Simple Recurrent Networks

- Until now we’ve consider “Static” models: Map a single, isolated, input to a particular output

- Dynamical Systems: Simple Recurrent Networks
  - Sequential XOR
  - Letter sequences
  - Detecting word boundaries
  - Learning lexical classes

- Acquisition of Syntax
  - Mapping sentences to meaning, generating sentence from meanings
  - Relating SRNs to Surprisal, Neurophysiological measures, and neuroanatomical models
The network architecture has 6 input and output units, with 20 hidden and context units.

Training:
- Each input vector is presented
- Trained to predict the next input
- 200 passes through the sequence

Tested on another random sequence (using same rules)

Error for part of the test is shown in the graph
- Low error predicting vowels
- High error on consonants

But this is the global pattern error for the 6 bit vector …

Deeper analysis of performance

Can predict which vowel follows a consonant, and how many (?)

Bit 1, represents the feature **Consonant** and bit 4 represents **High**
- All consonants have the same feature for Consonant, but not for High

Thus the network has also learned that after the correct number of vowels, it expects **some** consonant: This requires the context units
Predicting the next sound

- High error at the onset of words
- Decreases during a word, as the sequence is increasingly predictable
- High error at word onset demonstrates the network has “discovered” word boundaries

Structure of Training Environment

### Categories of lexical items

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOUN-HUM</td>
<td>man, woman</td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>cat, mouse</td>
</tr>
<tr>
<td>NOUN-INANIM</td>
<td>book, rock</td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>dragon, monster</td>
</tr>
<tr>
<td>NOUN-FRAG</td>
<td>glass, plate</td>
</tr>
<tr>
<td>NOUN-FOOD</td>
<td>cookie, sandwich</td>
</tr>
<tr>
<td>VERB-INTRAN</td>
<td>think, sleep</td>
</tr>
<tr>
<td>VERB-TRAN</td>
<td>see, chase</td>
</tr>
<tr>
<td>VERB-AGPAT</td>
<td>move, break</td>
</tr>
<tr>
<td>VERB-PERCEPT</td>
<td>smell, see</td>
</tr>
<tr>
<td>VERB-DESTROY</td>
<td>break, smash</td>
</tr>
<tr>
<td>VERB-EAT</td>
<td>eat</td>
</tr>
</tbody>
</table>

### Template for sentence generator

<table>
<thead>
<tr>
<th></th>
<th>WORD 1</th>
<th>WORD 2</th>
<th>WORD 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOUN-HUM</td>
<td>VERB-EAT</td>
<td>NOUN-FOOD</td>
<td></td>
</tr>
<tr>
<td>NOUN-HUM</td>
<td>VERB-PERCEPT</td>
<td>NOUN-INANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-HUM</td>
<td>VERB-DESTROY</td>
<td>NOUN-FRAG</td>
<td></td>
</tr>
<tr>
<td>NOUN-HUM</td>
<td>VERB-INTRAN</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-HUM</td>
<td>VERB-TRAN</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-HUM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>VERB-EAT</td>
<td>NOUN-FOOD</td>
<td></td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>VERB-TRAN</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-ANIM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-INANIM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-INANIM</td>
<td>VERB-AGPAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-DESTROY</td>
<td>NOUN-FRAG</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-EAT</td>
<td>NOUN-HUM</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-EAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-EAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-EAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
<tr>
<td>NOUN-AGRESS</td>
<td>VERB-EAT</td>
<td>NOUN-ANIM</td>
<td></td>
</tr>
</tbody>
</table>
### Input encoding & training

- Localist representation of each word (31 bits)
  - Nothing of the word class is reflected
- 10000 random 2-3 word sentences
  - 27,354 sequence of 31 bit vectors
- Architecture:
  - Trained on 6 complete passes through the sequence

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000000000000</td>
<td>000000000000000010000</td>
</tr>
<tr>
<td>(woman)</td>
<td>(smash)</td>
</tr>
<tr>
<td>0000000000000000001</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(smash)</td>
<td>(plate)</td>
</tr>
<tr>
<td>0000000000000000000</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(plate)</td>
<td>(cat)</td>
</tr>
<tr>
<td>0000000000000000000</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(cat)</td>
<td>(dog)</td>
</tr>
<tr>
<td>0000000000000000000</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(dog)</td>
<td>(mouse)</td>
</tr>
<tr>
<td>0000000000000000000</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(mouse)</td>
<td>(book)</td>
</tr>
<tr>
<td>0000000000000000000</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>(book)</td>
<td>(lion)</td>
</tr>
</tbody>
</table>

### Performance

- Training yields an RMS error of 0.88
- RMS error rapidly drops from 15.5 to 1, by simply learning to turn all outputs off (due to sparse, localist representations). Careful about looking at RMS alone!
- Prediction is non-deterministic: next input cannot be predicted with absolute certainty, but neither is it random
  - Word order and selectional restrictions partially constrain what words are likely to appear next, and which cannot appear.
  - We would expect the network to learn the frequency of occurrence of each possible successor, for a given input sequence
- Output bit should be activated for all possible following words
  - These output activations should be proportional to frequency
- Evaluation procedure:
  - Compare network output to the vector of probabilities for each possible next word, given the current word and context …
Calculating Performance

- Output should be compared to expected frequencies.

- Frequencies are determined from the training corpus:
  - Each word ($w_{\text{input}}$) in a sentence is compared with all other sentences that are up to that point identical (comparison set).
    - *Woman smash plate*
    - *Woman smash glass*
    - *Woman smash plate*
    - ...
  - Compute a vector of the probability of occurrence for each following word: this is the target, output for a particular input sequence.
    - Vector: \{0 0 0 p(plate|smash, woman) 0 0 p(glass|smash, woman) 0 ... 0 \}
  - This is compared to the output vector of the network, when the word *smash* is presented following the word *woman*.

- When performance is evaluated this way, RMS is 0.053.
  - Mean cosine of the angle between output and probability: 0.916
    - This corrects for the fact that the probability vector will necessarily have a magnitude of 1, while the output activation vector need not.

Remarks on performance

- Inputs contain no information about form class (orthogonal representations) which can be used for making predictions.
  - Generalisations about the distribution of form classes, and the composition of those classes, must be learned from co-occurrence.
  - We might therefore expect these generalisations to be captured by the hidden unit activations evoked by each word in its context.

- After 6 passes, connection strengths were “frozen.”

- The corpus was then presented to the network again: outputs ignored.
  - Hidden unit activations for each input + context were saved.
    - 27354, 150 bit vectors
  - The hidden unit vectors for each word, in all contexts, were averaged.
    - Yielding 29, 150 bit vectors

- The resulting vectors were clustered hierarchically …
Cluster analysis:

- Lexical items with similar properties are grouped lower in the tree

- The network has discovered:
  - Nouns vs. Verbs
  - Verb subcategorization
  - Animates/inanimates
  - Humans/Animals
  - Foods/Breakables/Objects
  - The network discovers ordering possibilities for various work categories and “subcategories”

Type-Token distinction

- Both symbolic systems and connectionist networks use representations to refer to things:
  - Symbolic systems use names
    - Symbols typically refer to well-defined classes or categories of entities
  - Networks use patterns of activations across hidden-units
    - Representations are highly context dependent

- The central role of context in SRNs results in a distinct representation of John, for every context in with John occurs (i.e. an infinite number of John)

- Claim: contextualised distributed representations provides a solution to the representation of type/token differences
  - Distributed representations can learn new concepts as a patterns of activations across a fixed number of hidden unit nodes
    - I.e. A fixed number of analogue units can in principle learn an infinite number of concepts
  - Since SRN hidden units encode prior context, the hidden unit can in principle provide an infinite memory
Type/Token continued

- In practice the number of concepts and memory is bounded
  - Units are not truly continuous (e.g. numeric precision on the computer)
  - Repeated application of logistic function to the memory results in exponential decay
  - Training environment may not be optimal for exploiting network capacity
  - Actually representational capacity remains an open question

- The SRN representation reflect aspects of word meaning and category
  - Apparent in the similarity structure of the “averaged” internal representation of each word: the network’s representation of the word types

- The network also distinguishes between specific occurrences of words
  - The internal representation for each token of a word are very similar
  - But do subtly distinguish between the same word in different contexts

- Thus SRNs provide a potentially interesting account of the type-token distinction, differs from the indexing/binding operations of symbol systems

---

Clustering of word “tokens”

- Hierarchical clustering of specific occurrences of BOY and GIRL
Summary of Elman 1990

• Some problems change their nature when expressed as temporally:
  • E.g. sequential XOR developed frequency sensitive units

• Time varying error signal can be a clue to temporal structure:
  • Lower error in prediction suggests structure exists

• Increased sequential dependencies don’t result in worse performance:
  • Longer, more variable sequences were successfully learned
  • Also, the network was able to make partial predictions (e.g. “consonant”)

• The representation of time and memory is task dependent:
  • Networks intermix immediate task, with performing a task over time
  • No explicit representation of time: rather “processing in context”
  • Memory is bound up inextricably with the processing mechanisms

• Representation need not be flat, atomistic or unstructured:
  • Sequential inputs give rise to “hierarchical” internal representations

  “SRNs can discover rich representations implicit in many tasks, including structure which unfolds over time”

Challenges for a connectionist account

• What is the nature of connectionist linguistic representations?
  • Localist representations seem too limited (fixed and simplistic)
  • Distributed have greater capacity, can be learned, are poorly understood

• How can complex structural relationships such as constituency be represented?
  Consider “noun” versus “subject” versus “role”:
  • The boy broke the window
  • The rock broke the window
  • The window broke

• How can “open-ended” language be accommodated by a fixed resource system?
  • Especially problematic for localist representations

• In a famous article, Fodor & Pylyshyn argue that connectionist models:
  • Cannot account for the fully compositional structure/nature of language
  • Cannot provide for the open-ended generative capacity
Learning Linguistic Structure

• Construct a language, generated by a grammar which enforces diverse linguistic constraints:
  • Subcategorisation
  • Recursive embedding
  • Long-distance dependencies

• Training the network:
  • Prediction task
  • Is structuring of the training data/procedure necessary?

• Assess the performance:
  • Evaluation of predictions (as in Elman 1990), not RMS error
  • Cluster analysis? Only reveals similarity of words, not the dynamics of processing
  • Principle component analysis: the role of specific hidden units

Learning Constituency: Elman (1991)

• So far, we have seen how SRNs can find structure in sequences

• How can complex structural relationships such as constituency be represented?

• The Stimuli:
  • Lexicon of 23 items
  • Encoded orthogonally, in 26 bit vector

• Grammar:
  • S → NP VP "."
  • NP → PropN | N | N RC
  • VP → V (NP)
  • RC → who NP VP | who VP (NP)
  • N → boy | girl | cat | dog | boys | girls | cats | dogs
  • PropN → John | Mary
  • V → chase | feed | see | hear | walk | live | chases | feeds | sees | hears | walks | lives

  • Number agreement, verb argument patterns
Training

- Verb subcategorization
  - Transitives: *hit, feed*
  - Optional transitives: *see, hear*
  - Intransitives: *walk, live*

- Interaction with relative clauses:
  - Dog *who chases cat sees girl*
  - Dog *who cat chases sees girl*
  - Agreement can span arbitrary distance
  - Subcategorization doesn’t always hold (locally)

- Recursion: Boys *who girls who dogs chase see hear*

- Viable sentences: where should end of sentence occur?
  - Boys see () dogs () who see () girls () who hear () .

- Words are not explicitly encoded for number, subcat, or category

---

Training: Starting Small

- At any given point, the training set contained 10000 sentences, which were presented to the network 5 times

- The composition of sentences varied over time:
  - Phase 1: Only simple sentences (no relative clauses)
    - 34,605 words forming 10000 sentences
  - Phase 2: 25% complex and 75% simple
    - Sentence length from 3-13 words, mean: 3.92
  - Phase 3: 50/50, mean sentence length 4.38
  - Phase 4: 75% complex, 25% simple, max: 16, mean: 6

- WHY?: Pilot simulations showed the network was unable to learn successfully when given the full range of complex data from the beginning.

- Focussing on simpler data first, the network learned quickly, and was then able to learn the more complex patterns.

- Earlier simple learning, usefully constrained later learning
Performance

• Weights are frozen and test on a novel set of data (as in phase 4).

• Since the solution is non-deterministic, the networks outputs were compared the context dependent likelihood vector of all words following the current input (as done in the previous simulation)
  • Error was 0.177, mean cosine: 0.852
  • High level of performance in prediction

• Performance on Specific Inputs

  • Simple agreement: BOY ..
  • BOYS ..

Subcategorization

• Intransitive: “Boy lives …”
  • Must be a sentence, period expected

• Optional: “Boy sees …”
  • Can be followed by either a period,
  • Or some NP

• Transitive: “Boy chases …”
  • Requires some object
Processing complex sentences

- “Boys who marry chases feed cats”
  - Long distance
    - Agreement: Boys … feed
    - Subcategorization: chases is transitive but in a relative clause
    - Sentence end: all outstanding “expectations” must be resolved

Prediction reconsidered

- SRNs are trained on the **prediction** task:
  - “Self-supervised learning”: no other teacher required

- Prediction forces the network to discover regularities in the temporal order of the input

- Validity of the prediction task:
  - It is clearly not the “goal” of linguistic competence
  - But there is evidence that people can/do make predictions
  - Violated expectation results in distinct patterns of brain activity (ERPs)

- If children do make predictions, which are then falsified, this might constitute an indirect form of negative evidence, required for language learning.
Results

- Learning was only possible when the network was forced to begin with simpler input
  - Restricted the range of data the networks were exposed to during initial learning
  - Contrasts with other results showing the entire dataset is necessary to avoid getting stuck in local minima (e.g. XOR)

- This behaviour partially resembles that of children:
  - Children do not begin by mastering language in all its complexity
  - They begin with simplest structures, incrementally building their “grammar”

- But the simulation achieves this by manipulation the environment:
  - Does not seem an accurate model of the situation in which children learn language
  - While adults do modify their speech, it is not clear they make grammatical modifications
  - Children *hear* all exemplars of language from the beginning

General results

- Limitations of the simulations/results:
  - Memory capacity remains un-probed
  - Generalisation is not really tested
    - Can the network inferentially extend what is know about the types of NPs learned to NPs with different structures
    - Truly a “toy” in terms of real linguistic complexity and subtlety
      - E.g. lexical ambiguity, verb-argument structures, structural complexity and constraints

- Successes
  - Representations are distributed, which means less rigid resource bounds
  - Context sensitivity, but can respond to contexts which are more “abstractly” defined
    - Thus can exhibit more general, abstract behaviour
    - Symbolic models are primarily context insensitive

- Connectionist models begin with local, context sensitive observations

- Symbolic models begin with generalisation and abstractions
A Second Simulation

• While it’s not the case that the environment changes, it true that the child changes during the language acquisition period

• Solution: keep the environment constant, but allow the network to undergo change during learning

• Incremental memory:
  • Evidence of a gradual increase in memory and attention span in children
  • In the SRN, memory is supplied by the “context” units
  • Memory can be explicitly limited by depriving the network, periodically, access to this feedback

• In a second simulation, training began with limited memory span which was gradually increased:
  • Train began from the outset with the full “adult” language (which was previously unlearnable)

Training with Incremental Memory

• Phase 1:
  • Training on corpus generated from the entire grammar
  • Recurrent feedback was eliminated after every 3 or 4 words, by setting all context units to 0.5
  • Longer training phase (12 epochs, rather than 5)

• Phase 2:
  • New corpus (to avoid memorization)
  • Memory window increased to 4-5 words
  • 5 epochs

• Phase 3: 5-6 word window

• Phase 4: 6-7 word window

• Phase 5: no explicit memory limitation implemented

• Performance: as good as on the previous simulation
Analysing the solution

- Hidden units permit the network to derive a *functionally-based* representation, in contrast to a *form-based* representation of inputs.

- Various dimensions of the internal representation were used for:
  - Individual words, category, number, grammatical role, level of embedding, and verb argument type
  - The high-dimensionality of the hidden unit vectors (70 in this simulation) makes direct inspection difficult

- Solution: Principle Component Analysis can be used to identify which dimensions of the internal state represent these different factors.
  - This allows us to visualise the movement of the network through a state space for a particular factor, by discovering which units are relevant.

Principle Component Analysis

- Suppose we’re interested in analysing a network with 3 hidden units and 4 patterns of activation, corresponding to: boy_{subj}, girl_{subj}, boy_{obj}, girl_{obj}
- Cluster analysis might reveal the following structure:
  - But nothing of the subj/obj representation is revealed.
- If we look at the entire space, however, we can get more information about the representations:

  ![Diagram](image)

  - Since visualising more than 3 dimensions is difficult, PCA permits us to identify which “units” account for most of the variation.
    - Reveals partially “localist” representations in the “distributed” hidden units.
Examples of Principle Components: 1

- Agreement
  - Boy who boys chase chases boy
  - Boys who boys chase chase boy

- The 2nd PCA encodes agreement in the main clause

Examples of Principle Components: 2

- Transitivity
  - Boy chases boy
  - Boy sees boy
  - Boy walks

- Two principle components: 1 & 3

- PCA 1:
  - Nouns on the right
  - Verbs left

- PCA 2:
  - Intrans: low
  - Optional trans: mid
  - Transitive: high
Examples of Principle Components: 3

• Right embedding:
  • *Boy chases boy*
  • *Boy who chases boy chases boy*
  • *Boy chases boy who chases boy*
  • *Boy chases boy who chases boy who chases boy*

• PCA 11 and 1:
  • “Embedded clause are shifted to the left”
  • “RCs appear nearer the noun they modify”

PCA analysis of “Starting Small”

• We can use “Principle Component Analysis” to examine particularly important dimensions of the networks solutions more globally:
  • Sample of the points visited in the hidden unit space as the network processes 1000 random sentences

• The results of PCA after training:

  Training on the full data set

  Incremental training

The right plot reveals are more clearly “organised” use of the state space
Comments

• To solve the task, the network must learn the sources of variance (number, category, verb-type, and embedding)

• If the network is presented with the complete corpus from the start:
  • The complex interaction of these factors, long-distance dependencies, makes discovering the sources of variance difficult
  • The resulting solution is imperfect, and internal representation don’t reflect the true sources of variance

• When incremental learning takes place (in either form):
  • The network begins with exposure to only some of the data
    • Limited environment: simple sentences only
    • Limited mechanisms: simple sentences + noise (hence longer training)
  • Only the first 3 sources of variance, and no long-distance dependencies

• Subsequent learning is constrained (or guided) by the early learning of, and commitment to, these basic grammatical factors
  • Thus initial memory limitations permit the network to focus on learning the subset of facts which lay the foundation for future success

The importance of starting small

• Networks rely on the representativeness of the training set:
  • Small samples may not provide sufficient evidence for generalisation
    • Possibly poor estimates of the populations statistics
    • Some generalisations may be possible from a small sample, but are later ruled out
  • Early in training the sample is necessarily small

• The representation of experience:
  • Exemplar-based learning models store all prior experience, and such early data can then be re-accessed to subsequently help form new hypotheses
  • SRNs do not do this: each input has it’s relatively minor effect on changing the weights (towards a solution), and then disappears. Persistence is only in the change made to the network.

• Constraints on new hypotheses, and continuity of search:
  • Changes in a symbolic systems may lead to suddenly different solutions
    • This is often ok, if it can be checked against the prior experience
  • Gradient descent learning makes it difficult for a network to make dramatic changes in its solution: search is continuous, along the error surface
  • Once committed to an erroneous generalisation, the network might not escape from a local minima
Starting small (continued)

• Network are most sensitive during the early period of learning:
  • Non-linearity (the logistic activation function) means that weight modifications are less likely as learning progresses
    • Input is "squashed" to a value between 0 and 1
    • Non-linearity means that the function is most sensitive for inputs around 0 (output is 0.5)
    • Nodes are typically initialised randomly about 0, so netinput is also near 0
    • Thus the network is highly sensitive
  • Sigmoid function become “saturated” for large +/- inputs
    • As learning proceeds units accrue activation
    • Weight change is a function of the error and slope of the activation function
    • This will become smaller as units activations become saturation, regardless of how large the error is
  • Thus escaping from local minima becomes increasingly difficult

• Thus most learning occurs when information is least reliable

Conclusions

• Learning language is difficult because:
  • Learning linguistic primitives is obscured by the full complexity of grammatical structure
  • Learning complex structure is difficult because the network lacks knowledge of the basic primitive representations

• Incremental learning shows how a system can learn a complex system by having better initial data:
  • Initially impoverished memory provides a natural filter for complex structures early in learning so the network can learn the basic forms of linguistic regularities
  • As the memory is expanded, the network can use what it knows to handle increasingly complex inputs
  • Noise, present in the early data, tends to keep the network in a state of flux, helping it to avoid committing to false generalisations
Rohde & Plaut (1999)

- Model: Predict next word of a sentence
  - Simulation 1: Significant advantage for starting with the full language; even more so if languages were made more natural by increasing the number of clauses obeying semantic constraints
  - Simulation 2: Failure to replicate starting small advantage even with Elman’s parameters and initial weights; instead advantage for full language
  - Simulation 3: Limited memory failed to provide an advantage over full memory even with increased training time—however, limited memory was generally less of a hindrance than simplified input
- Limitation: Syntactic prediction is not comprehension!
- Conclusion: Simulations call into the question the proposal that limited cognitive resources are necessary, or even beneficial for language acquisition.

Summary of SRNs …

- Finding structure in time/sequences:
  - Learns dependencies spanning more than a single transition
  - Learns dependencies of variable length
  - Learns to make partial predictions from structure input
    - Prediction of consonants, or particular lexical classes
- Learning from various input encodings:
  - Localist encoding: XOR and 1 bit per word
  - Distributed:
    - Structured: letter sequences where consonants have a distinguished feature
    - Random: words mapped to random 5 bit sequence
- Learns both general categories (types) and specific behaviours (tokens) based purely on distributional evidence in the linguistic signal
- Able to learn complex syntactic constraints, such as agreement, subcategorisation, and embeddings
- What are the limitations of SRNs
  - Do they simply learn co-occurrences and contingent probabilities?
  - Can they learn more complex aspects of linguistic structure?
  - Are they as successful for comprehension, as they are for prediction?