# Neural networks for natural language

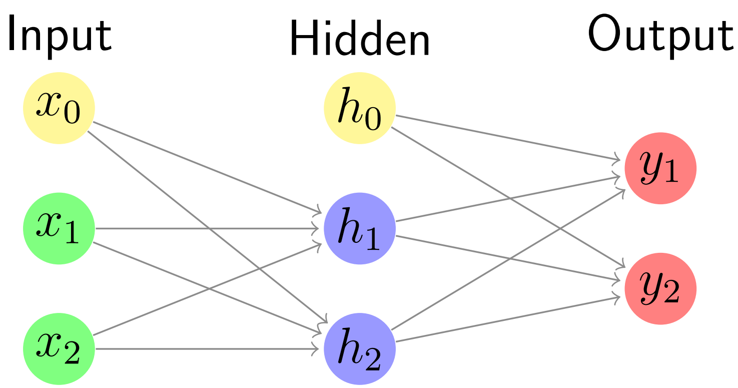
Roger Levy

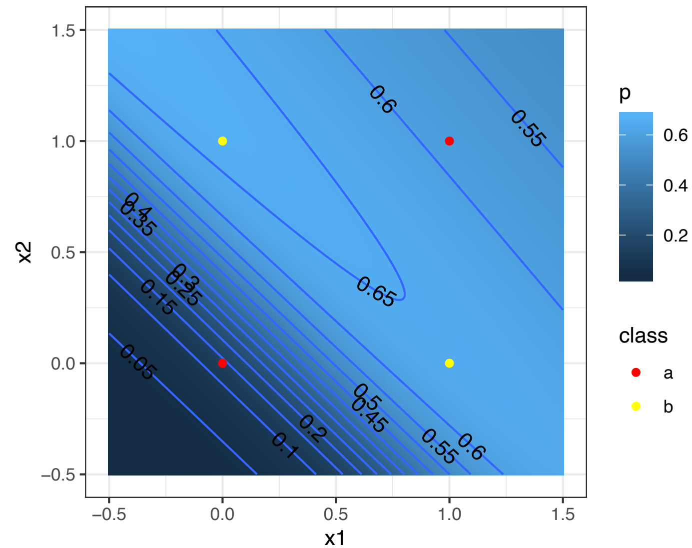
9.19: Computational Psycholinguistics

2 November 2023

# Agenda for the day

- Last time: with a hidden layer, a NN can learn XOR...




- ...but language isn't just 2D input+2-class output! So, **today:**
- Dealing with language in neural networks
- Recurrent neural networks (RNNs)
  - Simple recurrent networks (SRNs)
  - Gated recurrent units (GRUs)
  - Long short-term memory networks (LSTMs)
- Examining RNN behavior

# Dealing with language inputs

For language, input $\{x_i\}$ and output prediction $y$ seem discrete:

**`Adam adores zebras ...`**

Simplest approach is ***localist*** or ***one-hot*** representations:

$$\texttt{Adam} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \texttt{adores} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad \texttt{zebras} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

But lower-dimensional ***embeddings*** capture word similarities:

$$\texttt{Adam} \rightarrow \begin{bmatrix} 0.6 \\ 0.3 \end{bmatrix} \quad \texttt{adores} \rightarrow \begin{bmatrix} -0.3 \\ 0.4 \end{bmatrix} \quad \texttt{zebras} \rightarrow \begin{bmatrix} 0.7 \\ -0.1 \end{bmatrix}$$

# Example feed-forward+embedding LM

Bengio et al., 2003: Neural *n*-gram language model

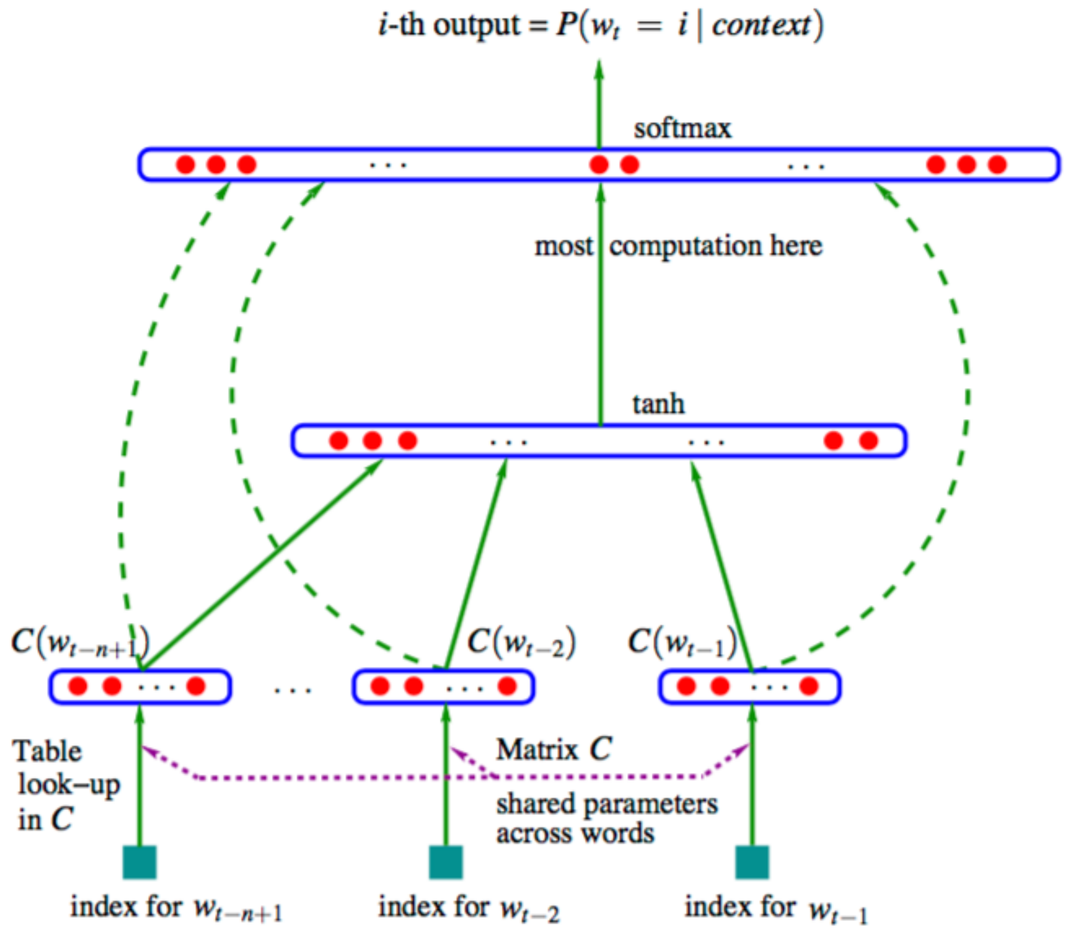



Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where $g$ is the neural network and $C(i)$ is the $i$-th word feature vector.

# Old (2003!) perplexity results on Brown corpus

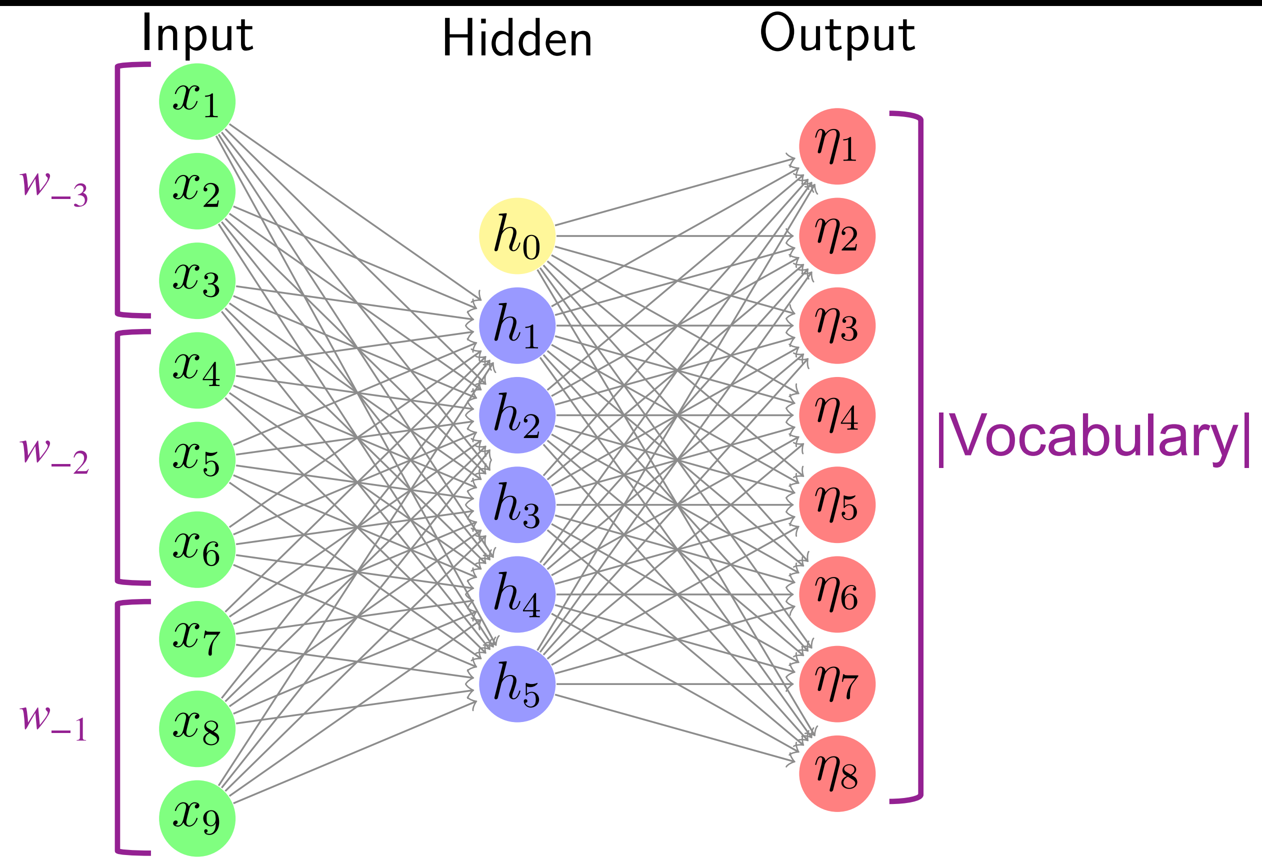
*neural language models*

*n-gram language models*

| | n | c | h | m | direct | mix | train. | valid. | test. |
|---|---|---|---|---|---|---|---|---|---|
| MLP1 | 5 | | 50 | 60 | yes | no | 182 | 284 | 268 |
| MLP2 | 5 | | 50 | 60 | yes | yes | | 275 | 257 |
| MLP3 | 5 | | 0 | 60 | yes | no | 201 | 327 | 310 |
| MLP4 | 5 | | 0 | 60 | yes | yes | | 286 | 272 |
| MLP5 | 5 | | 50 | 30 | yes | no | 209 | 296 | 279 |
| MLP6 | 5 | | 50 | 30 | yes | yes | | 273 | 259 |
| MLP7 | 3 | | 50 | 30 | yes | no | 210 | 309 | 293 |
| MLP8 | 3 | | 50 | 30 | yes | yes | | 284 | 270 |
| MLP9 | 5 | | 100 | 30 | no | no | 175 | 280 | 276 |
| MLP10 | 5 | | 100 | 30 | no | yes | | 265 | **252** |
| Del. Int. | 3 | | | | | | 31 | 352 | 336 |
| Kneser-Ney back-off | 3 | | | | | | | 334 | 323 |
| Kneser-Ney back-off | 4 | | | | | | | 332 | 321 |
| Kneser-Ney back-off | 5 | | | | | | | 332 | 321 |
| class-based back-off | 3 | 150 | | | | | | 348 | 334 |
| class-based back-off | 3 | 200 | | | | | | 354 | 340 |
| class-based back-off | 3 | 500 | | | | | | 326 | **312** |
| class-based back-off | 3 | 1000 | | | | | | 335 | 319 |
| class-based back-off | 3 | 2000 | | | | | | 343 | 326 |
| class-based back-off | 4 | 500 | | | | | | 327 | 312 |
| class-based back-off | 5 | 500 | | | | | | 327 | 312 |

*(Bengio et al., 2003)*

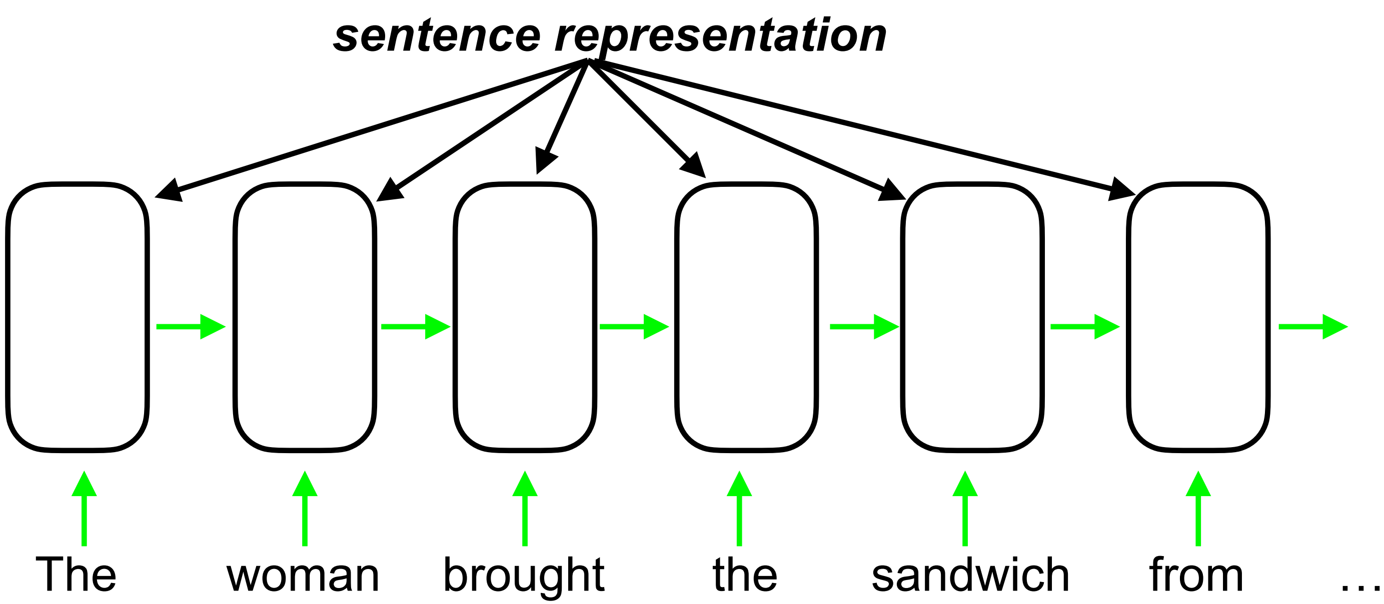
# The neural *n*-gram model




- **Advantages:** generalizes over *n*-gram contexts
- **Limitations:**
  - this is for a fixed dimensionality input context
  - how to model variable-length context, like sentences?

# Recurrent neural networks for language

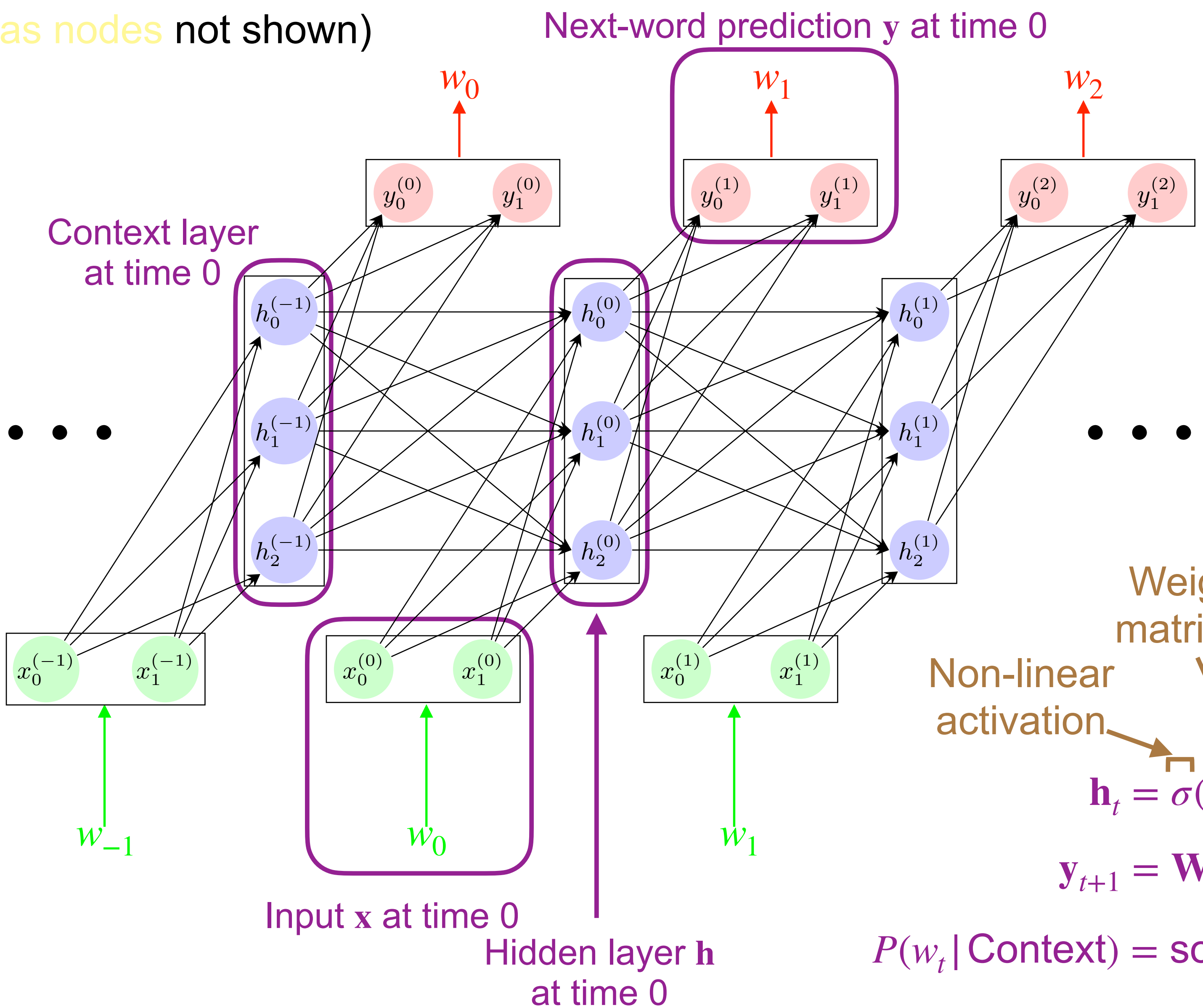
- Draw inspiration from real-time nature of human language processing
- Previous inputs must be integrated and remembered all together in a uniform representational space




*(Jordan, 1986; Elman, 1990)*

# The Simple Recurrent Network (SRN)

(bias nodes not shown)




Weight matrices

Non-linear activation

$$\mathbf{h}_t = \sigma(\mathbf{U_h}\mathbf{h}_{t-1} + \mathbf{W_h}\mathbf{x}_t)$$

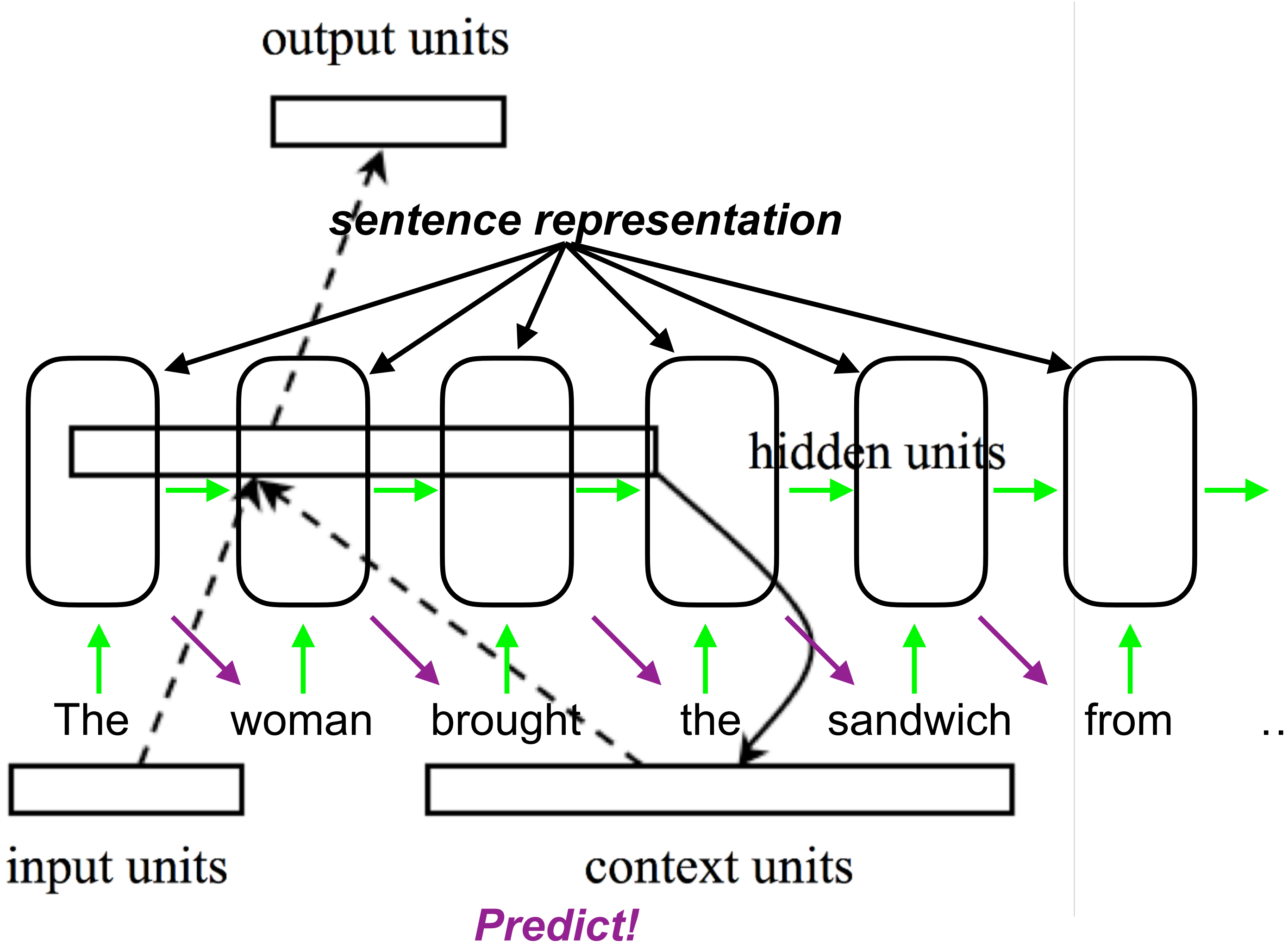
$$\mathbf{y}_{t+1} = \mathbf{W_y}\mathbf{h}_t$$

$$P(w_t \,|\, \text{Context}) = \text{softmax}(\mathbf{y}_t)$$

(Elman, 1990)

# SRN "rolled up" and unrolled

- A “rolled-up” representation (Elman, 1990); and unrolled:



# Learning with artificial language input

TABLE 3
Categories of Lexical Items Used in Sentence Simulation

| 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, break |
| VERB-INTRAN | think, sleep |
| VERB-TRAN | see, chase |
| VERB-AGPAT | move, break |
| VERB-PERCEPT | smell, see |
| VERB-DESTROY | break, smash |
| VERB-EAT | eat |

TABLE 4
Templates 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-INANIM |
| 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 |

*(Elman, 1990)*

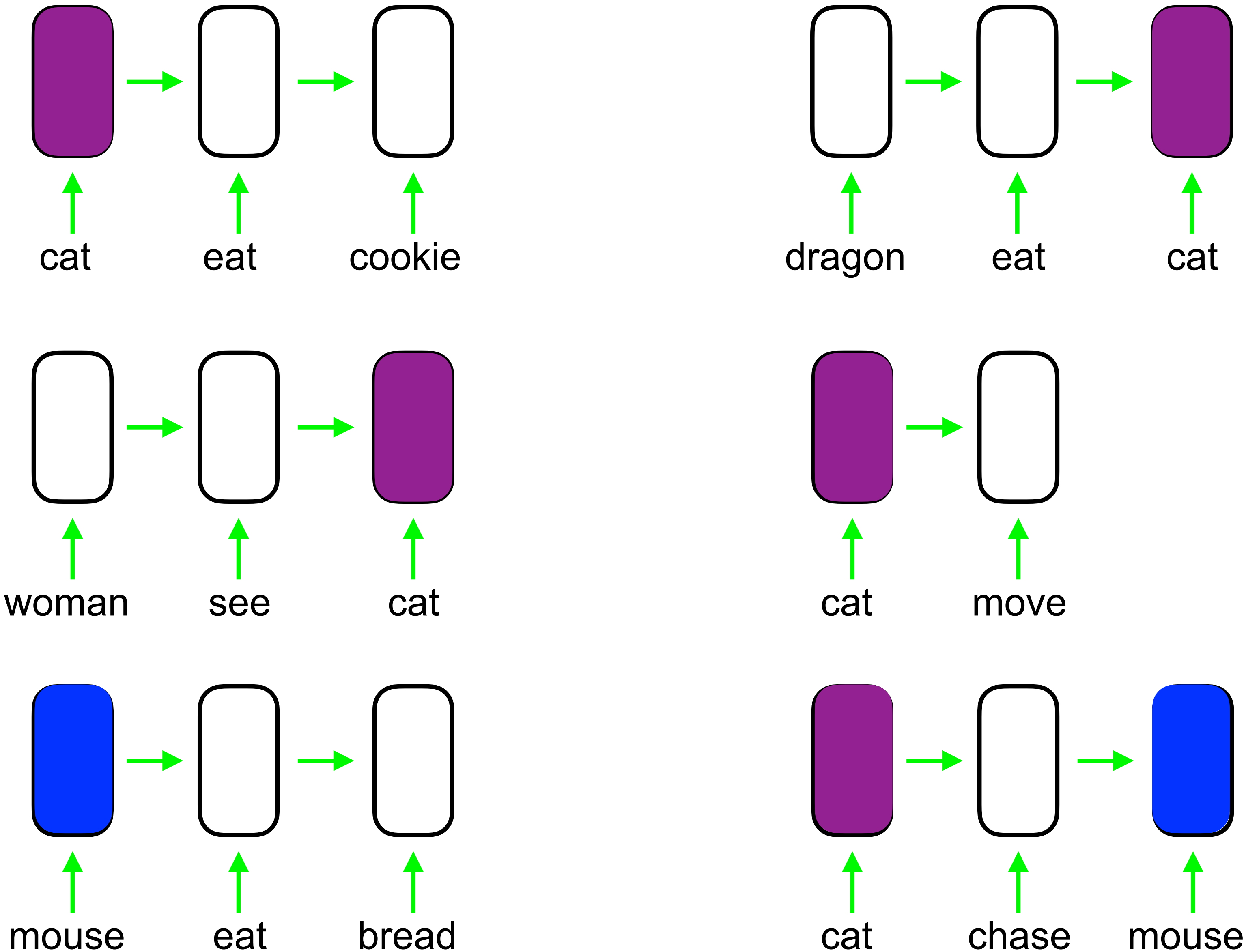
# Used *localist* word representations

**Fragment of Training Sequences for Sentence Simulation**

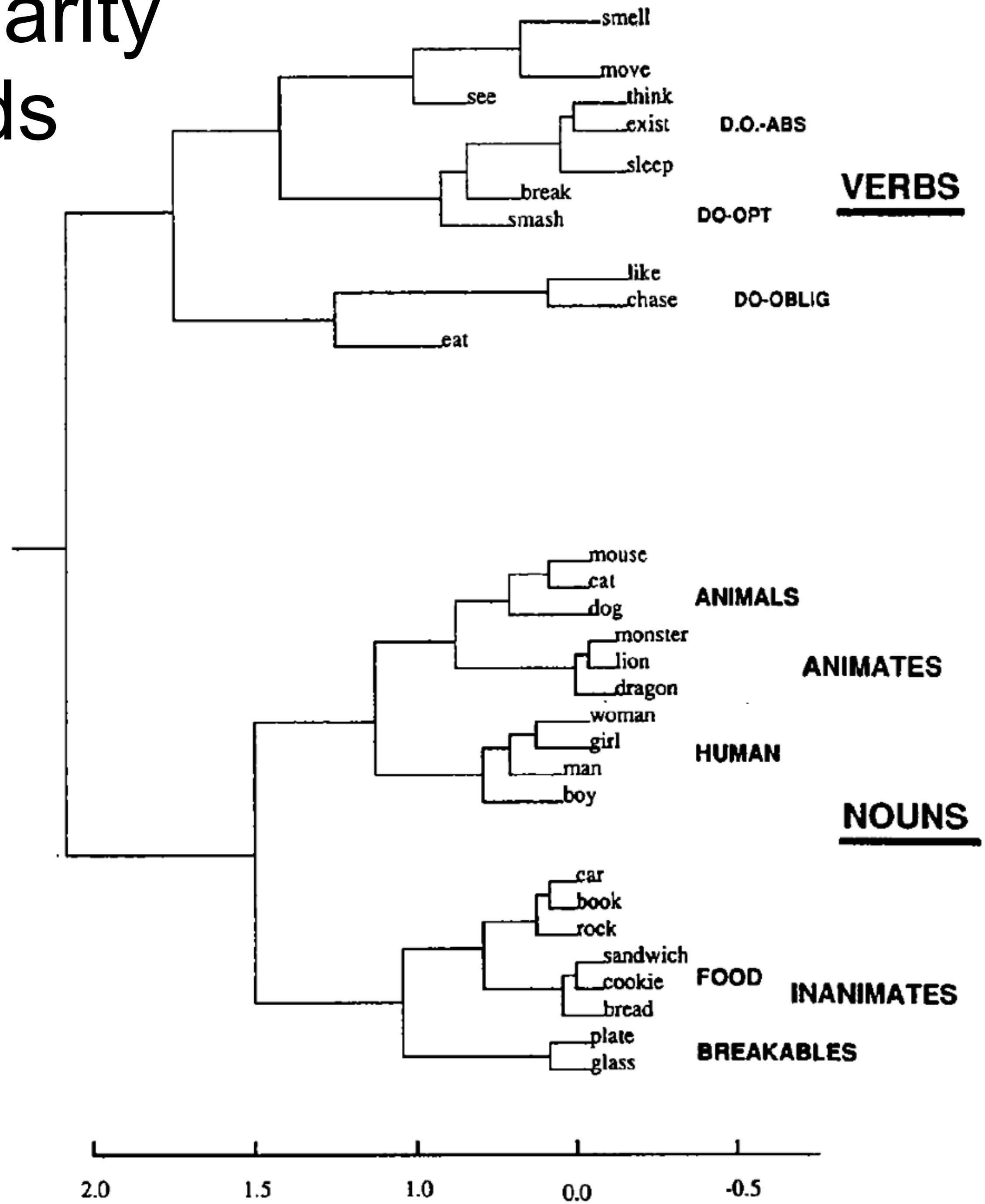
| Input | Output |
|---|---|
| 0000000000000000000000000000010 (*woman*) | 0000000000000000000000000010000 (*smash*) |
| 0000000000000000000000000010000 (*smash*) | 0000000000000000000001000000000 (*plate*) |
| 0000000000000000000001000000000 (*plate*) | 0000010000000000000000000000000 (*cat*) |
| 0000010000000000000000000000000 (*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*) |

*(Elman, 1990)*

# Learning word classes



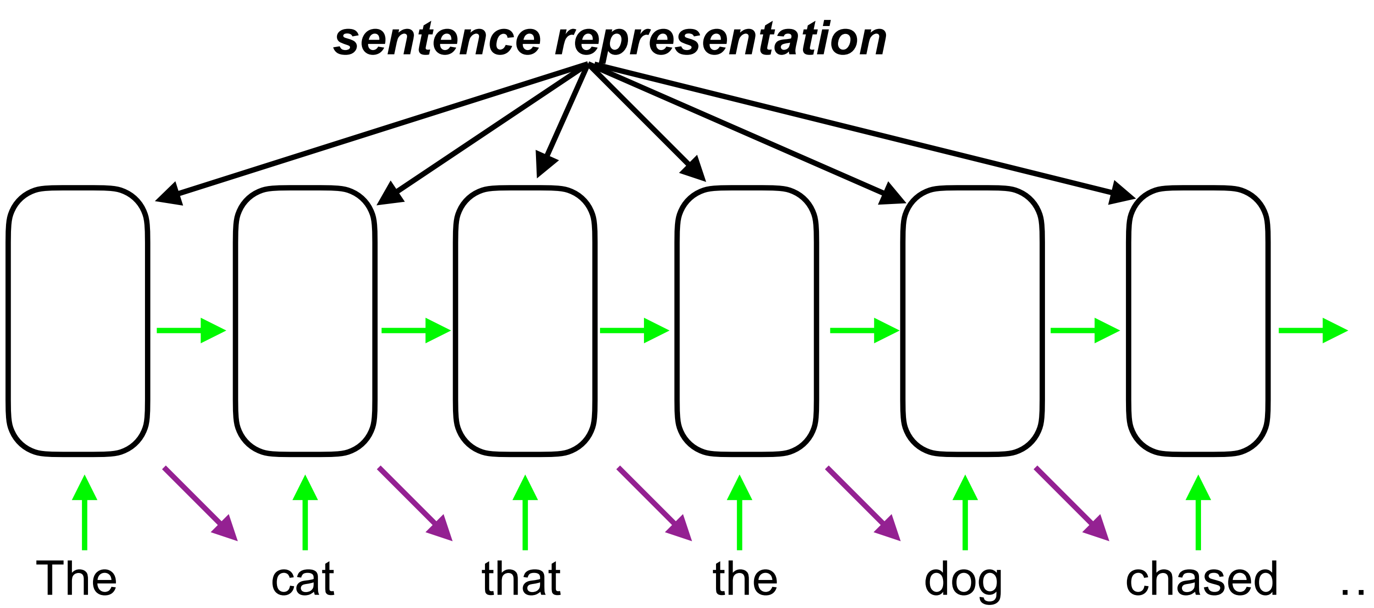
# Discovered similarity structure of words




*(Elman, 1990)*

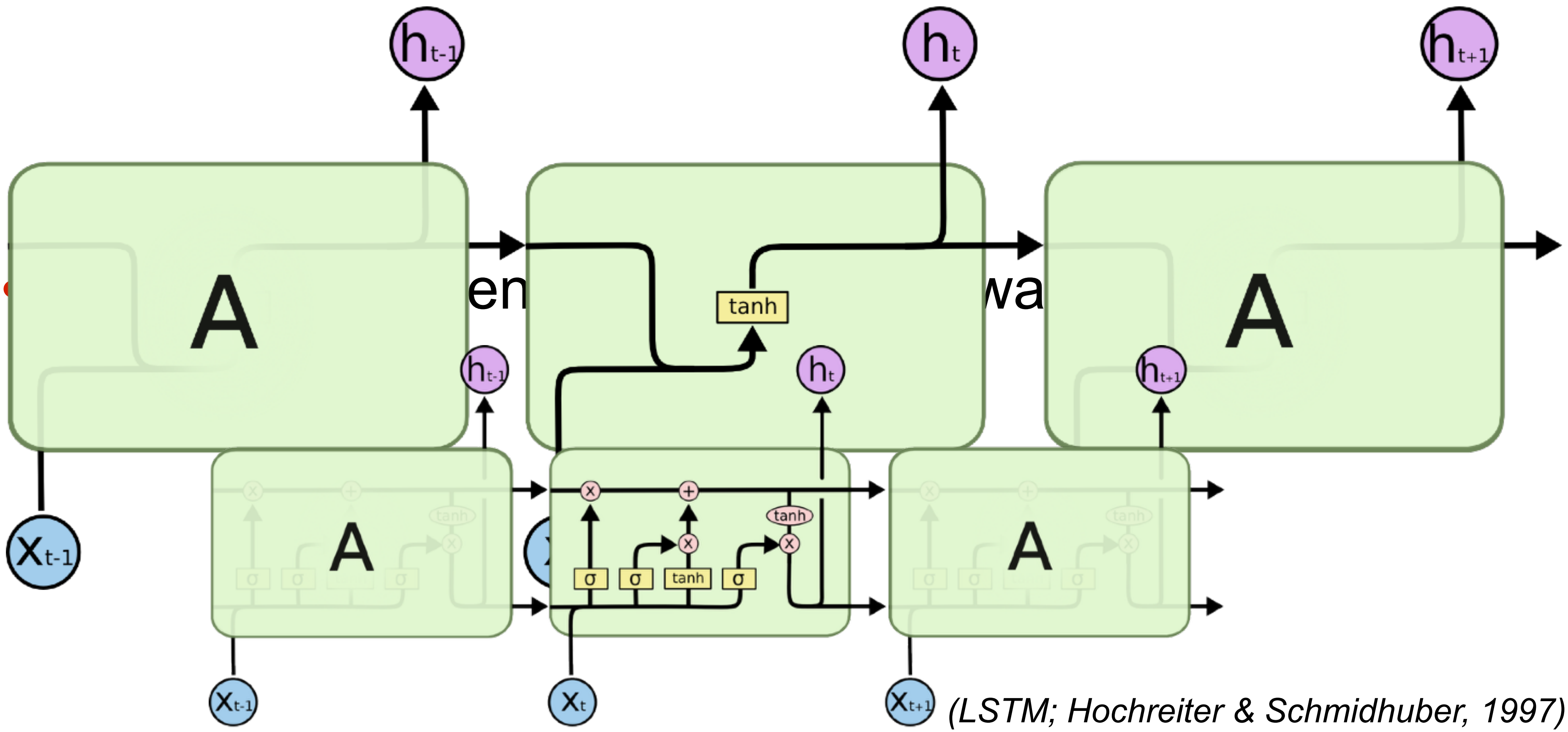
# Beyond the simple recurrent network

- The SRN has a very strong ***linear locality bias***
- But natural language syntax is characterized by ***hierarchical structure***
- SRNs can learn hierarchy (Elman, 1991), but ***it is hard***—their inductive bias disfavors it



# More sophisticated recurrent units

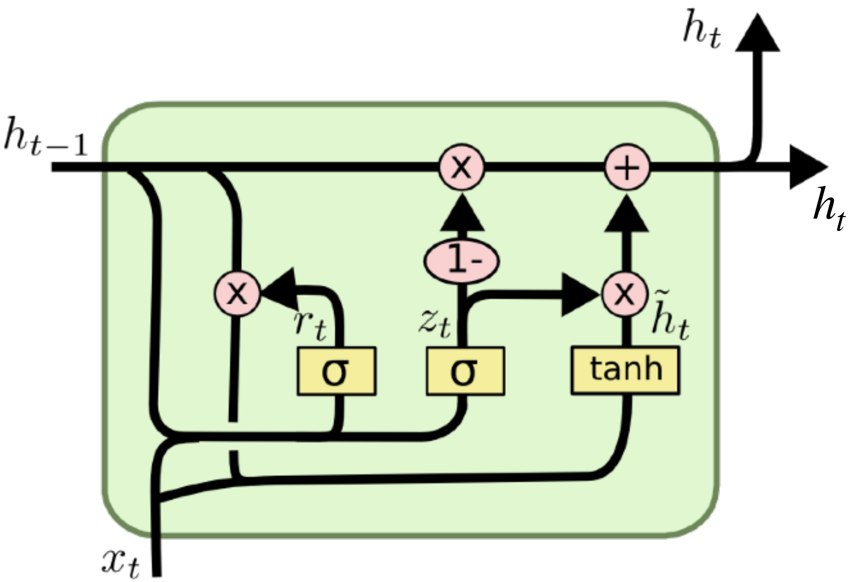
- Another view of an unrolled SRN:




*(LSTM; Hochreiter & Schmidhuber, 1997)*

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Gated Recurrent Unit (GRU) architecture




*logistic/sigmoid activation function*

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1})$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1}))$$

*element-wise multiplication*

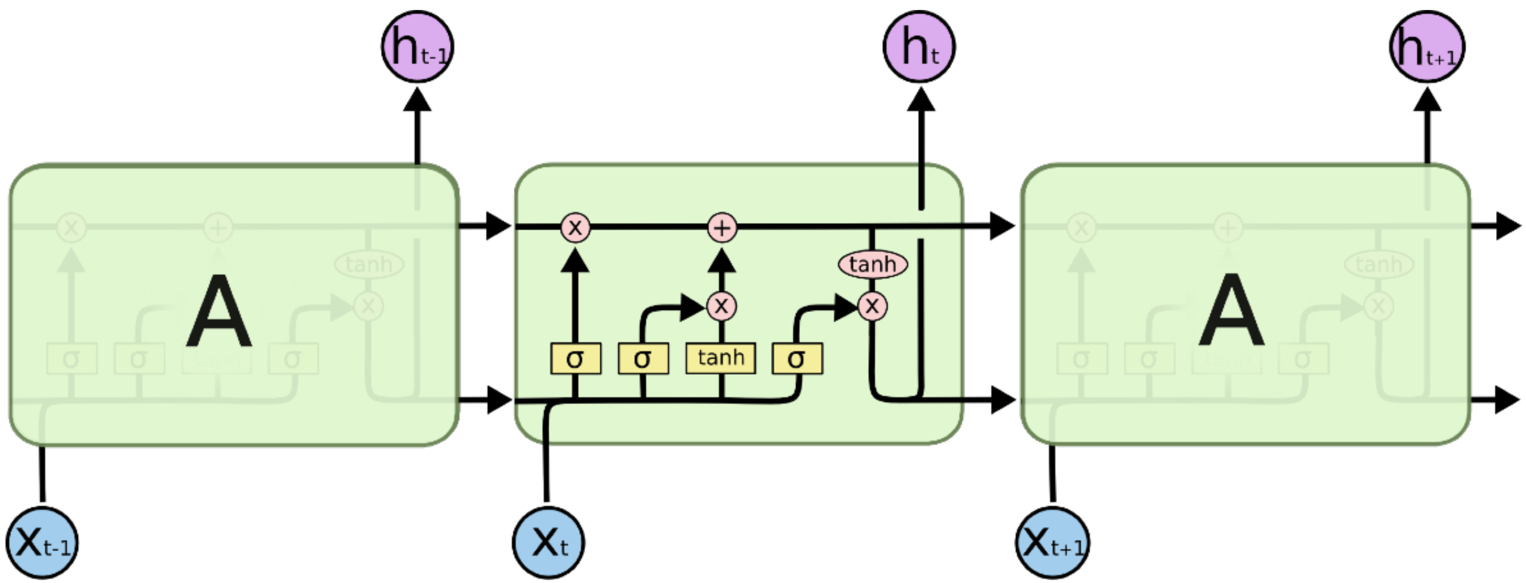
$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1})$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

(e.g., $\langle 1,2,3 \rangle \odot \langle 0.5,2,1 \rangle = \langle 0.5,4,3 \rangle$)

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Long short-term memory (LSTM) units

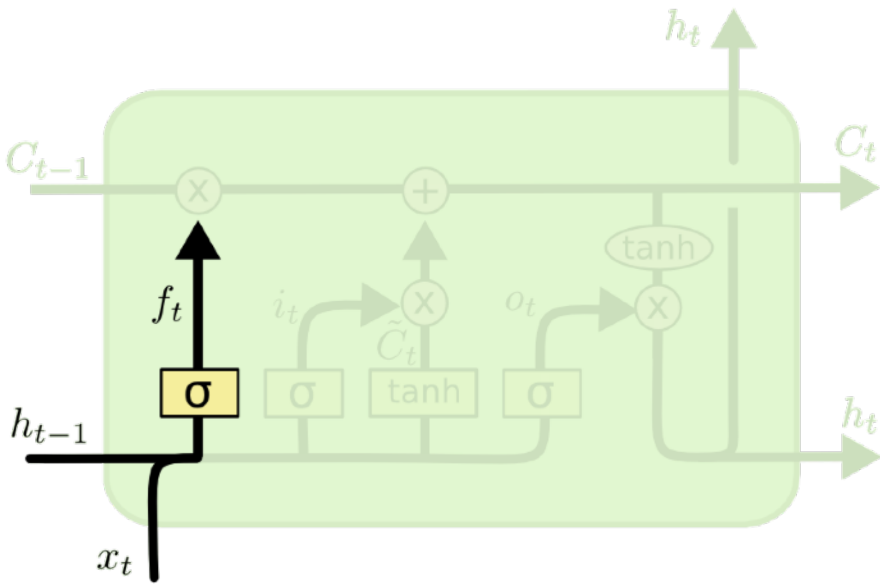



*(Hochreiter & Schmidhuber, 1997)*

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Inside the LSTM unit

- The “hidden layer” $\mathbf{h}_{t-1}$ was used to predict element *t* of the sequence
- It now gets passed through a “forget gate”



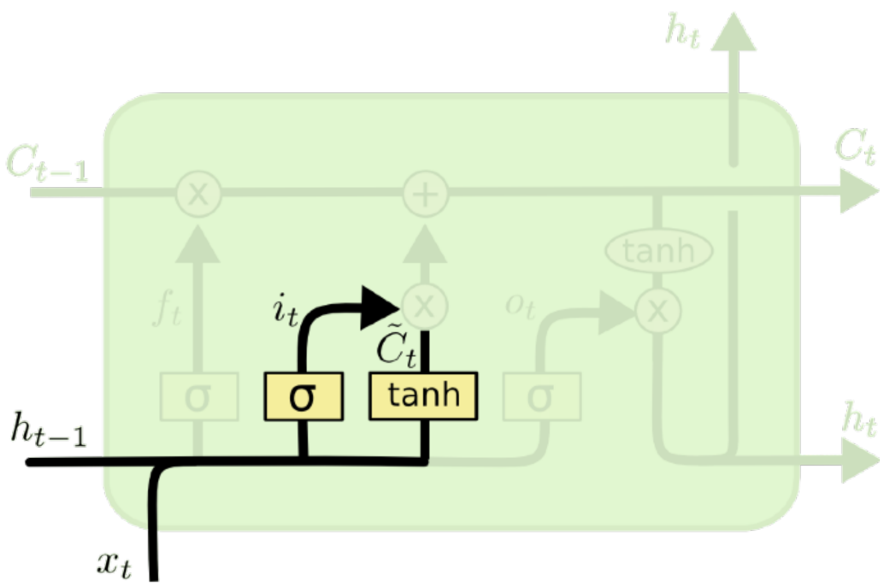

$$\mathbf{f}_t = \sigma(\mathbf{U_f h}_{t-1} + \mathbf{W}_f \mathbf{x}_t)$$

*(Hochreiter & Schmidhuber, 1997)*

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Inside the LSTM unit

- Other information from $h_{t-1}$ gets put into the memory store



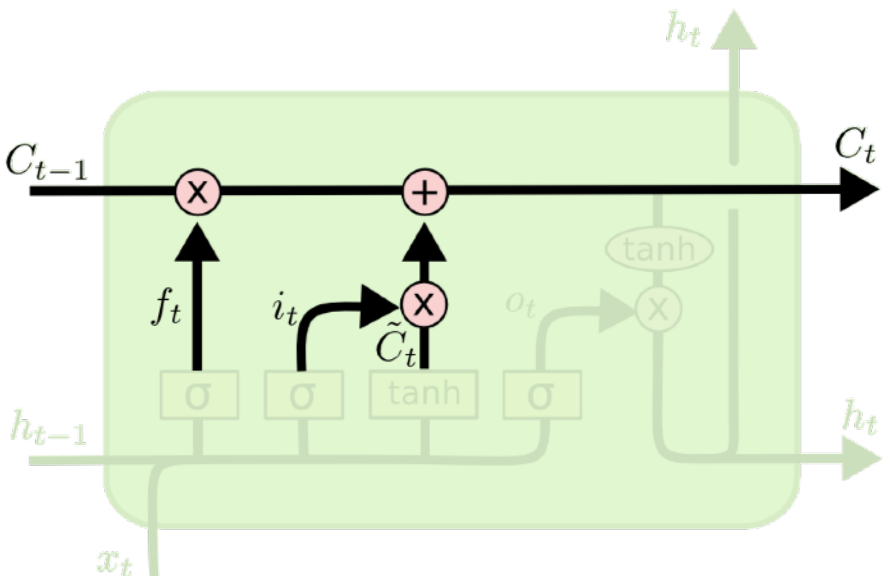

$$\mathbf{i}_t = \sigma(\mathbf{U_i h}_{t-1} + \mathbf{W_i x}_t)$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{U_C h}_{t-1} + \mathbf{W_C x}_t)$$

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Inside the LSTM unit

- That information gets integrated into the memory store (which also gets passed on to the future

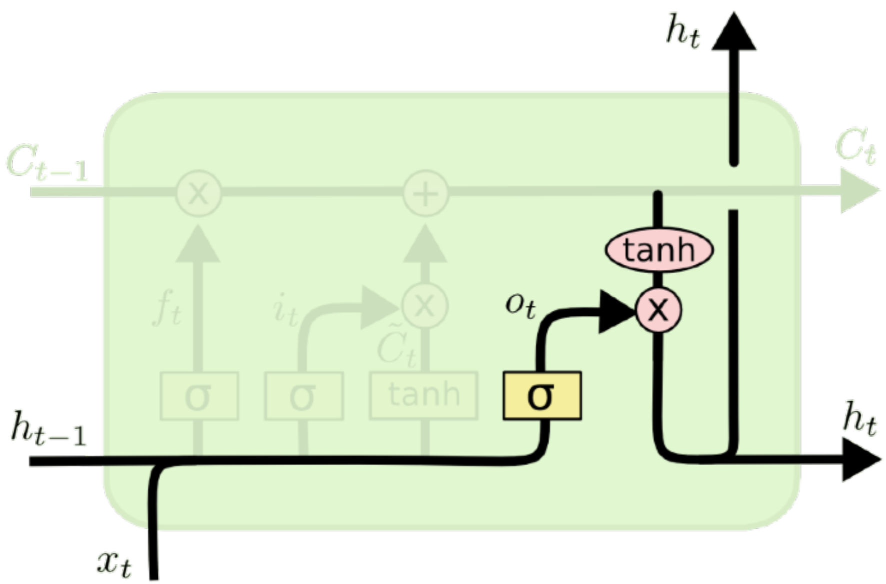



$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t$$

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Inside the LSTM unit

- Finally, we determine the new hidden layer to predict input *t+1*

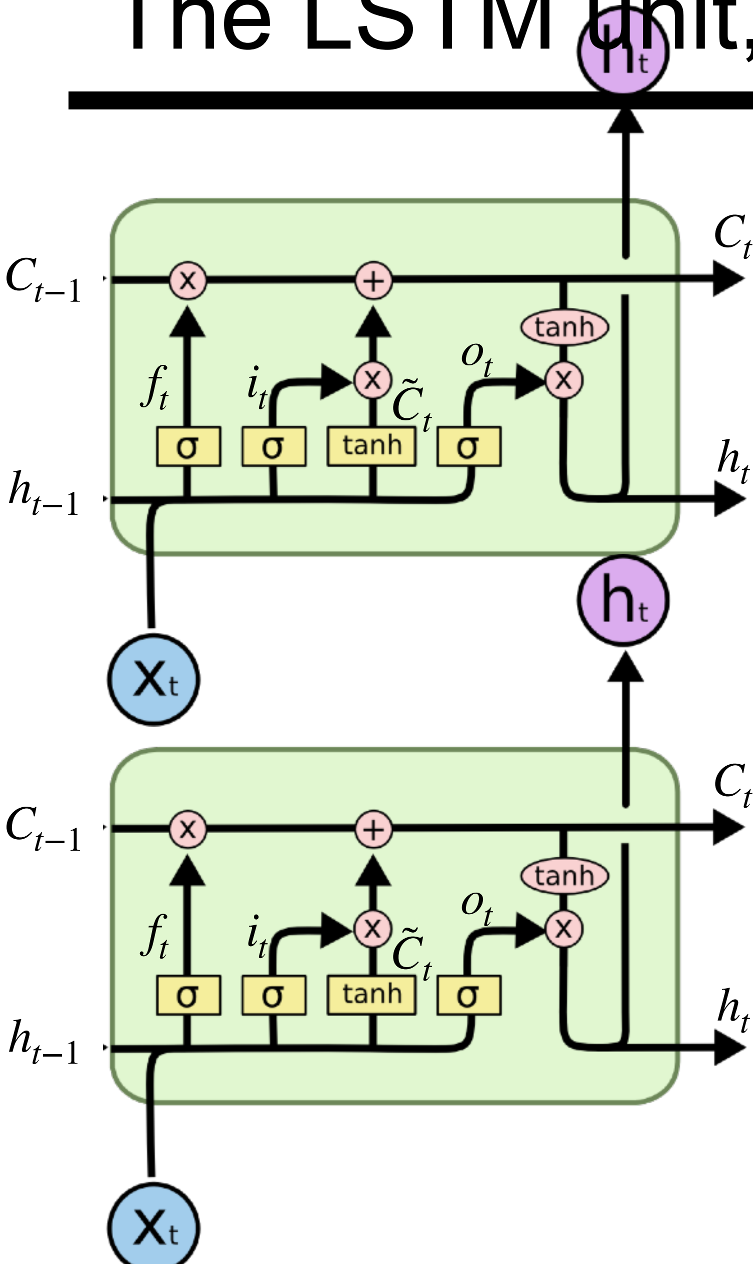



$$\mathbf{o}_t = \sigma(\mathbf{U_o h}_{t-1} + \mathbf{W_o x}_t)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# The LSTM unit, complete




$$\mathbf{f}_t = \sigma(\mathbf{U_f h}_{t-1} + \mathbf{W}_f \mathbf{x}_t)$$

$$\mathbf{i}_t = \sigma(\mathbf{U_i h}_{t-1} + \mathbf{W_i x}_t)$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{U_C h}_{t-1} + \mathbf{W_C x}_t)$$

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t$$

$$\mathbf{o}_t = \sigma(\mathbf{U_o h}_{t-1} + \mathbf{W_o x}_t)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

*visualization due to Christopher Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/*

# Learning the classic counting language

$a^n b^n$

Easily generable with a context-free grammar:

$$S \rightarrow a\ b$$
$$S \rightarrow a\ S\ b$$

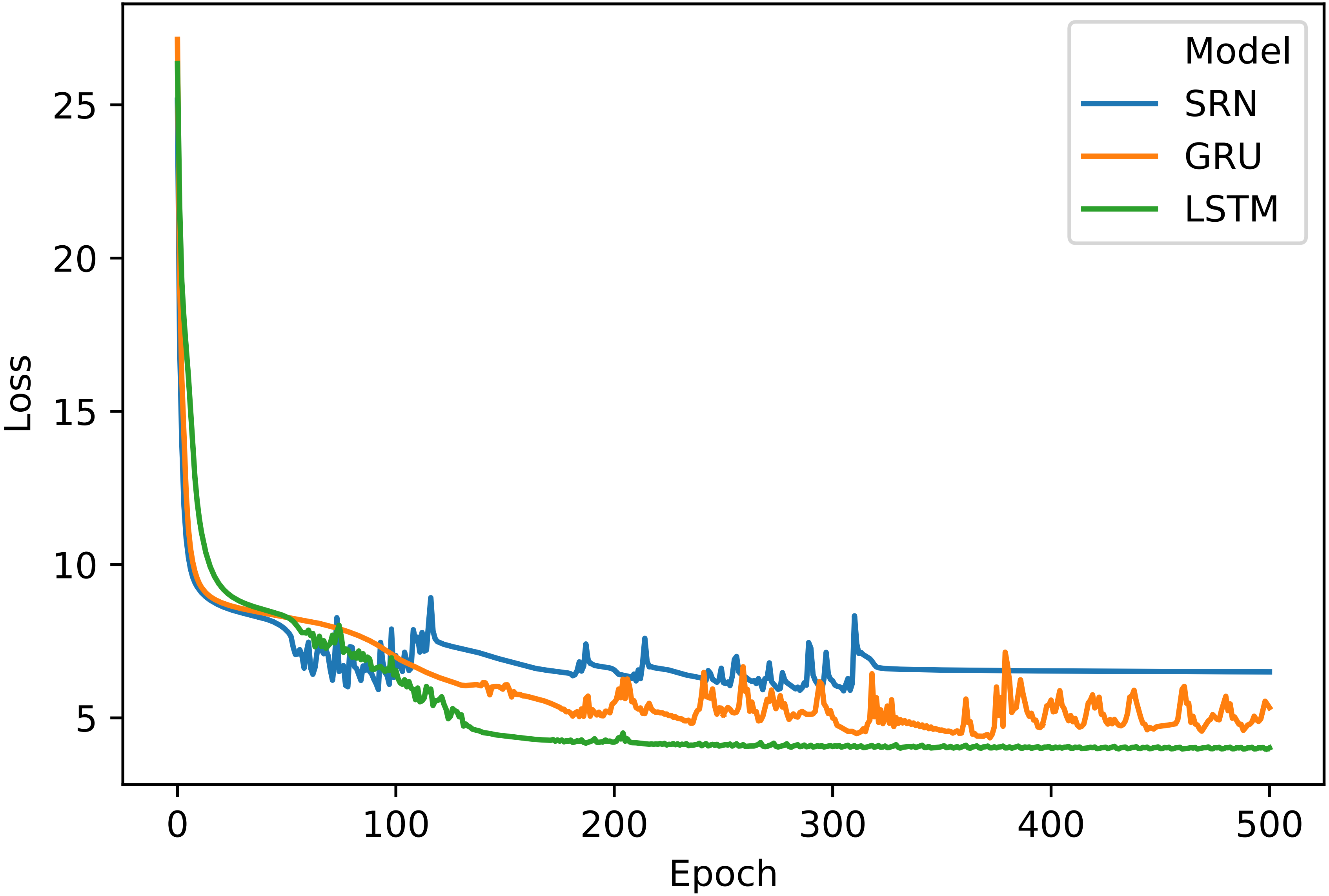
$D_{train}$

```
^ab$
^aabb$
^aaabbb$
^aaaabbbb$
^aaaaabbbbb$
^aaaaaabbbbbb$
^aaaaaaabbbbbbb$
⋮
```

$$\text{\^{}}\underbrace{\texttt{aaaaaaaaaaaaaaaaaaaa}}_{N=20}\underbrace{\texttt{bbbbbbbbbbbbbbbbbbbb}}_{N=20}\texttt{\$}$$

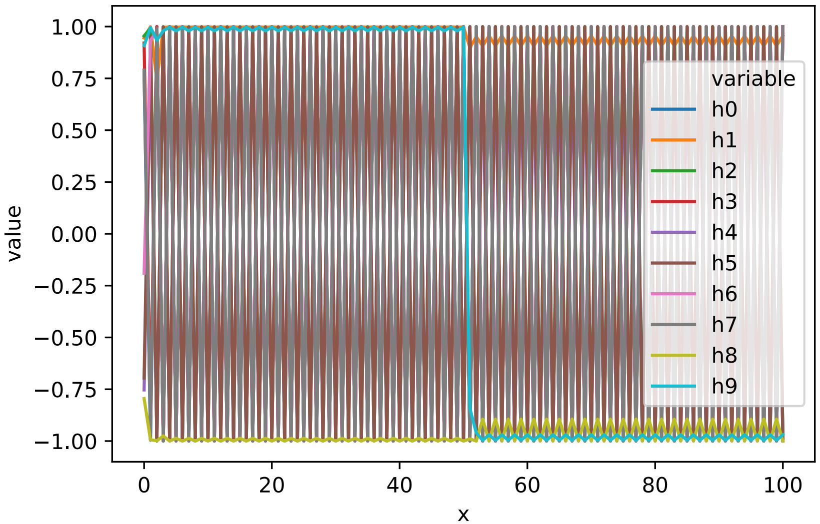
(Weiss et al. 2018, ACL)

# Training recurrent architectures on $a^n b^n$

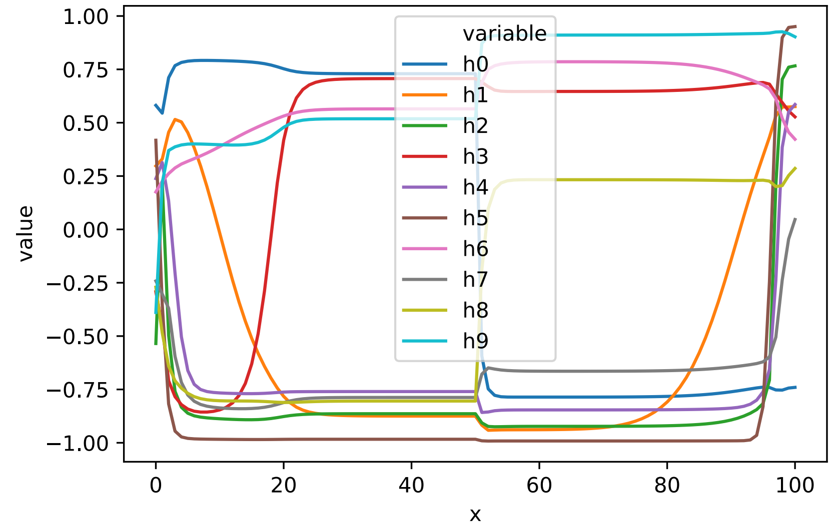


# Hidden & cell state contents

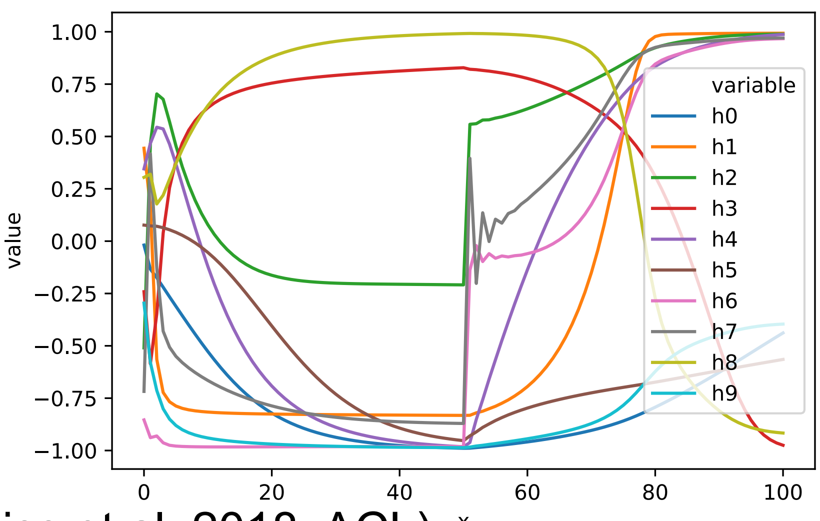
## SRN hidden state



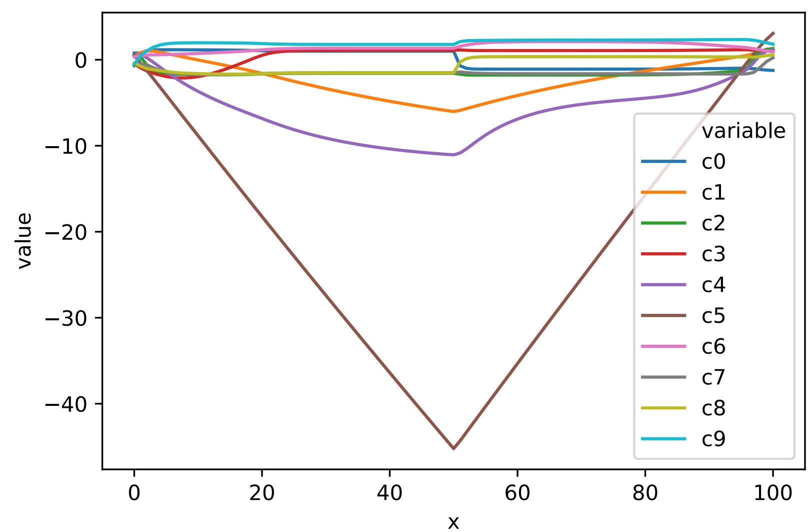

## LSTM hidden state




## GRU hidden state




## LSTM cell state




(Weiss et al. 2018, ACL)

# Summary

- Mechanisms for neural networks at the sentence level:
  - Learned word embeddings
  - Recurrent state representation
- Different units used for recurrent state representation:
  - Simple recurrent network (SRN)
  - Gated recurrent unit (GRU)
  - Long short-term memory (LSTM)
- For classic counting language, LSTM works the best