Computational Psycholinguistics

Lecture 12: Simple Recurrent Networks

Marshall R. Mayberry

Computerlinguistik
Universität des Saarlandes

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

Overview

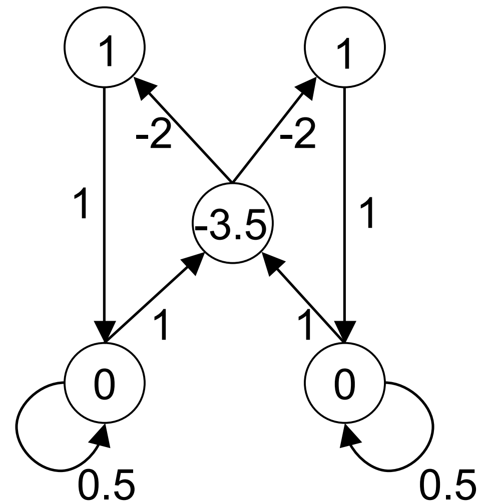
- Two models: Single (connectionist) mechanisms account for dual-route models of “rule-like” and “exceptional” behaviour:
  - Reading Aloud: Models of adult performance
    - Good performance on known and unknown words
    - Models (normal) human behaviour (frequency x regularity, etc)
  - English past-tense: Models Acquisition of Verb Morphology
    - Forming the past tense from the present
  - Problems: dual-route models better explain double dissociations
  - “Static”: 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

Representing Time

- Many cognitive functions involve processing sequences of inputs/outputs over time:
  - Sequences of motor movements
  - Sequences of sounds to produce a particular word
  - Sequences of words encountered incrementally
- We can directly represent time as “order” in the input pattern vector
  - Assumes buffering of events before processing, and processing takes place all at once (i.e. in parallel)
  - Maximum sequence length (duration) is fixed
  - Does not easily distinguish relative versus absolute temporal position, e.g.
    - 0 1 1 1 0 0 0 0 0
    - 0 0 0 1 1 1 0 0 0
    - Similar patterns are spatially distant (and learning such translational variance requires an external teacher)
- We need a richer, more general representation of time

Recurrent networks

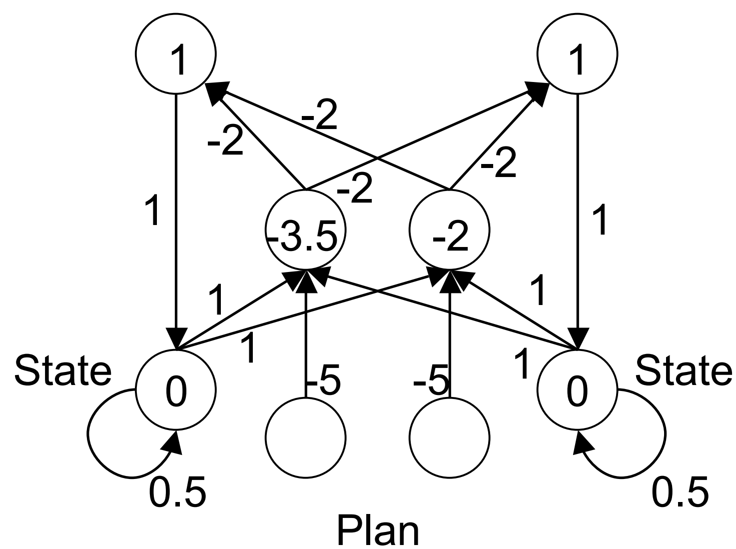
- Suppose we want a network to generate a sequence of outputs:
  - E.g.: AAAB
- Consider the following network:
  - Inputs are linear, rest are binary threshold units:
    - Positive = 1
    - Negative = 0
  - Let A = 1 1; B = 0 0
  - The neg. bias of the hidden node keeps activity from being propagated during first cycles



Time	Input 1		Input 2		Hidden		Output 1		Output 2		Resp
	In	Out	In	Out	In	Out	In	Out	In	Out	
1	0+0	0	0+0	0	0-3.5	0	0+1	1	0+1	1	A
2	1+0	1	1+0	1	2-3.5	0	0+1	1	0+1	1	A
3	1+.5	1.5	1+.5	1.5	3-3.5	0	0+1	1	0+1	1	A
4	1+.75	1.75	1+.75	1.75	3.5-3.5	1	-2+1	0	-2+1	0	B

Recurrent networks with state units

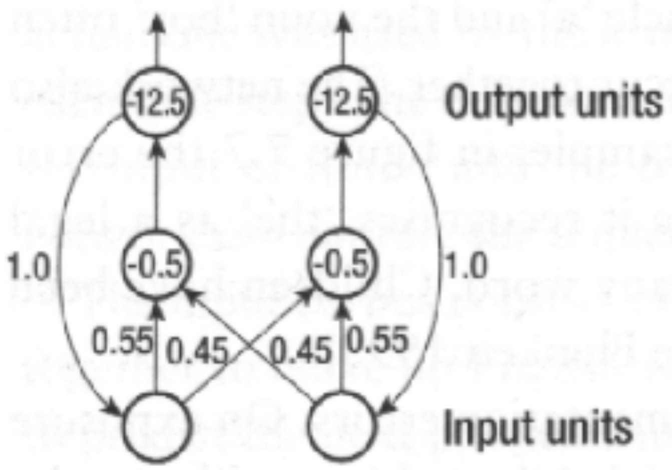
- We can add inputs to the recurrent network which modulate the effect of the state units:
  - These inputs are called "plan" units

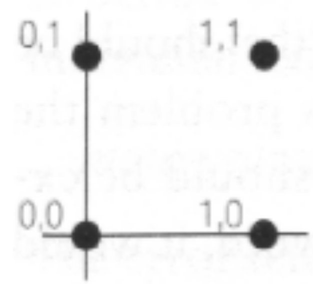


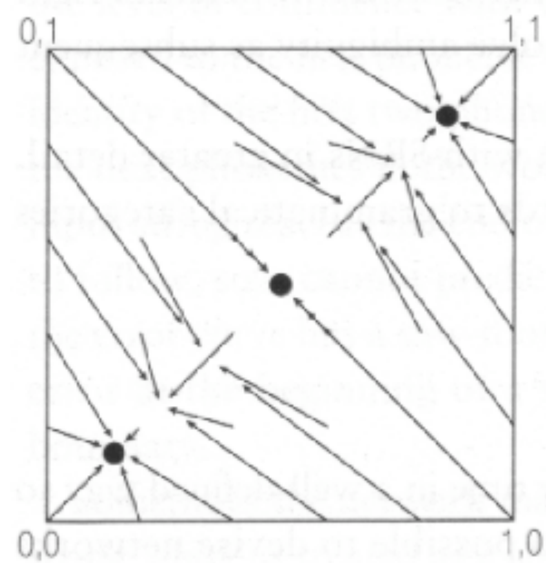
- In this way inputting (0 1) results in AAAB, while inputting (1 0) results in AB

Attractors

- Some recurrent networks change over time such that the output settles into a particular state: Attractor networks
  - The set of possible states are the attractors
- Ability to model reaction times, robust to noisy input
- Can perform an arbitrary mapping from input to output

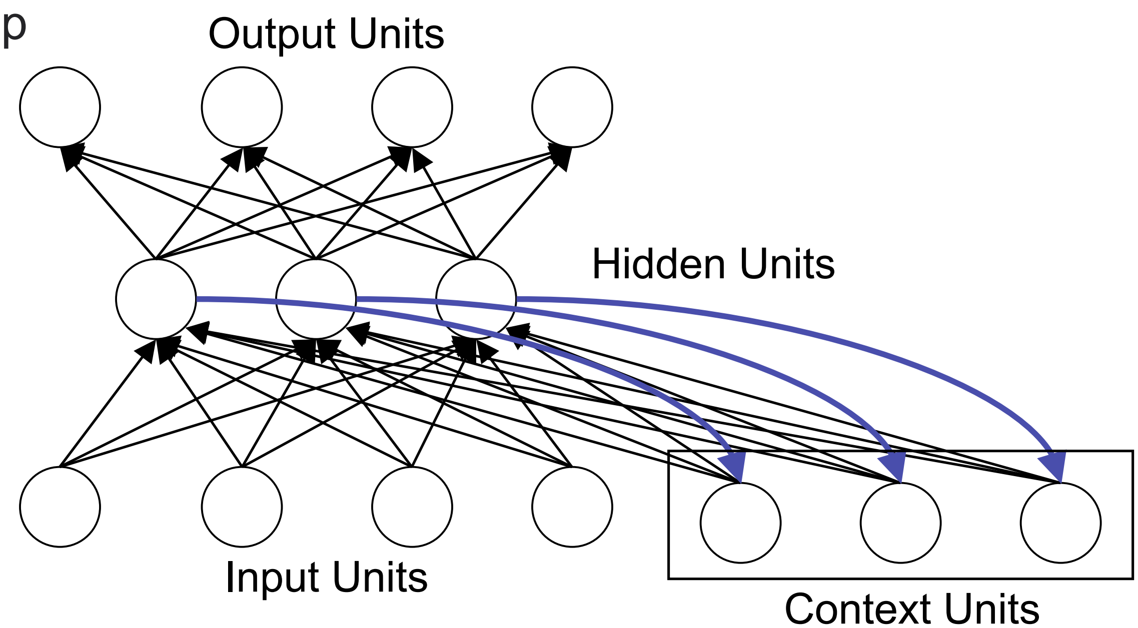






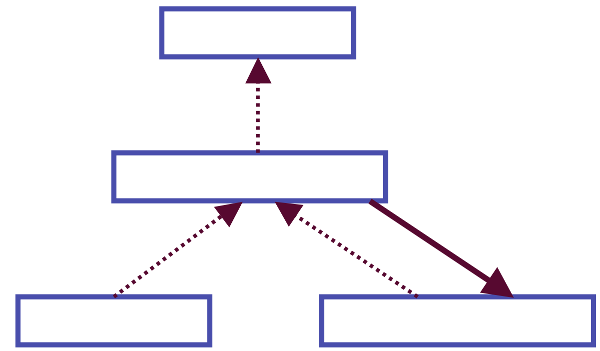
Simple Recurrent Networks

- Recurrent networks are powerful for executing and learning complex sequences, but difficult to design
- Simple recurrent networks can learn any sequence given as input
- We can tell they've learned by training them to predict the next item
- Hidden units are connected to "context" units:
  - These correspond to "state" units: they remember the state of the network on the previous time step
  - The hidden units are able to recycle information over multiple time steps
  - Dynamic memory: Identical inputs can be treated differently depending on context



SRNs

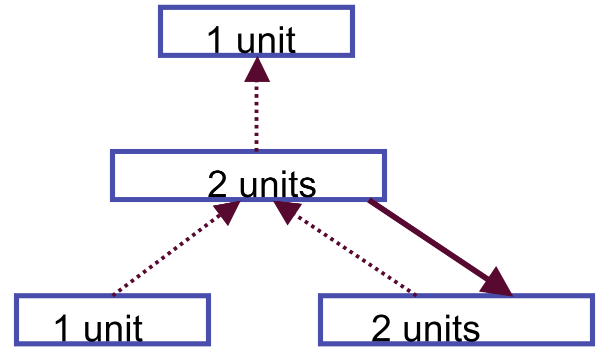
- Context units are <u>direct copies</u> of hidden units, the connections are not modifiable
  - Connections are one-to-one
  - Weights are fixed at 1.0
- Connections from context units to hidden units are modifiable; weights are learned just like all other connections
  - Training is done via the backpropagation learning algorithm



- Solution: let time be represented by its affect on processing
  - Dynamic properties which are responsive to temporal sequences
  - Memory
- Dynamical systems: "any system whose behaviour at one point in time depends in some way on its state at an earlier point in time"
  - See: *Rethinking Innateness*, Chapter 4.

Temporal XOR

- We know that XOR cannot be learned by a simple 1-layer network
- We can translate it into a “temporal” task by presenting input/output sequences:
  - Input: 1 0 1 0 0 0 0 1 1 1 1 0 1 0 1 …
  - Output: 0 1 0 0 0 0 1 1 1 1 0 1 0 1 ? …



- Training:
  - Construct a sequence of 3000 bits
  - 600 passes
  - Predict the next bit in the sequence
  - Prediction is based on both the current input **and** the network’s previous state

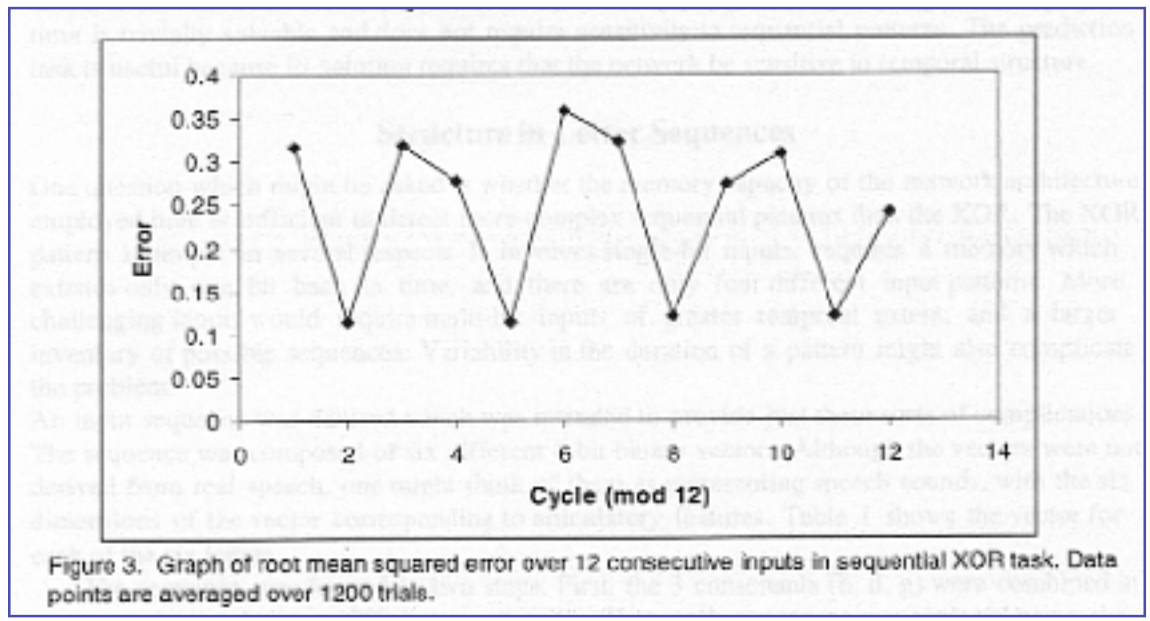


Figure 3. Graph of root mean squared error over 12 consecutive inputs in sequential XOR task. Data points are averaged over 1200 trials.

Observations of XOR

- The network successfully predicts every third bit:
  - Correct, since other bits are random
  - Note: actually attempts to apply the XOR rule for each input bit
- The network's solution:
  - At the hidden layer, 1 unit is active when the input contains a sequence of identical elements
  - The other unit is active when input elements alternate
  - Thus the network has become sensitive to high/low "frequency"
  - This is different from the static solution to the problem
- Note: the prediction task is analogous to autoassociation
  - Instead of exploiting redundancy in patterns, it must discover the temporal structure of the input

"Finding Structure in Time"

Structure in Letter Sequences

- A simple feedforward network can be trained to learn simple transitions between two adjacent inputs
- For XOR, the SRN has demonstrated the ability to learn dependencies spanning 3 adjacent inputs
  - Single bit inputs
  - Only 4 different patterns
- Is the memory capacity of SRN sufficient to detect more complex sequential patterns?
  - Multi-bit inputs
  - Greater temporal extent
  - Larger inventory of sequences
- Imagine a simplified system of speech sounds
  - 3 consonants
  - 3 vowels
  - Each consonant is followed by a fixed number of a particular vowel

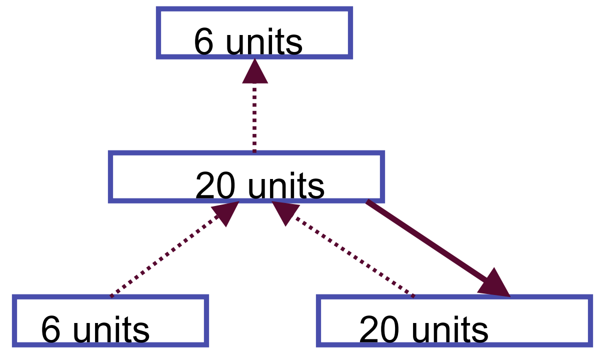
Performance

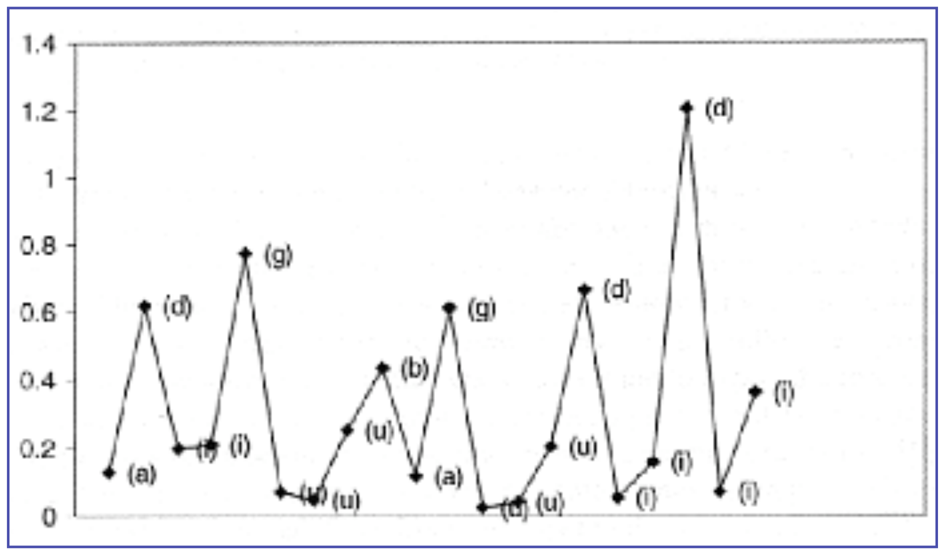
- Rules for "word" formation:
  - b → ba
  - d → dii
  - g → guuu
- The 3 consonants were randomly combined to generate a 1000 letter sequence
- The consonants were then replaced using the above rules
  - dbgbdd… → diibaguuubadiidii…
  - Each letter was then converted to a 6 bit distributed representation:

	Consonant	Vowel	Interrupted	High	Back	Voiced
b	1	0	1	0	0	1
d	1	0	1	1	0	1
g	1	0	1	0	1	1
a	0	1	0	0	1	1
i	0	1	0	1	0	1
u	0	1	0	1	1	1

Training & Performance

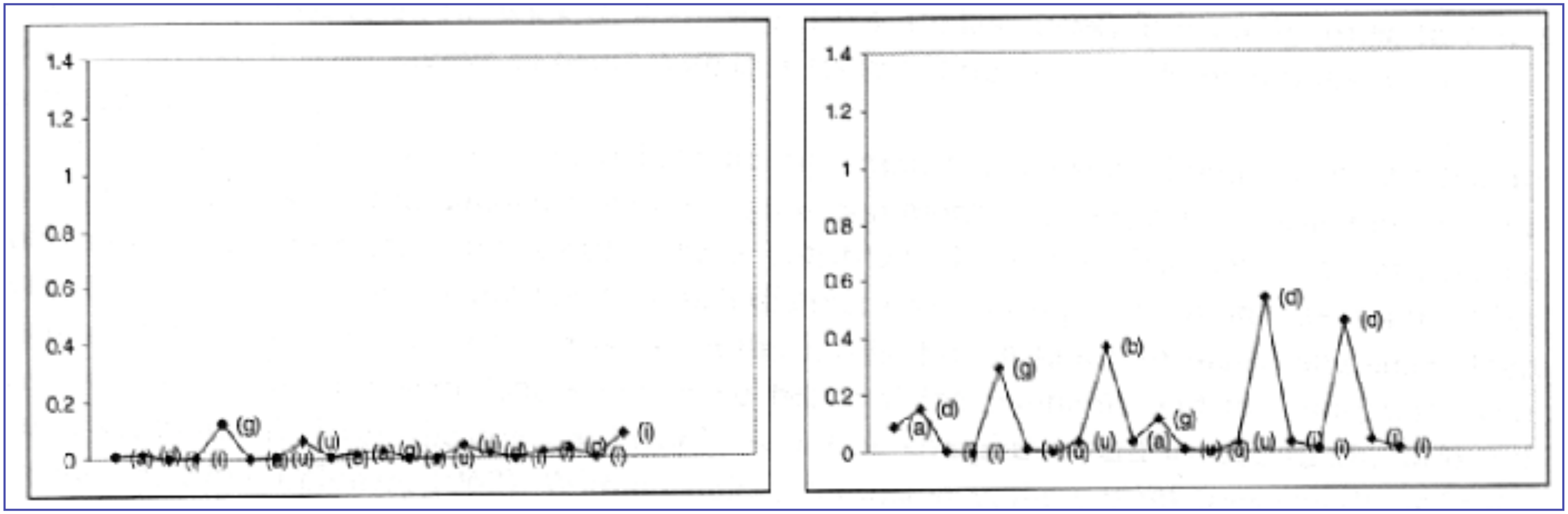
- The network architecture has 6 input and output units, with 20 hidden and context units
- Training:
  - Each input vector is presented
  - Trained to predict the next input
  - 200 passes through the sequence
- Tested on another random sequence (using same rules)
- Error for part of the test is shown 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 we predict which vowel follows a consonant, and how many (?)
- We can examine the error for the individual bits, e.g. [1] and [4]:



- 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

Remarks

- The network identifies patterns of longer duration than XOR
- The pattern length is variable
- Inputs are complex: 6 bit distributed representations

- *Subregularities* in the vector representations enable the network to make partial predictions even where complete prediction is not possible
  - Depends, of course, on structuring of the input data

- Possible conclusions:
  - Learning extended sequential dependencies is possible
  - If dependencies are appropriately structured, this may facilitate learning

Discovering word boundaries

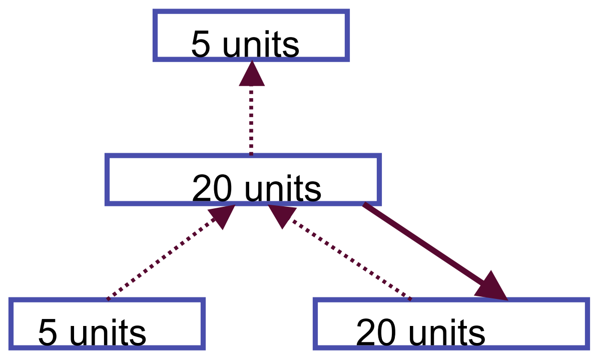
- We often take for granted the existence of words, and yet for the child language learner, input is largely in the form of an unsegmented acoustic stream.
- How do children learn to identify word boundaries in such a signal?

- Example: Predicting the next sound
  - Problem: discovering word boundaries in continuous speech
    - Approximated by a corpus of continuous phonemes
  - Task: network is presented with one phoneme and attempts to predict the next one
  - *Manyyearsagoaboyandgirllivedbytheseatheyplayedhappily*

- At time *t*: the network knows both the current input (phoneme at time *t*) and the results of processing at time *t-1* (context units)
  Problem: discovering word boundaries in continuous speech

The network and training

- We approximate the acoustic input with an orthographic representation:
  - Lexicon of 15 words and a sentence generating program generated 200 sentences of length 4 to 9 words
  - Concatenated to produce a stream of 1270 words, or 4963 letters
  - Each letter converted to a random (not structured) 5 bit vector
- Architecture:

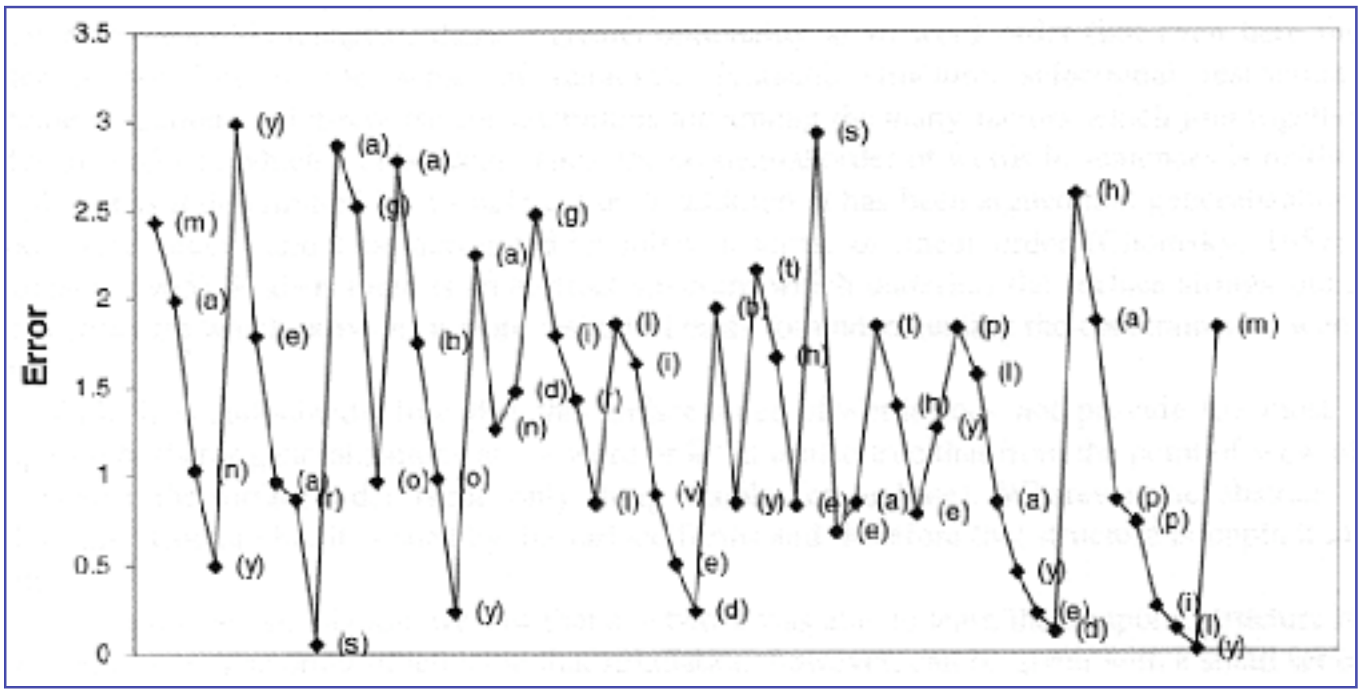


- Training:
  - 10 complete passes through the sequence

Input		Output	
0110	*m*	0000	*a*
0000	*a*	0111	*n*
0111	*n*	1100	*y*
1100	*y*	1100	*y*
1100	*y*	0010	*e*
0010	*e*	0000	*a*
0000	*a*	1001	*r*
1001	*r*	1001	*s*
1001	*s*	0000	*a*
0000	*a*	0011	*g*
0011	*g*	0111	*o*
0111	*o*	0000	*a*
0000	*a*	0001	*b*
0001	*b*	0111	*o*
0111	*o*	1100	*y*
1100	*y*	0000	*a*
0000	*a*	0111	*n*
0111	*n*	0010	*d*
0010	*d*	0011	*g*
0011	*g*	0100	*i*
0100	*i*	1001	*r*
1001	*r*	0110	*l*
0110	*l*	1100	*y*
1100	*y*		

Predicting the next sound

- We can examine the error:
  - 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



Remarks

- Network learns statistics of co-occurrences, which are graded
  - Criteria for boundaries is relative
  - E.g. see the ambiguity of “y”
  - Could misidentify common co-occurrences as individual words
    - Some evidence of this in early child language acquisition: idioms = words

- This simulation is <u>not</u> proposed as a model of word acquisition
  - While listeners are often able to make “predictions” from partial input, it is not the major goal of language learning
  - Sound co-occurrences are only part of what identifies “words”
  - This simulation considers only one aspect of available information

- The simulation demonstrates that there is information in the input signal which serves as a cue to word boundaries
- The simulation demonstrates the sensitivity of SRNs to this information

Discovering lexical classes from word order

- Surface word order is influenced by numerous factors
  - Syntax, selectional and subcategorization restrictions, discourse factors …
  - Symbolic treatments appeal to relatively abstract, interacting rules which often depend on rich, hierarchical representations
    - Often, these accounts assume innately specified constraints
  - Discovering information from word order might therefore be beyond the capacity of the demonstrated sequential learning abilities of SRNs

- Maxim of empirical linguistics (Firth): “You shall know a word by the company it keeps”
  - verbs typically follow auxilliaries, and precede determiners
  - nouns are often preceded by determiners
  - Also, selectional information: verbs are followed by specific kinds of nouns

- First simulation: a sentence generator produced a set of simple (2 and 3 word) sentences using 29 lexical items from 13 “classes”

Structure of Training Environment

- Categories of lexical items

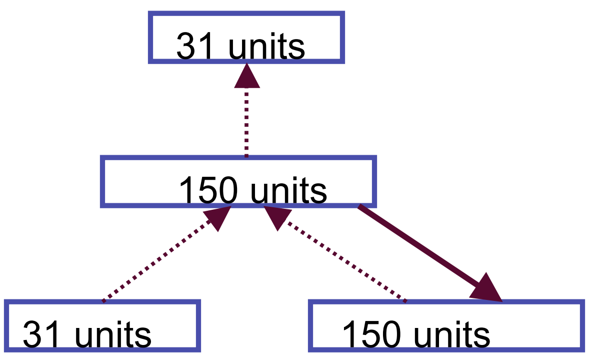
Category	Examples
NOUN-HUM	man,woman
NOUN-ANIM	cat,mouse
NOUN-INANIM	book,rock
NOUN-AGRESS	dragon,monster
NOUN-FRAG	glass,plate
NOUN-FOOD	cookie,sandwich
VERB-INTRAN	think,sleep
VERB-TRAN	see,chase
VERB-AGPAT	move,break
VERB-PERCEPT	smell,see
VERB-DESTROY	break,smash
VERB-EAT	eat

- Template for sentence generator

WORD 1	WORD 2	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-ANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTROY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

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

INPUT		OUTPUT	
0000000000000000000000000000010	(woman)	0000000000000000000000000010000	(smash)
0000000000000000000000000010000	(smash)	0000000000000000000001000000000	(plate)
0000000000000000000001000000000	(plate)	0000010000000000000000000000000	(cat)
0000010000000000000000000000000	(cat)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000100000000000000	(man)
0000000000000000100000000000000	(man)	0001000000000000000000000000000	(break)
0001000000000000000000000000000	(break)	0000100000000000000000000000000	(car)
0000100000000000000000000000000	(car)	0100000000000000000000000000000	(boy)
0100000000000000000000000000000	(boy)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000001000000000000000000	(girl)
0000000000001000000000000000000	(girl)	0000000000100000000000000000000	(eat)
0000000000100000000000000000000	(eat)	0010000000000000000000000000000	(bread)
0010000000000000000000000000000	(bread)	0000000010000000000000000000000	(dog)
0000000010000000000000000000000	(dog)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000001000000000000	(mouse)
0000000000000000001000000000000	(mouse)	0000000000000000000100000000000	(move)
0000000000000000000100000000000	(move)	1000000000000000000000000000000	(book)
1000000000000000000000000000000	(book)	0000000000000001000000000000000	(lion

Performance

- Training yields an RMS error of 0.88
- RMS error rapid drops from 15.5 to 1, by simply learning to turn all outputs off (due to sparse, localist representations)

- Prediction is non-deterministic: next input cannot be predicted with absolute certainty, but neither is it random
  - Word order and selectional restrictions <u>partially</u> constrain what words are likely to appear next, and which cannot appear.
  - We would expect the network to learn the <u>frequency</u> 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_{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*
    - …
  - We then 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.

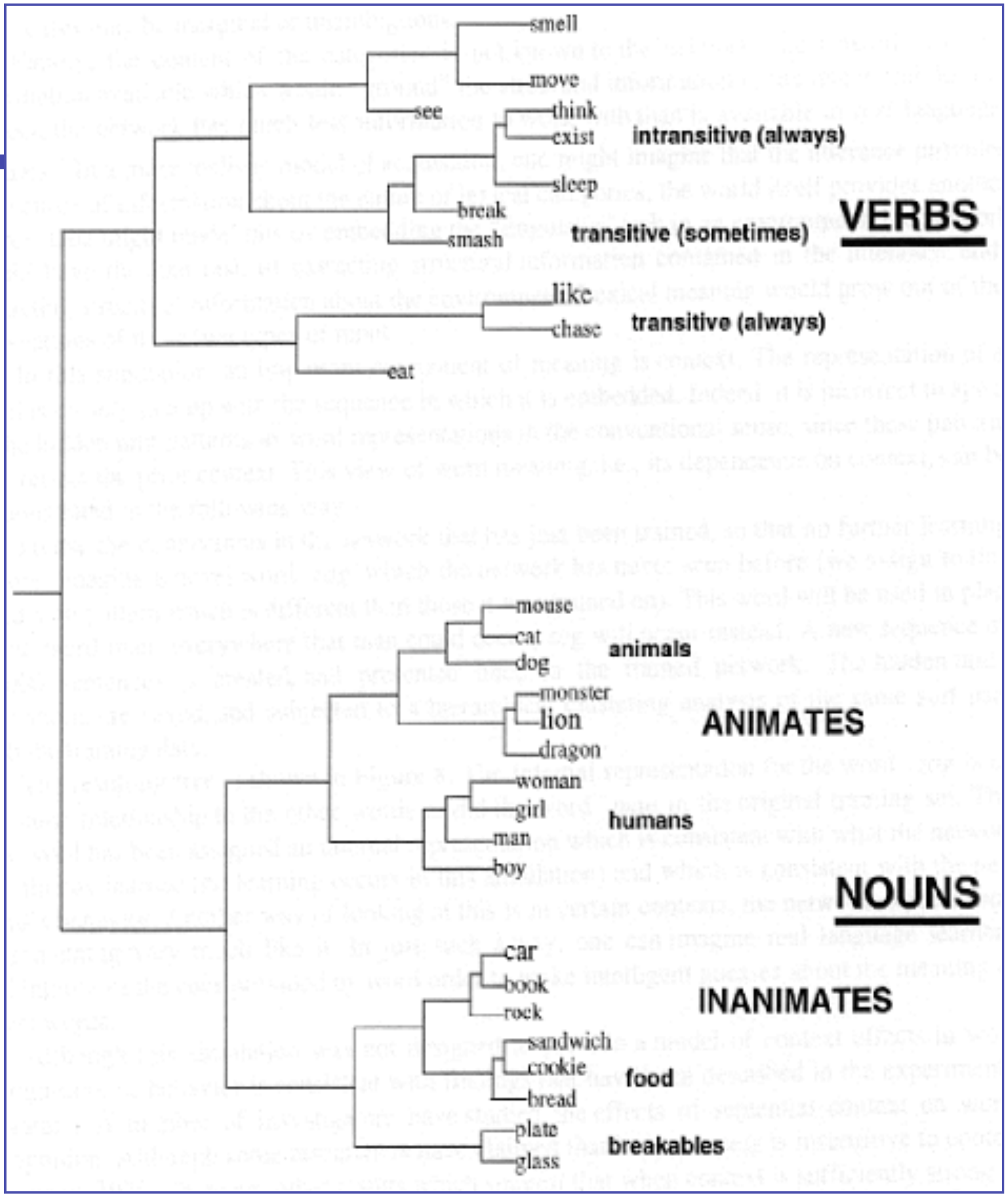
Remarks on performance

- 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 <u>word in its context</u>
- 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"

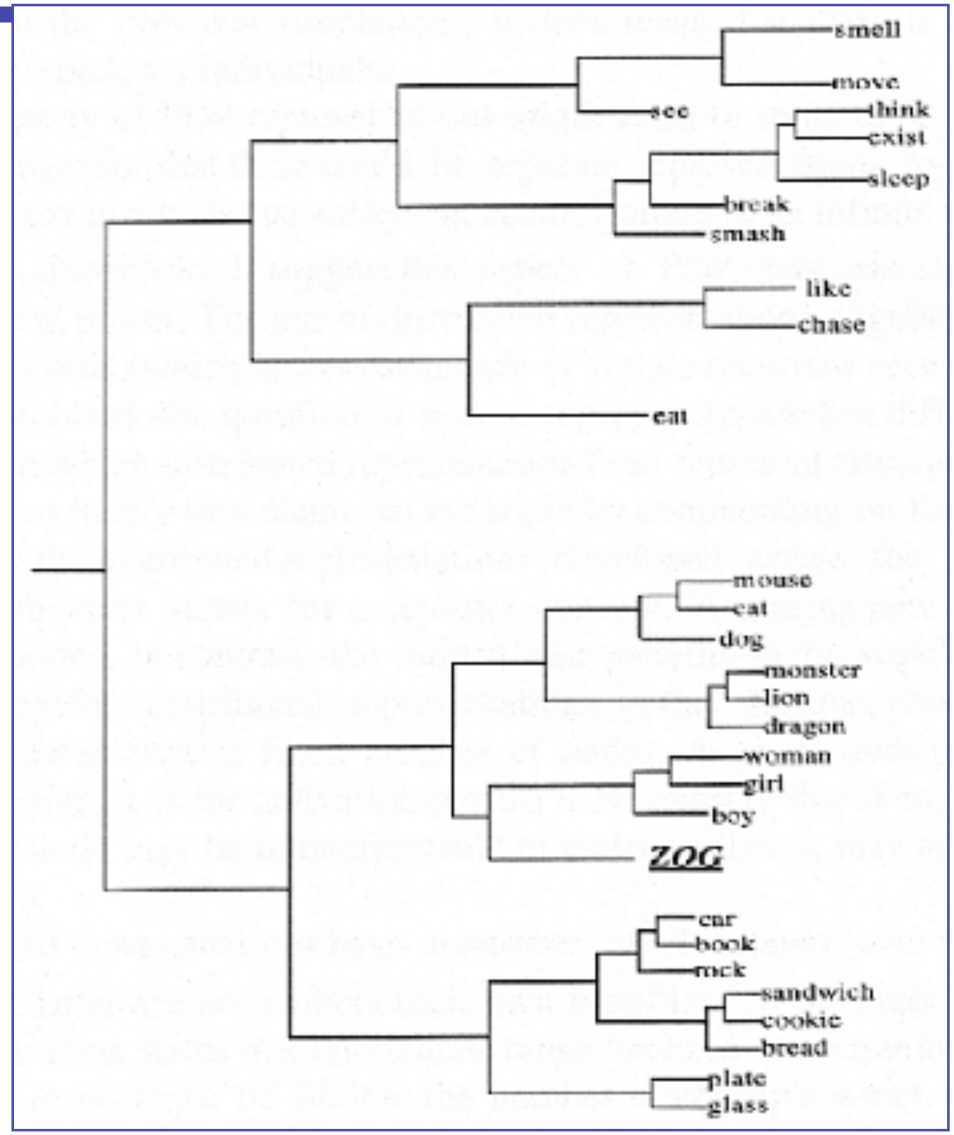


General Remarks

- Representations near one another form classes
- Higher level categories correspond to larger, more general regions
  - Categories are hierarchical
- The hierarchical categorisation is “soft”
  - Some categories are categorially distinct
  - Others share properties and have less distinct boundaries
  - Category membership can be marginal or unambiguous
- Cannot assign a given input to multiple positions
  - I.e. cannot learn to distinguish multiple word “senses”
- Categories have no “content”: they are not grounded in the real world
  - While learners do have, e.g. correlated visual input
- An important component of the word’s meaning is its context
  - Hidden units reflect both the word and its prior context
  - Words take much of their meaning from the context they appear in
  - We should therefore be able to assign meaning to unknown words …

Unknown words

- If we replace "man" with a novel word "zog"
  - "Zog" is represented by a new input vector
  - We can now present the new testing corpus to the frozen network
  - Re-perform the hierarchical cluster analysis …
- "Zog" bears the same relationship to other words as "man" did in the original training set
- The new word's internal rep'n is based on its behaviour



General discussion

- The network learns hierarchical categories and classes
  - Such classes are determined from word order/co-occurrence
  - Learning takes place purely on the basis of observable data
    - No pre-specified localist representations, etc.

- Predicts “context” effects in processing:
  - Consistent with findings that human lexical access is sensitive to context
    - Controversial: there is evidence both for (Tabossi) and against (Swinney) immediate context effects in lexical access
  - And that it is word classes that are predicted, not individual words