Learning, using, and analyzing graph grammars of hierarchical sequential behaviors

Shimon Edelman

Cornell University
Lashley’s problem of “serial order in behavior”

“Certainly language presents in a most striking form the integrative functions that are characteristic of the cerebral cortex and that reach their highest development in human thought processes. Temporal integration is not found exclusively in language: the coordination of leg movements in insects, the song of birds, the control of trotting and pacing in a gaited horse, the rat running the maze, the architect designing a house, the carpenter sawing a board present a problem of sequences of action ...”

— Karl Lashley (1951)
episodic memories of sequential experience
episodic memories of sequential experience

look at these words carefully now
please read once again
what does it mean to learn language

Zellig S. Harris (1909-1992):
“When only a small percentage of all possible sound-sequences actually occurs in utterances, one can identify the boundaries of words [and phrases], and their relative likelihoods, from their sentential environment; this, even if one was not told (in words) that there exist such things as words.”

Thus, one can learn a generative model of the regularities —
\[ P(\text{utterances, situations, intents}) \]
— a “grammar.”

“It follows that [...] a fundamental task is to state
the departures from equiprobability [= statistics] in sound- and word-sequences.”
a useful analogy for “serial order in behavior”

A corpus of [language] behavioral data is like... a graph!
when a corpus of language is analyzed,...
... the structure of language emerges

I would like to know
you would like to hear

collocactions
... the structure of language emerges

structure = statistical dependencies (departures from equiprobability) defined over the corpus graph
... the structure of language emerges

structure = statistical dependencies (departures from equiprobability) defined over the corpus graph
... the structure of language emerges

structure = statistical dependencies (departures from equiprobability) defined over the corpus graph
experience/behavior as a graph

<table>
<thead>
<tr>
<th>LEXICON-GRAMMAR</th>
<th>↔ the entire transit system</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRAMMATICAL UTTERANCES</td>
<td>↔ possible trips</td>
</tr>
<tr>
<td>LEARNING</td>
<td>↔ constructing the transit system</td>
</tr>
<tr>
<td>PROBABILISTIC SELECTION</td>
<td>↔ traffic flow</td>
</tr>
<tr>
<td>STRUCTURE VS. STATISTICS</td>
<td>↔ routes vs. traffic</td>
</tr>
<tr>
<td>FLUENCY AND DISFLUENCY</td>
<td>↔ navigation proficiency</td>
</tr>
<tr>
<td>MEANING</td>
<td>↔ the outside world</td>
</tr>
</tbody>
</table>
what a (hi)graph grammar looks like

Learning a graph-structured generative probabilistic grammar of linguistic experience
Kolodny, Lotem, & Edelman (2015)
The basic computational operations: alignment and comparison
re alignment+comparison: variation sets
English:
- those are checkers
- two checkers yes
- play checkers
English:
those are checkers
two checkers yes
play checkers

Italian:
dove sono
dove sono i coniglietti
English:
those are checkers
two checkers yes
play checkers

Russian:
vot tvoi papa ne xochet s toбой ostаться
kak ne xochet xochet
xochet papa xochet

Italian:
dove sono
dove sono i coniglietti
<table>
<thead>
<tr>
<th>English:</th>
<th>Hebrew:</th>
</tr>
</thead>
<tbody>
<tr>
<td>those are checkers</td>
<td>מה לא רוצה את אתי להספר</td>
</tr>
<tr>
<td>two checkers yes</td>
<td>ahu רצות לא לספר</td>
</tr>
<tr>
<td>play checkers</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Russian:</th>
<th>Italian:</th>
</tr>
</thead>
<tbody>
<tr>
<td>вот твой папа не хочет с тобой остаться</td>
<td>dove sono</td>
</tr>
<tr>
<td>как не хочет хочет</td>
<td>dove sono i coniglietti</td>
</tr>
<tr>
<td>хочет папа хочет</td>
<td></td>
</tr>
</tbody>
</table>
[re alignment+comparison: variation sets]

**English:**
those are checkers  
two checkers yes  
play checkers

**Hebrew:**
מה לא  
מה את לא רוצה  
את רוצה לספר

**Russian:**
вот твой папа не хочет с тобой остаться  
как не хочет хочет  
хочет папа хочет

**Italian:**
dove sono  
dove sono i coniglietti

**Korean:**
제일 이뻐  
누가 제일 이뻐  
지원이 제일 이뻐 맞아
### English:
- those are checkers
- two checkers yes
- play checkers

### Russian:
- вот твой папа не хочет с тобой остаться
- как не хочет хочет
- хочет папа хочет

### Hebrew:
- לא
- מה לא רוצה
- את רוצה לספר

### Italian:
- dove sono
- dove sono i coniglietti

### Korean:
- 제일 이뻐
- 누가 제일 이뻐
- 지원이 제일 이뻐 맞어

### Mandarin:
- 这是 什么 呀
- 哎呀 是 什么 呀

(Heidi Waterfall)
learning a graph grammar: algorithm outline

1. **Add** to graph: if not encountered previously, add the token to the graph as a base node.

2. **Update** short-term memory and **search** for alignments (top-down segmentation):
   - Insert the new token into the short term memory queue.
   - Create a list of newly-completed alignments within the queue.
   - Add each element in the list to the graph with probability inversely proportional to the memory decay factor and the distance to existing elements.
   - If a newly-added sequence contains an existing one, remove the shorter sequence.

3. **Update** temporal relations and **construct** collocations (bottom-up chunking):
   - Create a list of all nodes in the graph that terminate the short-term memory sequence.
   - Create a secondary list of sequences that fill the slot of slot-collocations in the primary list.
   - Update or add temporal edges between each node in the current list (X) and the nodes in a previously found list that contains the nodes preceding X.
   - Update slot-candidacy edges of all nodes that are within slot-collocations in the primary list.
   - For each pair of nodes A,B between which a temporal link has been updated, create a new supernode, A+B, if sanctioned by Barlow’s (1990) principle of suspicious coincidence, subject to a prior.
Four different **node addition** modes —

1. **Flat Markov**: only base tokens are added to the graph (thus no hierarchical structures, hence “flat”);
2. **Phonoloop collocation**: add base tokens and recurring sequences found in the short-term memory;
3. **Bottom-up collocation**: add base tokens and significant runs of adjacent units;
4. **Normal**: all of the above.

When a new node is added to the graph, a search is conducted for **existing nodes** whose content overlaps with that of the new node. Pointer updates take place accordingly if the new node is found to be a super-node or a sub-node of any of them.

The **similarity** between two nodes is calculated as a weighted average of the following three measures:

1. All temporal edges;
2. Common occurrence in slot;
3. Within-slot interchangeability within a short time window.
1. Begin:
   • With a very low probability $P_{\text{rand}}$, choose a node from the graph at random with probability proportional to its weight (= smoothing);
   else:
   • Choose from among the successors of the current node, biasing toward nodes that contain longer sequences.

2. With probability $P_{\text{generalize}}$, replace the node by another node, chosen proportionally to its similarity (substitutability) to the node chosen in (1).

3. If the chosen node contains a slot, choose with probability $P_{\text{rand}}$ a filler from among all the nodes in the graph with proportion to their weight; choose with probability $1 - P_{\text{rand}}$ a filler from among the slot-filler candidates in the slot, with proportion to weights of the slot-candidacy edges.
To estimate the probability of producing a sentence, the model must find all possible covers of it in terms of paths through the graph.

The model repeatedly attempts to cover the sentence using a number of nodes ranging from 1 to the number of base tokens in the sentence.

In each attempt, a search routine finds all the possible single-node covers of the beginning of the sentence, then for each of these recursively calls itself on the remainder of the sentence, until it finds a complete cover or determines that such a cover does not exist (cf. left-corner parsing: Resnik, 1992).

Once all complete covers of the sentence are found, the probability of production of each of these is calculated. The probability assigned to the sentences is the sum of production probabilities of all covers.
testing the model

Study 1: Measures of generative ability of a grammar learned from a corpus of natural language: perplexity and precision.

Study 2: Characteristics of the learned representation: equivalence (substitutability) of phrases and the similarity structure of the phrase space.

Study 3: Replication of a variety of results in sequence segmentation and chunking.

Study 4: Replication of results in artificial grammar learning.

Study 5: Replication of results regarding certain types of structure dependence.
perplexity and precision

**Training corpus:** the first 15,000 utterances (81,370 word tokens) of the Suppes corpus of transcribed child-directed speech, from the CHILDES collection (MacWhinney, 2000; Suppes, 1974). Adult-produced utterances only were used.

**Perplexity:** 40.07.
Control: perplexity of a trigram model (SRILM; Stolcke, 2002):
- Good-Turing smoothing — 24.36;
- Kneser-Ney smoothing — 22.43.

There is a tradeoff between perplexity and precision; the precision of the tri-gram model was much worse (see next).

By modifying our model’s similarity-based generalization and smoothing parameters, perplexity could be reduced to as low as 34, at a cost to the precision.

At the other extreme, with the similarity-based generalization parameter set to zero, the perplexity rises to 60.04.
Precision: 5.87 out of 7
“lexical” similarity

MDS plot for words—only with frequency between 95 and 100 percentiles
“lexical” similarity

MDS plot for words—only with frequency between 75 and 80 percentiles
We present the 20 most frequent nodes, because the statistics are the most extensive, and so their categories are likely to be meaningful, and 11 examples of slightly less frequent nodes, which provide some insight into the model’s categorization (see main text). The symbol that appears as a node or as part of a node pertains to the sequence’s, which is transcribed in the corpus as a stand-alone, as in that’s a bunny.
distinguishing between words and part-words

**TASK:**
Given a corpus of sentences, learn to distinguish words that comprise them.

![Graph showing sentence length vs. score with BAGEL model](Frank et al., 2010, exp.1)
distinguishing between words and part-words

(Frank et al., 2010, exp.1)
words vs. non-words (phonetic encoding)

**TASK:**
Given a corpus of phonetically encoded natural child-directed speech, learn to distinguish words from nonwords.

(French et al., 2011, exp.8)
words vs. non-words (phonetic encoding)

(French et al., 2011, exp.8)
distinguishing words from two languages

**TASK:**
Given a mixed corpus consisting of words from two “languages”, learn to distinguish L1 words from L2 words.

(French et al., 2011, exp.10)
distinguishing words from two languages

(French et al., 2011, exp.10)
words vs. non-words (backward stats)

**TASK:**
Given a mixed corpus consisting of words from two “languages”, defined by backward transition probabilities, learn to distinguish L1 words from L2 words.

(French et al., 2011, exp.11)
words vs. non-words (backward stats)

(French et al., 2011, exp.11)
<table>
<thead>
<tr>
<th>Language 1</th>
<th>Language 2</th>
<th>Test strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → { aXd, bXe, cXf }</td>
<td>S → { aXe, bXf, cXd }</td>
<td>Language 1</td>
</tr>
<tr>
<td>X → x₁, x₂, …, xₙ ; n = 2, 6, 12 or 24</td>
<td></td>
<td>pel wadim rud</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vot wadim jic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dak wadim tood</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pel kicey rud</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vot kicey jic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dak kicey tood</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Language 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pel wadim jic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vot wadim tood</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dak wadim rud</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pel kicey jic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>vot kicey tood</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dak kicey rud</td>
</tr>
</tbody>
</table>

(Gomez, 2002, exp.1)
grammatical vs. ungrammatical strings

(Gomez, 2002, exp.1)
grammatical vs. ungrammatical strings

Sentences in L1: aX or bY, where a, b, X, and Y are word categories.
Sentences in L2: aY or bX.

\[
\begin{array}{cccc}
a & b & X & Y \\
alt & ong & coo mo & deech \\
ush & erd & fen gle & ghope \\
& ki cey & lo ga & jic \\
& pay lig & wa zil & skige \\
& & & vabe \\
& & & tam \\
\end{array}
\]

(Gomez & Lakusta, 2004)
grammatical vs. ungrammatical strings

(Gomez & Lakusta, 2004)
variation set effect

**TASK:**
Given a corpus of the same sentences appearing in a random order (scrambled) or in variation sets, learn the words.

(Onnis et al., 2008, exp.1)
variation set effect

(Onnis et al., 2008, exp.1)
auxiliary verb fronting

**TASK:**
Given a corpus of child-directed language, learn to distinguish correct / incorrect auxiliary verb fronting.

*Is the bunny that is on the chair sleeping?*

*(Reali & Christiansen, 2005)*
auxiliary verb fronting

(Reali & Christiansen, 2005)
lessons

In modeling language, opting for a very powerful formalism (ATNs or Live Sequence Charts) may actually pay off, given that

- it seems natural given the data;
- it seems natural given the known constraints on the learner’s implementation (brains),

as long as

- the learner is highly conservative;
- the learner makes use of all the available cues;
- the learner does not strive for the impossible (such as acquiring *precisely* the grammar shared by the community)
looking for structure in zebra finch songs

corpus

iiiiiaiiabcfe
iabcfd
iiabcfd
iiabbfdiaabcfd
iiiiabcfdiaabcfd
iiabcfd
iiabcfdiaabcfd
iiabcfdhgbaabcfd
iiiiabcfdhgbaabcfd
iiabcfe
iiia
iiia
iiabcfe
iabcfd
iabcfd
iiabcfd
iiabcfd
iiabcfd
iiabcfd
iiabcfd
looking for structure in zebra finch songs

<table>
<thead>
<tr>
<th>corpus</th>
<th>lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>iiiiaiiabcfed</td>
<td>BEGIN</td>
</tr>
<tr>
<td>iabcf</td>
<td>END</td>
</tr>
<tr>
<td>iabcf</td>
<td>g</td>
</tr>
<tr>
<td>iabcf</td>
<td>c</td>
</tr>
<tr>
<td>iabcf</td>
<td>e</td>
</tr>
<tr>
<td>iabcf</td>
<td>iabcf</td>
</tr>
<tr>
<td>iabcf</td>
<td>iabcf</td>
</tr>
<tr>
<td>iabcf</td>
<td>iabcf</td>
</tr>
<tr>
<td>iabcf</td>
<td>iabcf</td>
</tr>
<tr>
<td>iabcf</td>
<td>iabcf</td>
</tr>
<tr>
<td>iabcf</td>
<td>h</td>
</tr>
</tbody>
</table>
zebra finch song grammar(s)

grammar: U-MILA (also Sequitur; ADIOS)
which is the right grammar to use?

One possibility:

The best grammar is the one that best predicts a withheld part of the training corpus.
which is the right grammar to use?

One possibility:

The best grammar is the one that best predicts a withheld part of the training corpus.

Use the LOO: Leave One Out cross-validation.
quantifying the **structural similarity** of songs

One possibility:

Consider the structure of the graph grammar that best describes the song.
quantifying the **structural similarity** of songs

One possibility:

Consider the structure of the graph grammar that best describes the song.

Use spectral graph theory.
quantifying the **structural similarity** of songs

One possibility:

Consider the structure of the graph grammar that best describes the song.

Use spectral graph theory.

One application / question to ask:

Is a bird’s song more similar to that of its father/tutor than to that of a random unrelated male?
The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j) = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases}$$
The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j) = \begin{cases} 
1 & \text{if }(i, j) \in E \\
0 & \text{otherwise.}
\end{cases}$$
quantifying graph structure: the spectral theory

The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j) = \begin{cases} 
1 & \text{if } (i, j) \in E \\
0 & \text{otherwise.}
\end{cases}$$
The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j)$$

Directed Cayley graph of $S_4$

As the graph is directed, the matrix is not symmetric.
The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j) = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise.} \end{cases}$$
The properties of an operator naturally associated with a graph $G$, and therefore of $G$ itself, are revealed by its *spectrum* of eigenvalues and eigenvectors.

One such operator is the adjacency matrix $A_G$:

$$A_G(i, j) = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$

For a weighted graph:

$$A_G(a, b) = \begin{cases} w(a, b) & \text{if } (a, b) \in E \\ 0 & \text{otherwise} \end{cases}$$
adjacency matrix quantifies activation spread
adjacency matrix quantifies activation spread

infect activation

<table>
<thead>
<tr>
<th>Labeled graph</th>
<th>Adjacency matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Labeled Graph" /></td>
<td><img src="image" alt="Adjacency Matrix" /></td>
</tr>
</tbody>
</table>

Coordinates are 1-6.
Eigenvalues and eigenvectors are used to understand what happens when one repeatedly applies an operator to a vector. If $A$ is an $n$-by-$n$ matrix having a basis of right-eigenvectors $v_1, \ldots, v_n$ with

$$A v_i = \lambda_i v_i$$

then we can use these eigenvectors to understand the impact of multiplying a vector $x$ by $A$. We first express $x$ in the eigenbasis

$$x = \sum_i c_i v_i$$

and then compute

$$A^k x = \sum_i c_i A^k v_i = \sum_i c_i \lambda_i^k v_i$$
father

son

stranger
zebra finch song grammar(s): COL+i

Kruskal–Wallis [dSame < dDiff]: $p < 0.003$

Per–pair sign–rank tests significant: 21 out of 31
(better than chance at $p=0.035378$, binomial)
Models of language acquisition are beginning to exhibit impressive performance (precision+recall) and psycholinguistic realism.

Some of those models, such as U-MILA, are directly applicable to the learning of any other type of symbolic sequential hierarchical structure.

An application of the U-MILA model to zebra finch song corpora reveals non-trivial grammar structure.

The best — most generative/predictive — such grammar takes the form of a probabilistically weighted, hierarchically structured graph defined over learned collocations (COL).

Using spectral distance over the graph grammars corresponding to different individuals, we found greater structural similarity between the songs of tutors and tutees (e.g., fathers and sons) than between songs of unrelated individuals.
thanks to...

The U-MILA model

Oren Kolodny
Arnon Lotem

The zebra finch data

Mike Goldstein
Tim DeVoogd
Otilia Menyhart
Language is part of multimodal, embodied, spatially situated, socially engaged experience.
Language is part of multimodal, embodied, spatially situated, socially engaged experience.

This makes learning it considerably easier.
Language is part of multimodal, embodied, spatially situated, socially engaged experience. This makes learning it considerably easier.

**General cognitive principles for learning structure in time and space**
what next?

“I was wondering when you’d notice there’s lots more steps.”