Pattern (its significance value S must exceed a fixed threshold α) and is added as a new vertex to the RDS graph, replacing the constituents and edges it subsumes (Figure 1(b)). Note that only those edges of the multi-graph that belong to the detected pattern are rewired; edges that belong to sequences not subsumed by the pattern are left intact. This highly context-sensitive approach to pattern abstraction, which is unique to our model, allows ADIOS to achieve a high degree of representational parsimony without sacrificing its generalization power.

During the pass over the corpus, the list of equivalence sets is updated continuously; new significant patterns are found using the *current* equivalence classes. For each set of candidate paths, the algorithm tries to fit one or more equivalence classes from the pool it maintains. Because a constituent can appear in several classes, the algorithm must check different combinations of equivalence classes. The winner combination is always the largest class for which most of the members are found among the candidate paths in the set (the ratio between the number of members that have been found among the paths and the total number of members in the equivalence class is compared to a fixed threshold as the configuration acceptance criterion). When not all the members appear in the existing set, the algorithm creates a new equivalence class containing only those members that did appear. Thus, as the algorithm processes more and more text, it "bootstraps" itself and enriches the RDS graph structure with new SPs and their accompanying equivalence sets. The recursive nature of this process enables the algorithm to form more and more complex patterns, in a hierarchical manner.

The relationships among the distilled patterns can be visualized in a tree format, with tree depth corresponding to the level of recursion (e.g., Figure 3). This tree can be seen as a blueprint for creating valid sequences of constituents (strings). The number of all possible string configurations can be estimated and compared to the number of examples seen in the training corpus. The reciprocal of their ratio, η, is the generalization factor, which can be calculated for each pattern in the RDS graph (e.g., in Figure 1(c), $\eta = 0.33$). Patterns whose significance score S and generalization factor η are beneath certain thresholds are rejected. The PA algorithm halts if it processes a given amount of text without finding a new SP or equivalence set (in real-life language acquisition this process may never stop).

A collection of patterns distilled from a corpus can be seen as an empirical grammar of sorts; cf. [5], p.63. The patterns can eventually become highly abstract, thus endowing the model with an ability to generalize to unseen

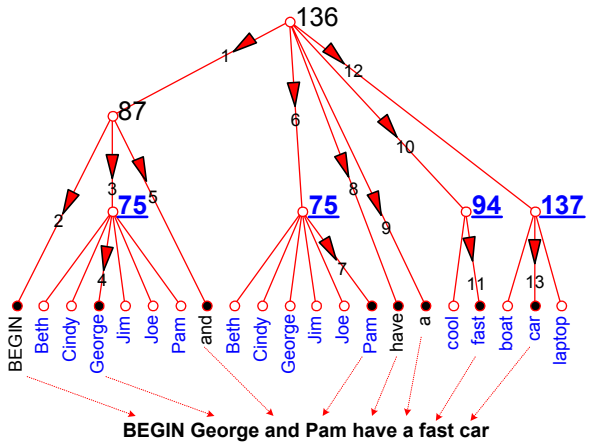

Figure 3: The outcome of the acquisition process is a set of significant patterns, which can be visualized recursively as a tree, whose depth corresponds to the level of recursion. The tree structure of a pattern can be seen as a blueprint for generating valid strings; here, the arrows illustrate the generation process – a depth-first search. For each non-terminal, the children are scanned from left to right; for each equivalence class (underscored numbers), one member is chosen. The scan continues from the node corresponding to that member, with the constituents reached at the terminal nodes being written out. The figure shows a pattern (#136) generated during training on an artificial Context Free Grammar corpus.

inputs. Generalization is possible, for example, when two equivalence classes are placed next to each other in a pattern, creating new paths among the members of the equivalence classes. Generalization can also ensue from partial activation of existing patterns by novel inputs. This function is supported by the *input module*, designed to process a novel sentence by forming its distributed representation in terms of activities of existing patterns (Figure 5). These are computed by propagating activation from bottom (the terminals) to top (the patterns) of the RDS. The initial activities w_j of the terminals c_j are calculated given the novel input $s_1, \ldots, s_k$ as follows:

$$w_j = \max_{l=1..k} \left\{ P(s_l, c_j) \log \frac{P(s_l, c_j)}{P(s_l)P(c_j)} \right\} \qquad (4)$$

where $P(s_l, c_j)$ is the joint probability that s_l and c_j will appear in the same equivalence class. while $P(s_l)$ and $P(c_j)$ are the probabilities that s_l and c_j will appear in any equivalence class. For an equivalence class, the value propagated upwards is the strongest non-zero activation of its members; for a pattern, it is the average weight of the children nodes, on the condition that all the children were activated by adjacent inputs. Activity propagation continues until it reaches the top nodes of the pattern lattice. When the algorithm encounters a novel word, all the members of the terminal equivalence class contribute a value of $\epsilon = 0.01$, which is then propagated upwards as usual. This enables the model to make an educated guess as to the meaning of the unfamiliar word, by considering the patterns that become active (Figure 5).

Results

We now briefly describe the results of several studies designed to evaluate the viability of the ADIOS model, in which it was exposed to corpora of varying size and complexity.

Emergence of syntactic structures. Figure 3 shows an example of a sentence from a corpus produced by a simple artificial grammar and its ADIOS analysis (the use of a simple grammar, constructed with Rmutt, http://www.schneertz.com/rmutt, in these initial experiments allowed us to examine various properties of the model on tightly controlled data). The abstract representation of the sample sentence in Figure 3 successfully identified the grammatical structure used to generate its data.

Generalization. To measure the capacity of the ADIOS algorithm for true learning, as distinguished from memorization or table lookup [10], we exposed the algorithm to an artificial corpus generated by a very simple finite-state grammar. Because this grammar is finite, so is the set of all possible sentences. We ran several learning sessions, in which we varied the proportion of sentences used to train the algorithm from 10% to 100%. Figure 4 presents the performance (the precision in accepting unseen sentences) versus the fraction of sentences in the training data. The model is seen to perform nearly perfectly after exposure to 20% of the corpus.

Novel inputs; systematicity. An important characteristic of a cognitive representation scheme is its systematicity, measured by the ability to deal properly with structurally related items (see [12] for a definition and discussion). We have assessed the systematicity of the ADIOS model by training the algorithm on the corpus generated by the grammar of Figure 3 and by examining the representations of unseen sentences. The general finding was of Level 3 systematicity according to the nomenclature of [12]. The ADIOS system's input module allows it to process a novel sentence by forming its distributed representation in terms of activities of existing patterns. Figure 5 shows the activation of pattern #185 by a phrase from the test set that contains three novel words, never before seen by the model.

Working with real data: the CHILDES corpus. To illustrate the scalability of our method, we describe here briefly the outcome of applying the PA algorithm to a subset of the CHILDES collection [13], which consists of transcribed speech produced by, or directed at, children. The corpus we selected contained 9665 sentences (74500 words) produced by parents. The results, one of which is shown in Figure 6, were encouraging: the algorithm found intuitively significant SPs and produced semantically adequate corresponding equivalence sets. Altogether, 317 patterns and 404 equivalence classes were established, representing the corpus in terms of these constituents.

S → N V N
S → N and N V N
S → N that N V
S → N and N that N V

N = {Joe, Beth, Jim, the cat, the dog, the cow}
V = {scolded, loved, adored, worshiped}

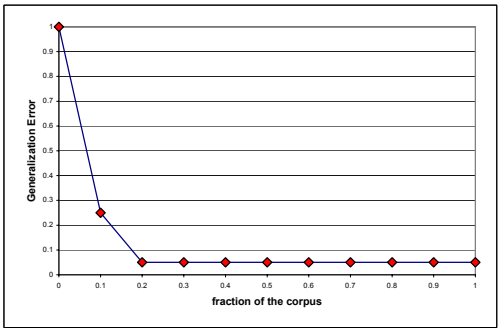

Figure 4: *Top:* a very simple finite-state grammar that can generate a corpus of 2016 phrases. *Bottom:* generalization error (a measure of precision), defined as $1-hits$, where $hits$ is the proportion of the total possible number of phrases that has been correctly accepted by the model, plotted vs. the fraction of the corpus used for training. In this case, the model performs nearly perfectly after exposure to 20% of the corpus. Note that this kind of test is highly relevant to psycholinguistic explorations of productivity, e.g., [11].

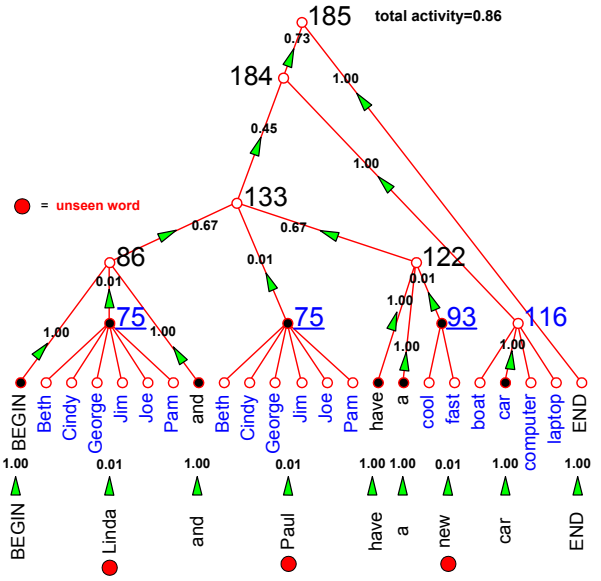

Figure 5: The input module in action; the most relevant (i.e., highly active) pattern responding to the novel input Linda and Paul have a new car. Leaf activation is determined by equation 4, then propagated up the tree by taking the average at each junction.

Concluding remarks

The ADIOS model learns (morpho)syntax on the basis of distributional information in the "raw" input, and supports the distillation of structural regularities (which can be thought of as constructions [4]) out of the accrued statistical knowledge. Although our pattern-based representations may look like collections of finite automata, the information they contain is much richer, because of the recursive invocation of one pattern by another, and because of the context sensitivity implied by relationships among patterns. Thus, unlike probabilistic finite automata such as Hidden Markov Models, ADIOS can learn Context Sensitive Languages. ADIOS is also better at dealing with the problem of sparse data, because it bootstraps the relevant statistics using the extracted equivalence classes. Furthermore, the recursive nature of the ADIOS algorithm allows it to capture longer-range dependencies than those found in any fixed-width window. Finally, the sensitivity to context of pattern abstraction (during learning) and use (during generation) contributes greatly both to the conciseness of the ADIOS representation and to the conservative nature of its generative behavior (note that ADIOS defines patterns in terms of specific morphemes or combinations of other patterns and equivalence classes, rather than in terms of idealized "part of speech" categories; cf. [4]). This context sensitivity — in particular, the manner whereby ADIOS balances syntagmatic and paradigmatic cues provided by the data — is mainly what distinguishes it from other current work on unsupervised probabilistic learning of syntax [8, 14, 15]. The present paper describes a viable algorithmic approach to unsupervised distributional learning of language patterns. The ultimate goal of this project is to achieve a computationally explicit, empirically proven, integrated understanding of three aspects of the representation of linguistic structures: theoretical, computational, and psychological.

Acknowledgments. Supported by the US-Israel Binational Science Foundation, the Dan David Prize Foundation, the Adams Super Center for Brain Studies at TAU, and the Horowitz Center for Complexity Science.

References

[1] Z. S. Harris. *Language and information*. Columbia University Press, New York, 1988.

[2] P. J. Hopper. Emergent grammar. In M. Tomasello, editor, *The new psychology of language*, pages 155–175. Erlbaum, Mahwah, NJ, 1998.

[3] J. Hutchens, M. Alder, and Y. Attikiouzel. Natural language grammatical inference. Technical Report HT94-03, University of Western Australia, 1994.

[4] W. Croft. *Radical Construction Grammar: syntactic theory in typological perspective.* Oxford University Press, Oxford, 2001.

[5] R. W. Langacker. *Foundations of cognitive grammar*, volume I: theoretical prerequisites. Stanford University Press, Stanford, CA, 1987.

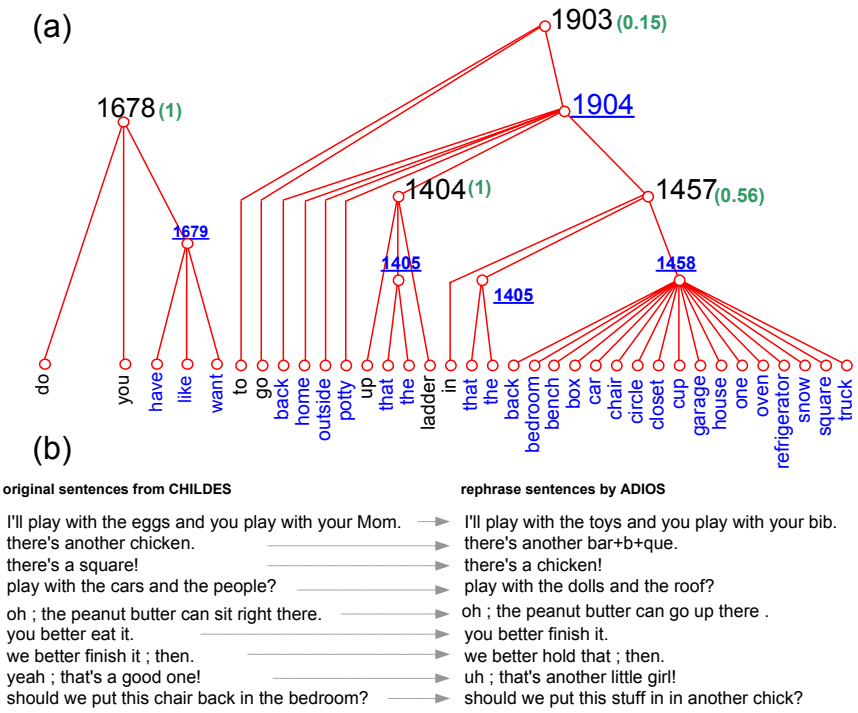

Figure 6: (a) Two strongly connected patterns extracted from a subset of the CHILDES collection [13]. Hundreds of such patterns and equivalence classes (underscored) together constitute a concise representation of the raw data. Some of the phrases that can be described/generated by patterns #1903 and #1678 are: do you have to go home?; do you like to go in that truck?; do you want to go up the ladder?; do you like to go the bench?. None of these sentences appear in the training data, illustrating the ability of ADIOS to generalize. The numbers in parentheses denote the generalization factor η of the patterns and their components (e.g., pattern #1903 generates 85% new strings, while pattern #1678 generates none). (b) Some of the phrases generated by ADIOS (left) using sentences from CHILDES (right) as examples. The generation module works by traversing the top-level pattern tree, stringing together lower-level patterns and selecting randomly one member from each equivalence class. Extensive testing (currently under way) is needed to determine whether the acceptability of the newly generated phrases (which is at present less than ideal) improves with more training data.

[6] R. L. Goldstone. Unitization during category learning. *Journal of Experimental Psychology: Human Perception and Performance*, 26:86–112, 2000.

[7] F. Pereira. Formal grammar and information theory: Together again? *Philosophical Transactions of the Royal Society*, 358(1769):1239–1253, 2000.

[8] M. van Zaanen and P. Adriaans. Comparing two unsupervised grammar induction systems: Alignment-based learning vs. EMILE. Report 05, School of Computing, Leeds University, 2001.

[9] M. Gross. The construction of local grammars. In E. Roche and Y. Schabès, editors, *Finite-State Language Processing*, pages 329–354. MIT Press, Cambridge, MA, 1997.

[10] S. Patarnello and P. Carnevali. Learning networks of neurons with Boolean logic. *Europhysics Letters*, 4:503–508, 1987.

[11] R. L. Gómez and L. Gerken. Infant artificial language learning and language acquisition. *Trends in Cognitive Science*, 6:178–186, 2002.

[12] T. J. van Gelder and L. Niklasson. On being systematically connectionist. *Mind and Language*, 9:288–302, 1994.

[13] B. MacWhinney and C. Snow. The child language exchange system. *Journal of Computational Linguistics*, 12:271–296, 1985.

[14] D. Klein and C. D. Manning. Natural language grammar induction using a constituent-context model. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[15] A. Clark. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, COGS, University of Sussex, 2001.