Linguistic Knowledge in Neural Networks

Chris Dyer

DeepMind

Carnegie Mellon University

What is linguistics?

- How do human languages represent meaning?
- How does the mind/brain process and generate language?
- What are the possible/impossible human languages?
- **How do children learn language from a very small sample of data?**

An important insight
Sentences are hierarchical

(1) a. The talk I gave did **not** appeal to **anybody**.

Examples adapted from Everaert et al. (*TICS* 2015)

An important insight
Sentences are hierarchical

(1) a. The talk I gave did **not** appeal to **anybody**.
 b. *The talk I gave appealed to **anybody**.

Examples adapted from Everaert et al. (*TICS* 2015)

An important insight
Sentences are hierarchical

NPI

(1) a. The talk I gave did **not** appeal to **anybody**.

b. *The talk I gave appealed to **anybody**.

Examples adapted from Everaert et al. (*TICS* 2015)

An important insight
Sentences are hierarchical

NPI

(1) a. The talk I gave did **not** appeal to **anybody**.

b. *The talk I gave appealed to **anybody**.

Generalization hypothesis: **not** must come before **anybody**

Examples adapted from Everaert et al. (*TICS* 2015)

An important insight
Sentences are hierarchical

(1) a. The talk I gave did **not** appeal to **anybody**.

b. *The talk I gave appealed to **anybody**.

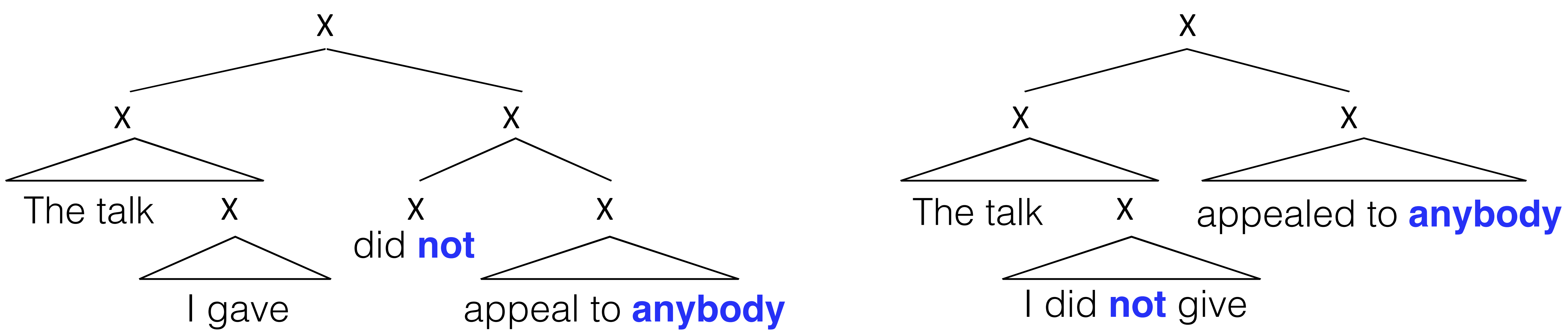
NPI

Generalization hypothesis: **not** must come before **anybody**

(2) *The talk I did **not** give appealed to **anybody**.

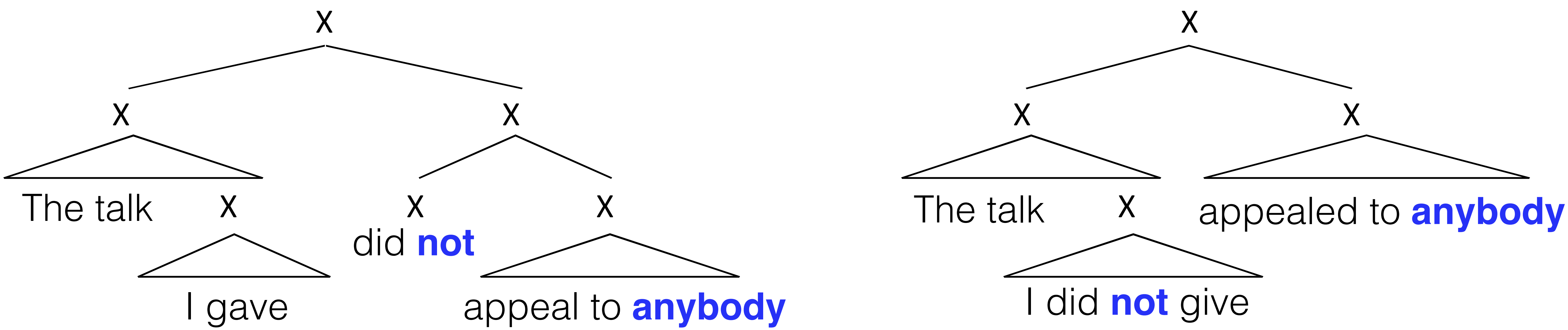
Examples adapted from Everaert et al. (*TICS* 2015)

Language is hierarchical



Examples adapted from Everaert et al. (*TICS* 2015)

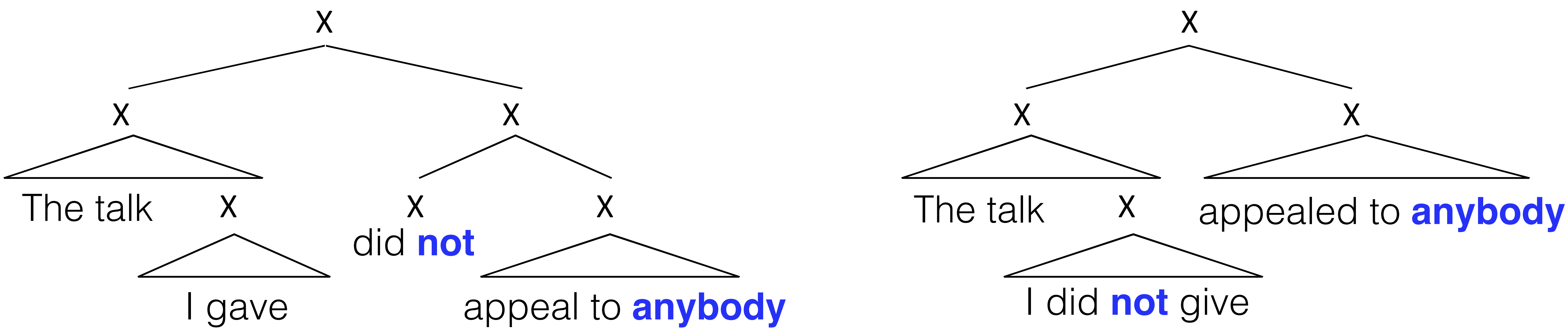
Language is hierarchical



Generalization: **not** must “structurally precede” **anybody**

Examples adapted from Everaert et al. (*TICS* 2015)

Language is hierarchical



Generalization: **not** must "structurally precede" **anybody**

- the psychological reality of **structural sensitivty** is **not** empirically controversial
  - many different theories of the *details* of structure
- hypothesis: kids learn language easily because they don't consider many "obvious" structurally insensitive hypotheses

Examples adapted from Everaert et al. (*TICS* 2015)

Recurrent neural networks
A good model of language?

- Recurrent neural networks are incredibly powerful models of sequences (e.g., of words)

  - In fact, RNNs are Turing complete!
    (Siegelmann, 1995)

Recurrent neural networks
A good model of language?

- Recurrent neural networks are incredibly powerful models of sequences (e.g., of words)
  - In fact, RNNs are Turing complete! (Siegelmann, 1995)
- But do they make good generalizations from **finite** samples of data?
  - **What inductive biases do they have?**
  - **What assumptions about representations do models that use them make?**

Recurrent neural networks
Inductive bias

- Understanding the biases of neural networks is tricky
    - We have enough trouble understanding the representations they learn in specific cases, much less general cases!

Recurrent neural networks
Inductive bias

- Understanding the biases of neural networks is tricky
  - We have enough trouble understanding the representations they learn in specific cases, much less general cases!
- But, there is lots of evidence RNNs prefer **sequential recency**
  - Evidence 1: Gradients become attenuated across time
    - Analysis; experiments with synthetic datasets
      (yes, LSTMs help but they have limits)
  - Evidence 2: Training regimes like reversing sequences in seq2seq learning
  - Evidence 3: Modeling enhancements to use attention (direct connections back in remote time)

Recurrent neural networks
Inductive bias

- Understanding the biases of neural networks is tricky
    - We have enough trouble understanding the representations they learn in specific cases, much less general cases!
- But, there is lots of evidence RNNs prefer **sequential recency**
    - Evidence 1: Gradients become attenuated across time
        - Analysis; experiments with synthetic datasets
          (yes, LSTMs help but they have limits)
    - Evidence 2: Training regimes like reversing sequences in seq2seq learning
    - Evidence 3: Modeling enhancements to use attention (direct connections back in remote time)
- **Chomsky** (to crudely paraphrase 60 years of work):
  sequential recency is not the right bias for effective learning of human language.

Topics

- Recursive neural networks for sentence representation
- Recurrent neural network grammars and parsing
- Word representations by looking inside words (words have structure too!)
- Analysis of neural networks with linguistic concepts

Topics

- Recursive neural networks for sentence representation
- Recurrent neural network grammars and parsing
- Word representations by looking inside words (words have structure too!)
- Analysis of neural networks with linguistic concepts

Representing a sentence

Input sentence:

this film is hardly a treat

Task: Classify this sentence as having either positive or negative sentiment.

Representing a sentence

Input sentence:

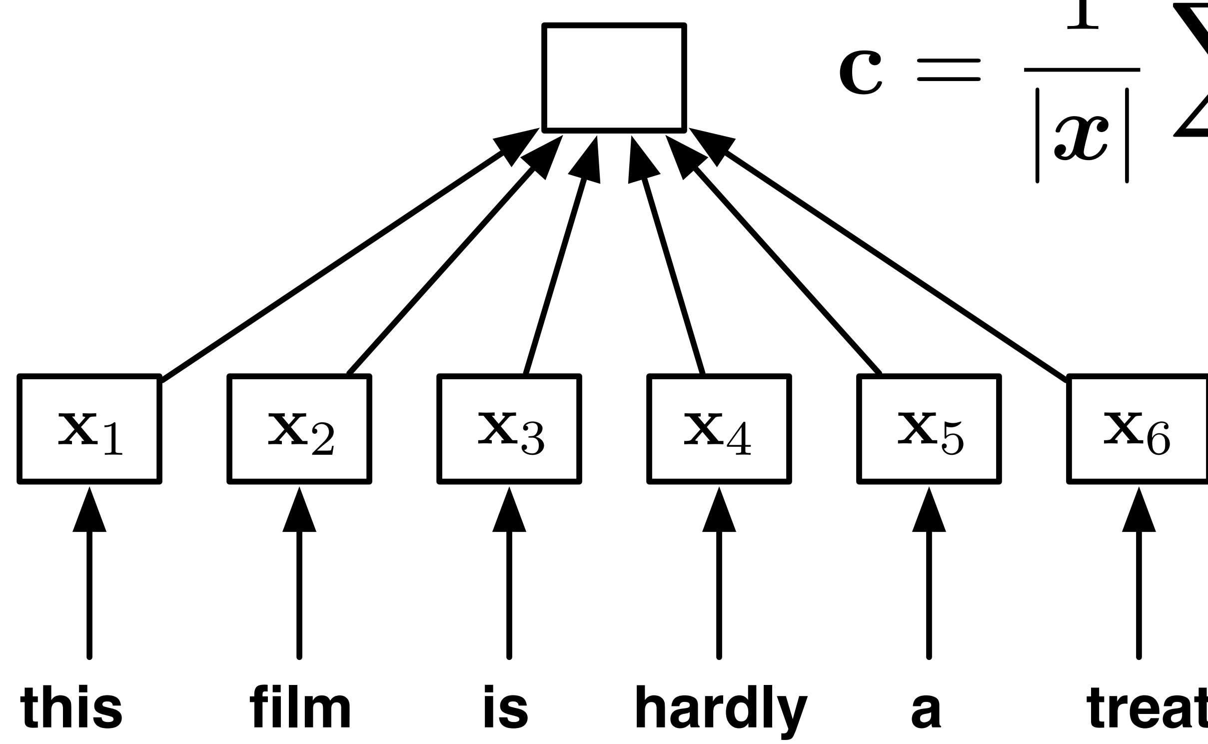
this film is hardly a treat

Task: Classify this sentence as having either positive or negative sentiment.

Why might this sentence pose a problem for interpretation?

Representing a sentence

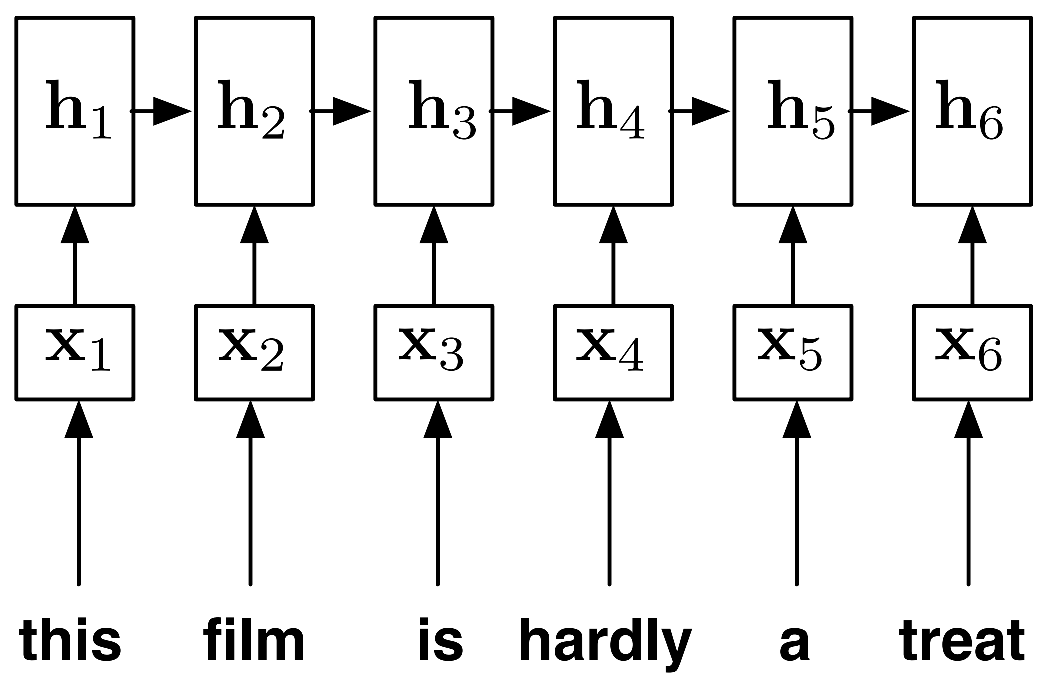
Bag of words:



$$\mathbf{c} = \frac{1}{|\boldsymbol{x}|}\sum_i \mathbf{x}_i$$

Representing a sentence

Recurrent neural network



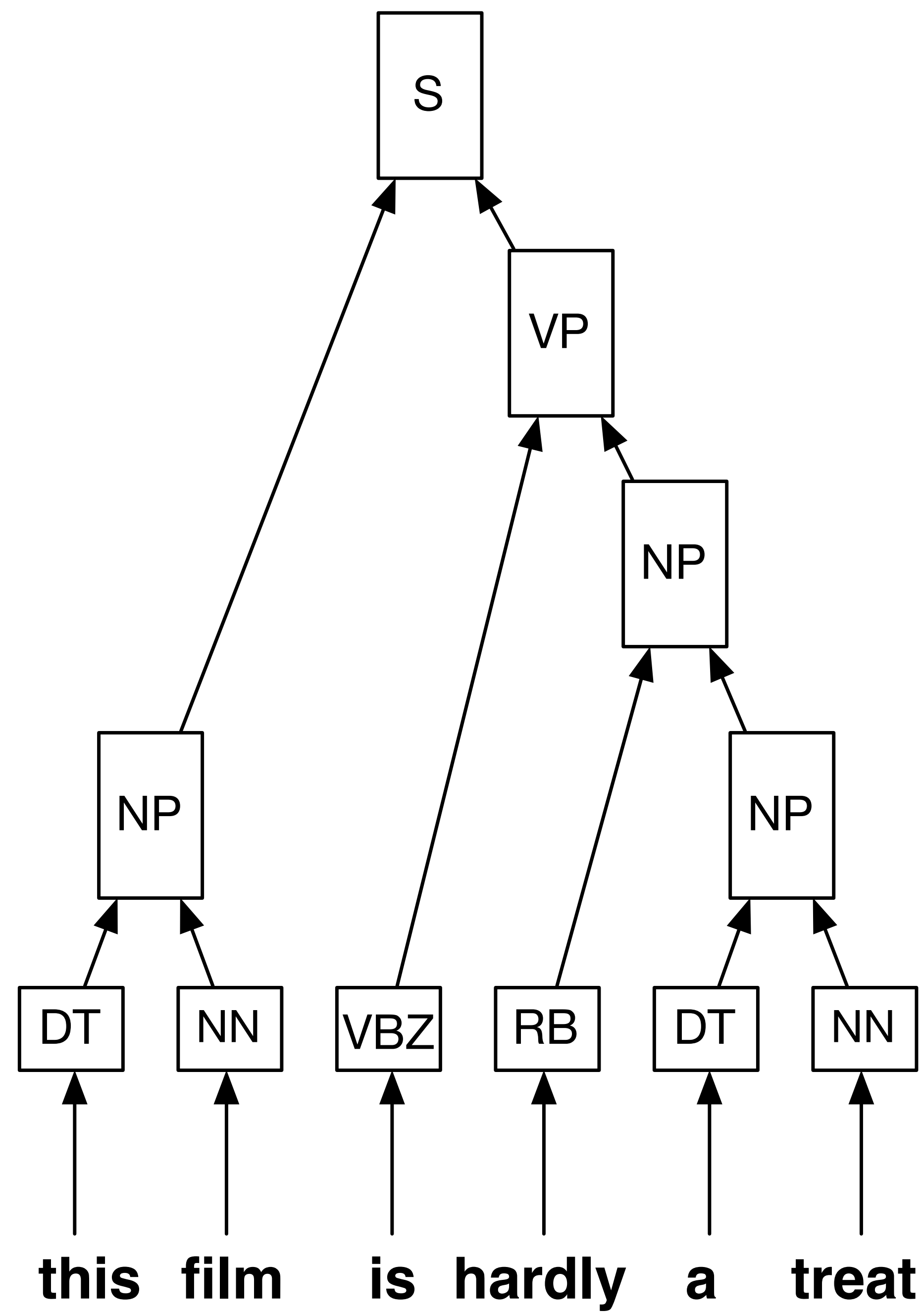
$$\mathbf{c} = \mathbf{h}_6$$

How do languages express meaning?

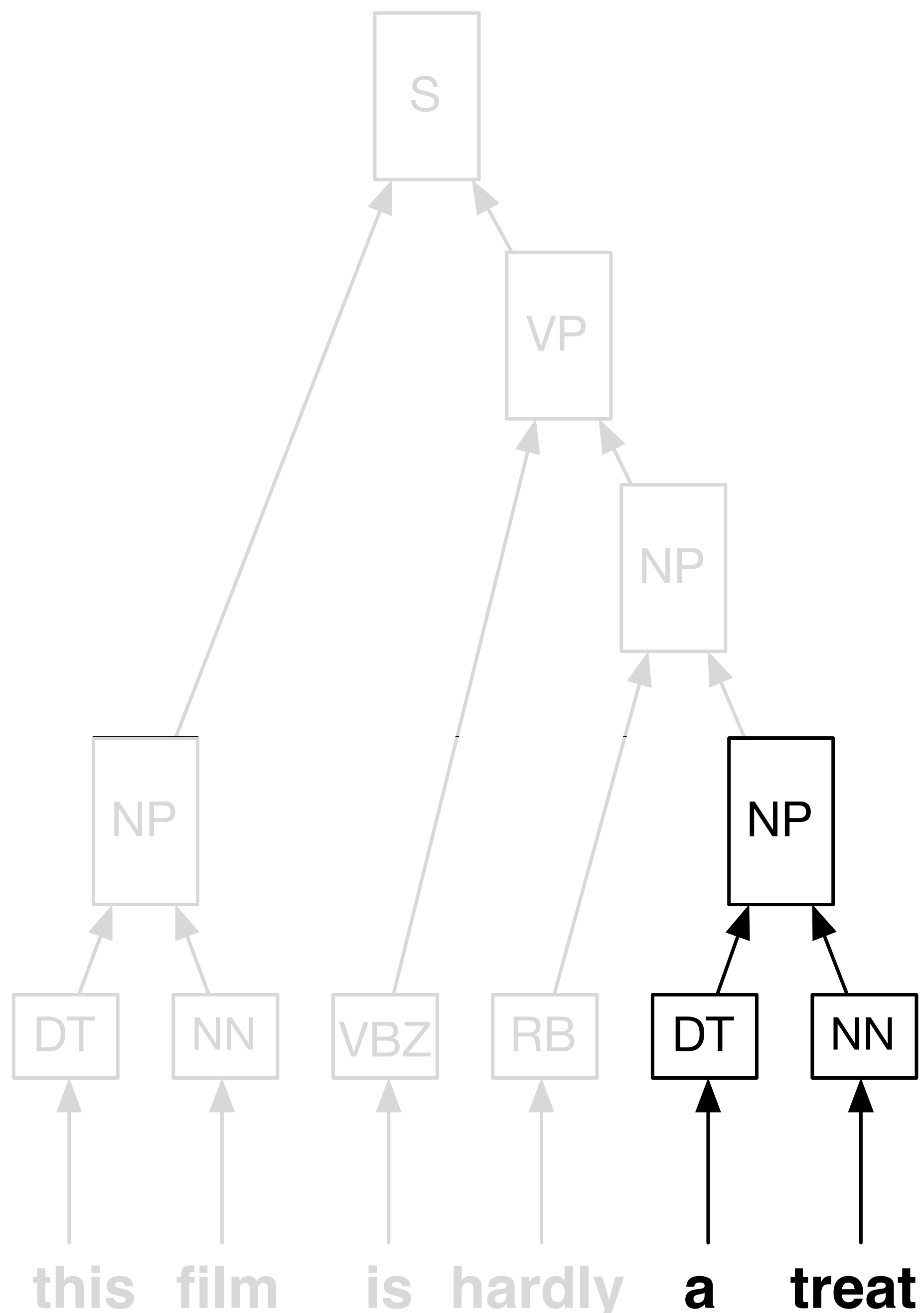
- **Principle of compositionality**: the meaning of a complex expression is determined by the meanings of its **constituent expressions** and the **rules that combine** them.

- Syntax and parsing
  - Syntax is the study of how words fit together to form phrases and ultimately sentences
  - We can use **syntactic parsing** to decompose sentences into constituent expressions and rules that were used to construct them out of more primitive expressions (and ultimately individual words)

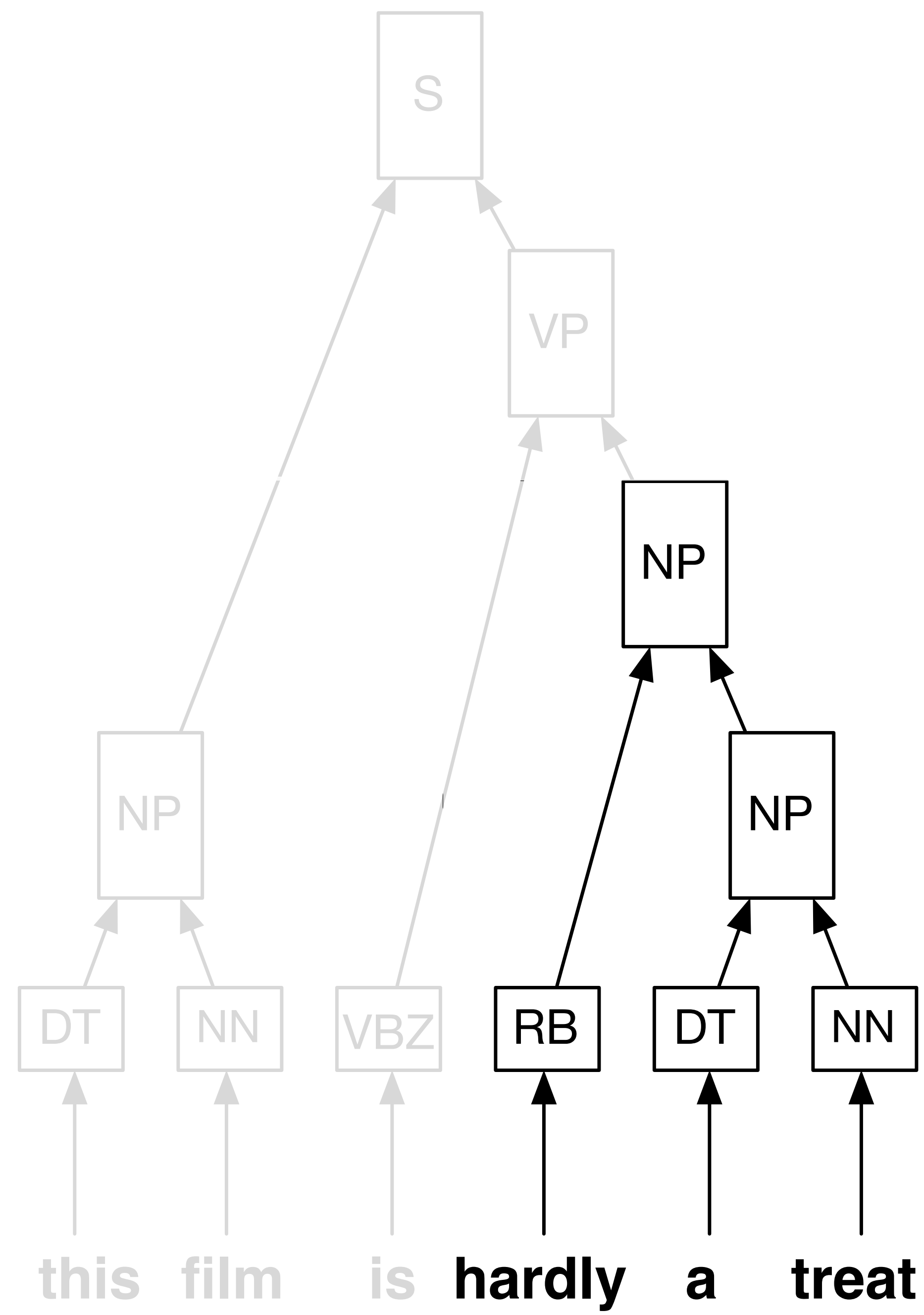
Syntax as Trees



Syntax as Trees

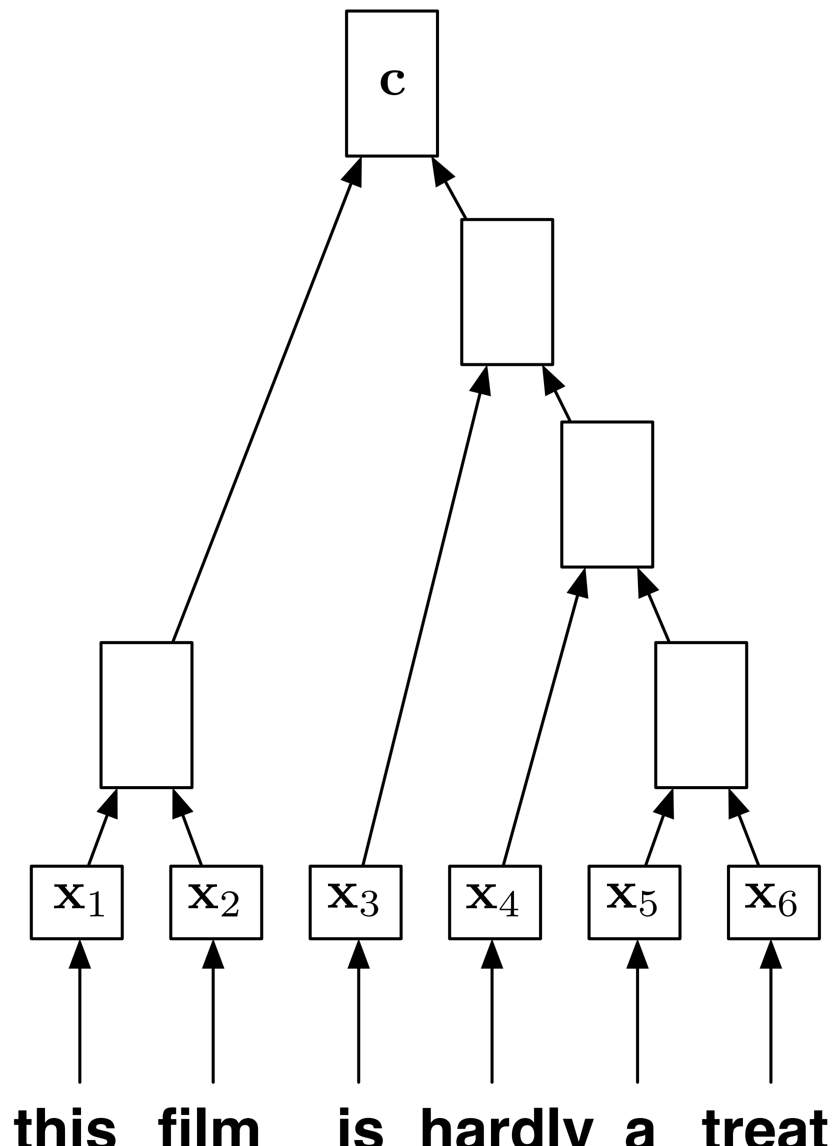


Syntax as Trees



Representing a sentence

Recursive Neural Network

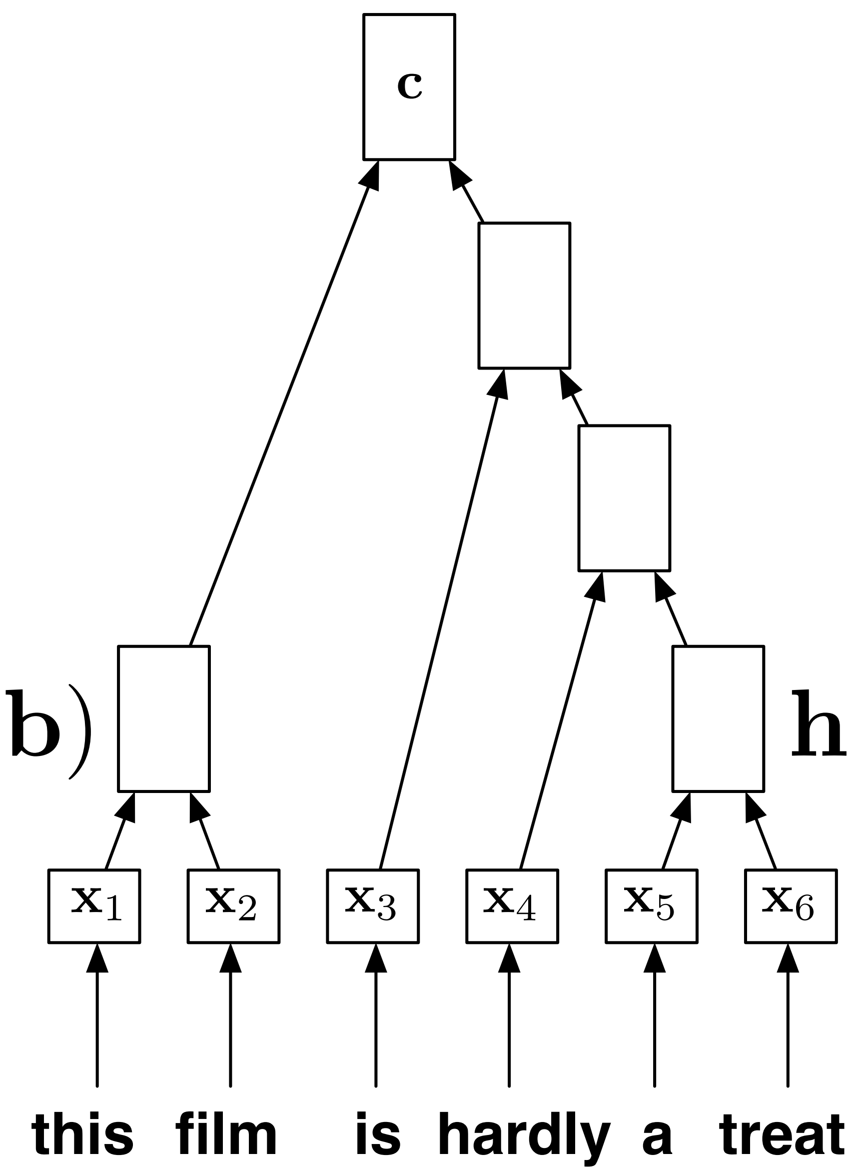


Socher et al. (ICML 2011), *et passim*

Representing a sentence

Recursive Neural Network

$$\mathbf{h} = \tanh(\mathbf{W}[\boldsymbol{\ell}; \mathbf{r}] + \mathbf{b})$$

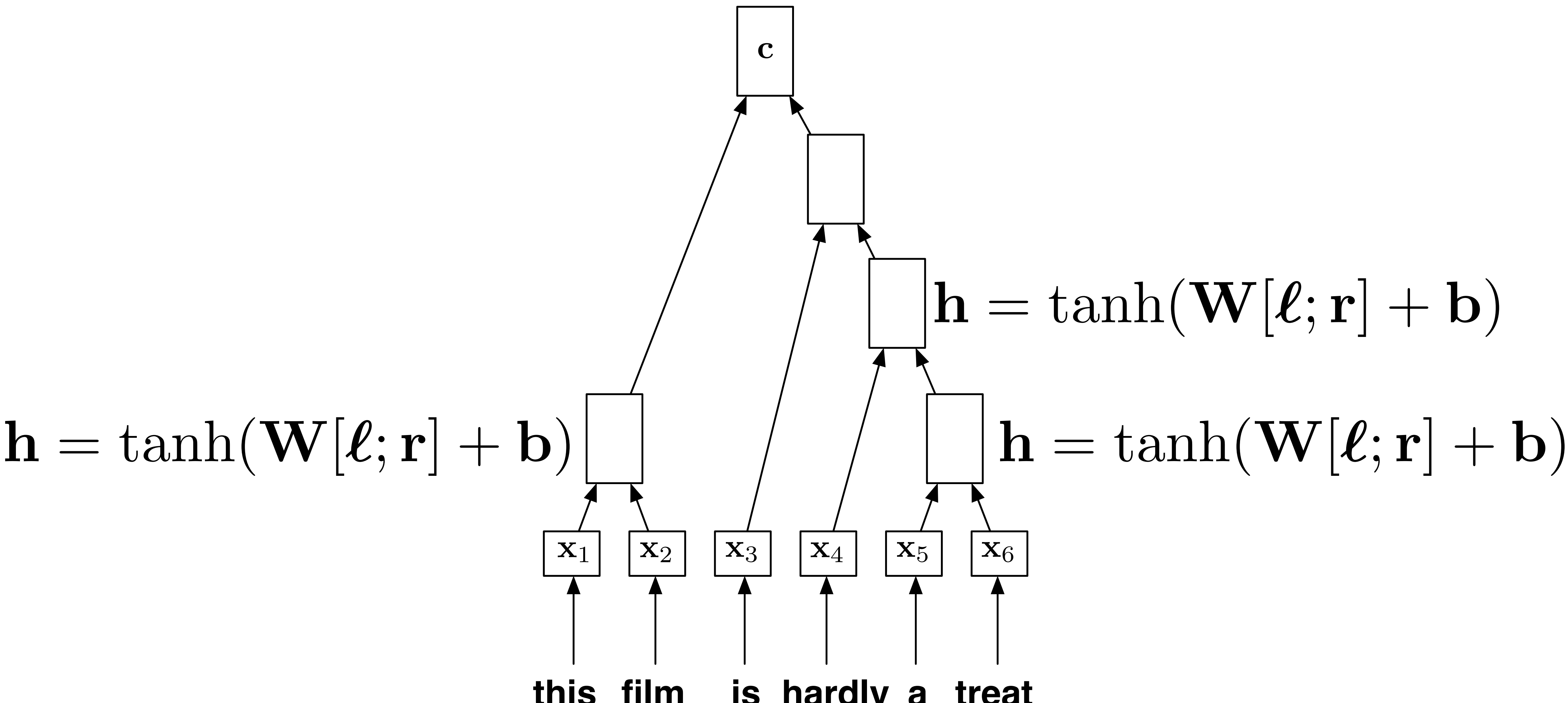


$$\mathbf{h} = \tanh(\mathbf{W}[\boldsymbol{\ell}; \mathbf{r}] + \mathbf{b})$$

Socher et al. (ICML 2011), *et passim*

Representing a sentence

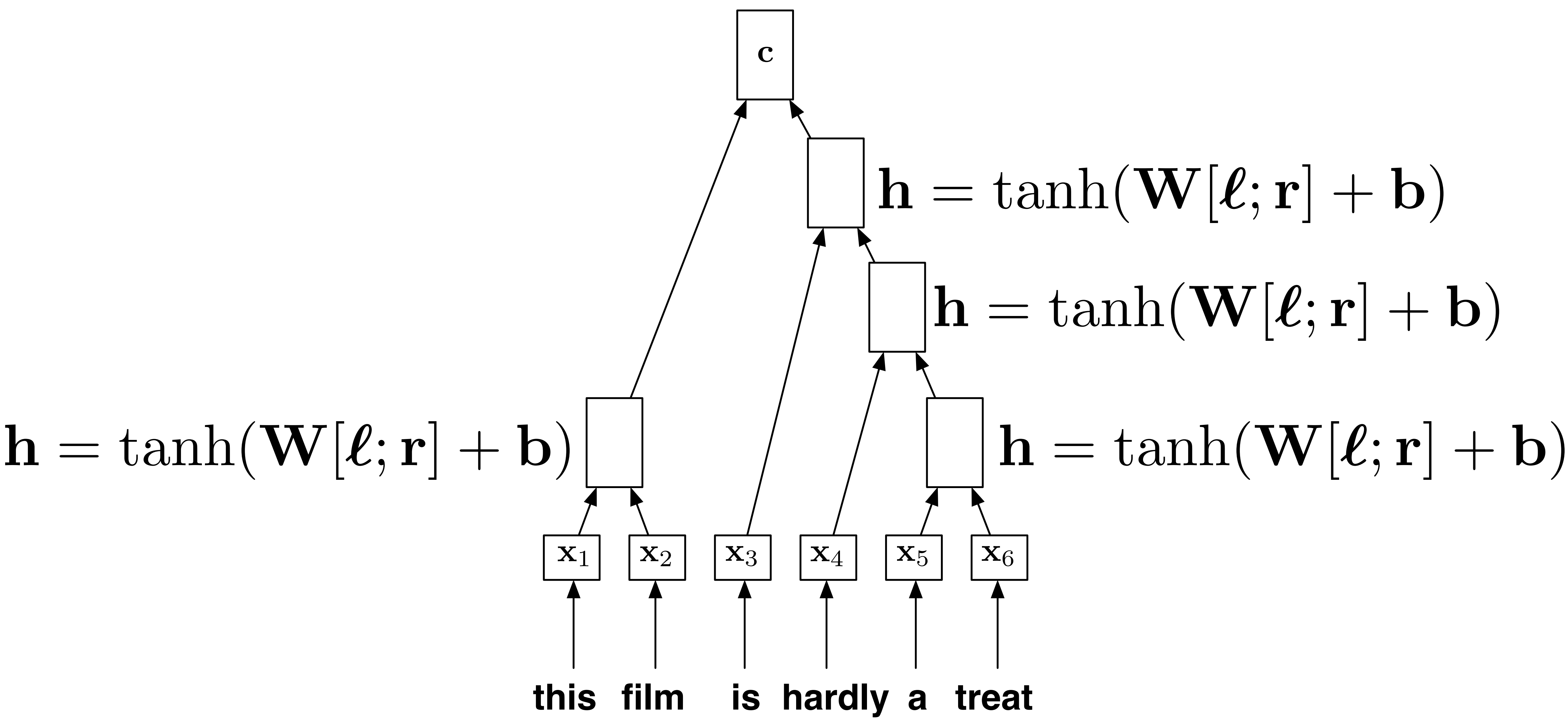
Recursive Neural Network



Socher et al. (ICML 2011), *et passim*

Representing a sentence

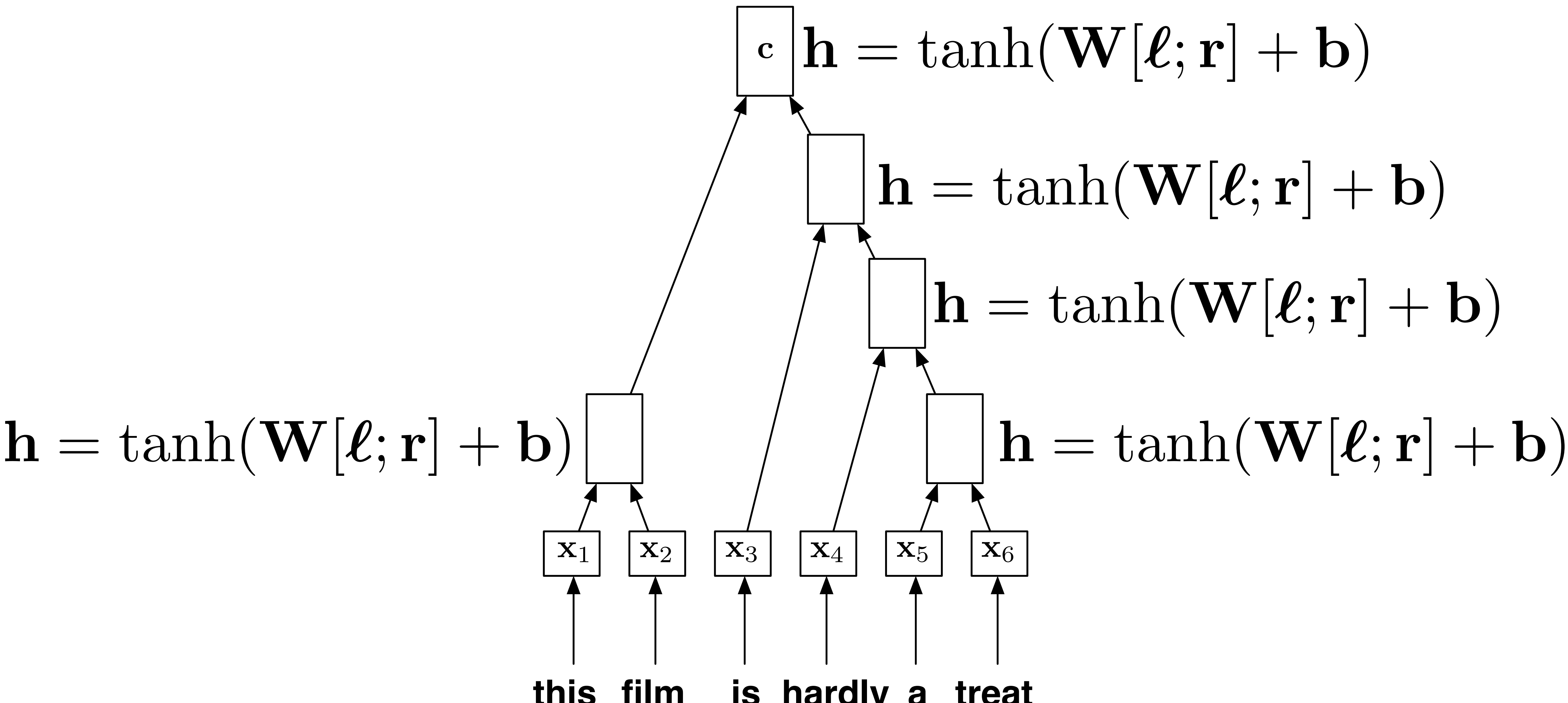
Recursive Neural Network



Socher et al. (ICML 2011), *et passim*

Representing a sentence

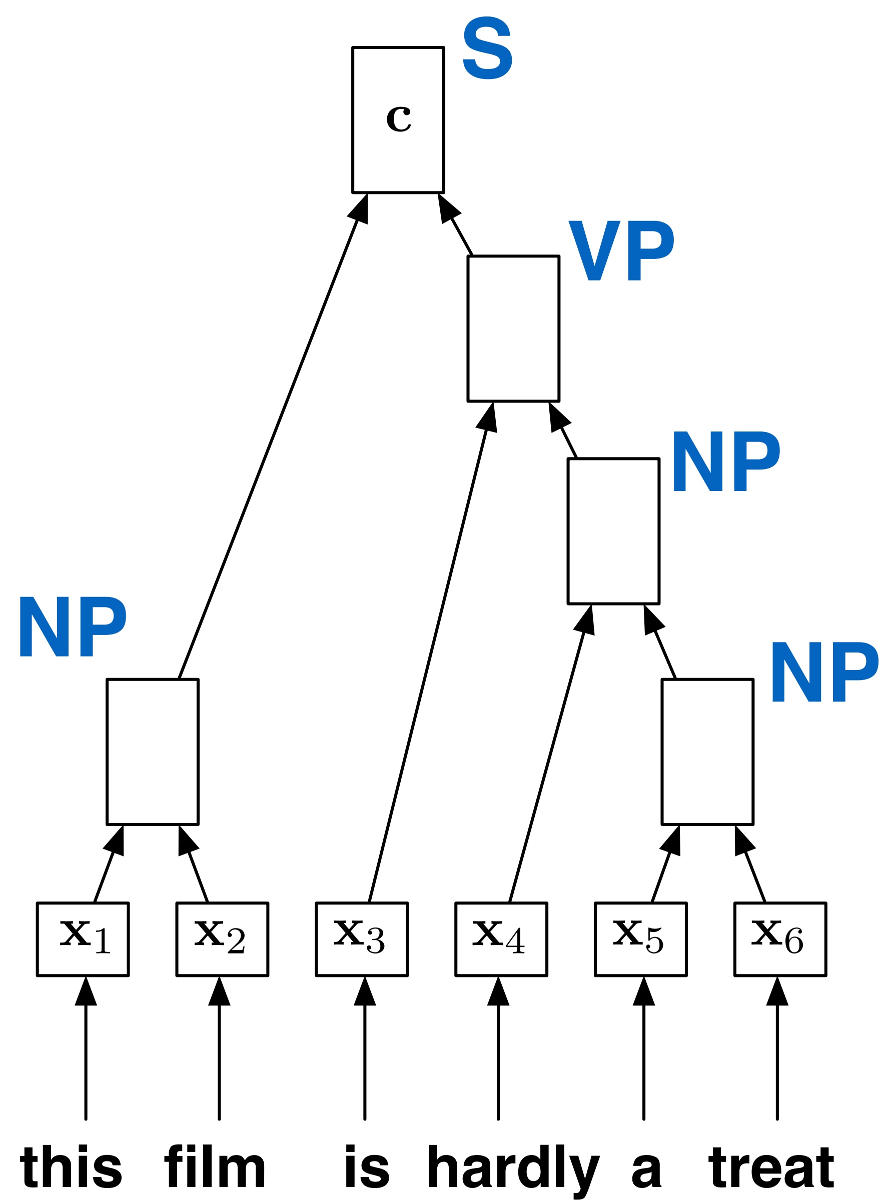
Recursive Neural Network



Socher et al. (ICML 2011), *et passim*

Representing a sentence

Recursive Neural Network

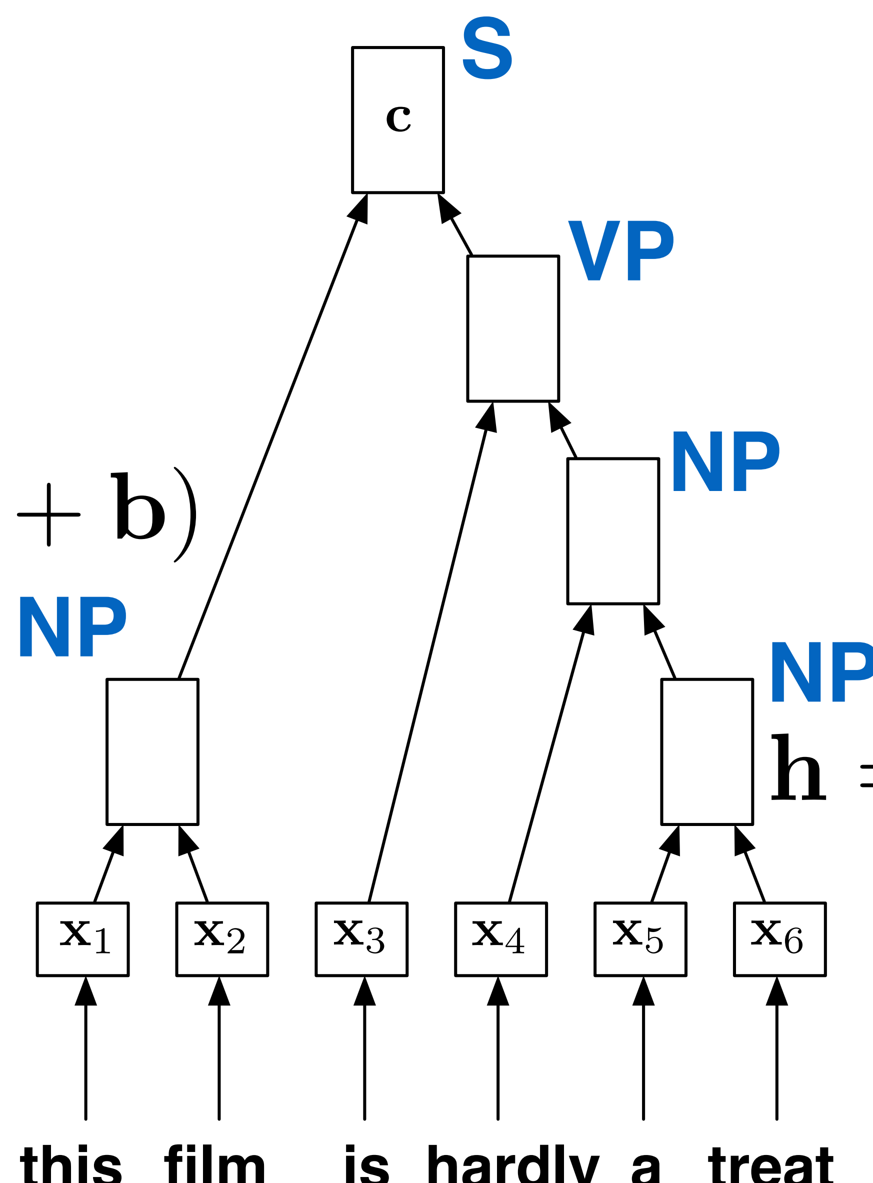


“Syntactic untying”

Representing a sentence

Recursive Neural Network

$$\mathbf{h} = \tanh(\mathbf{W}^{\mathrm{NP}}[\boldsymbol{\ell};\mathbf{r}] + \mathbf{b})$$

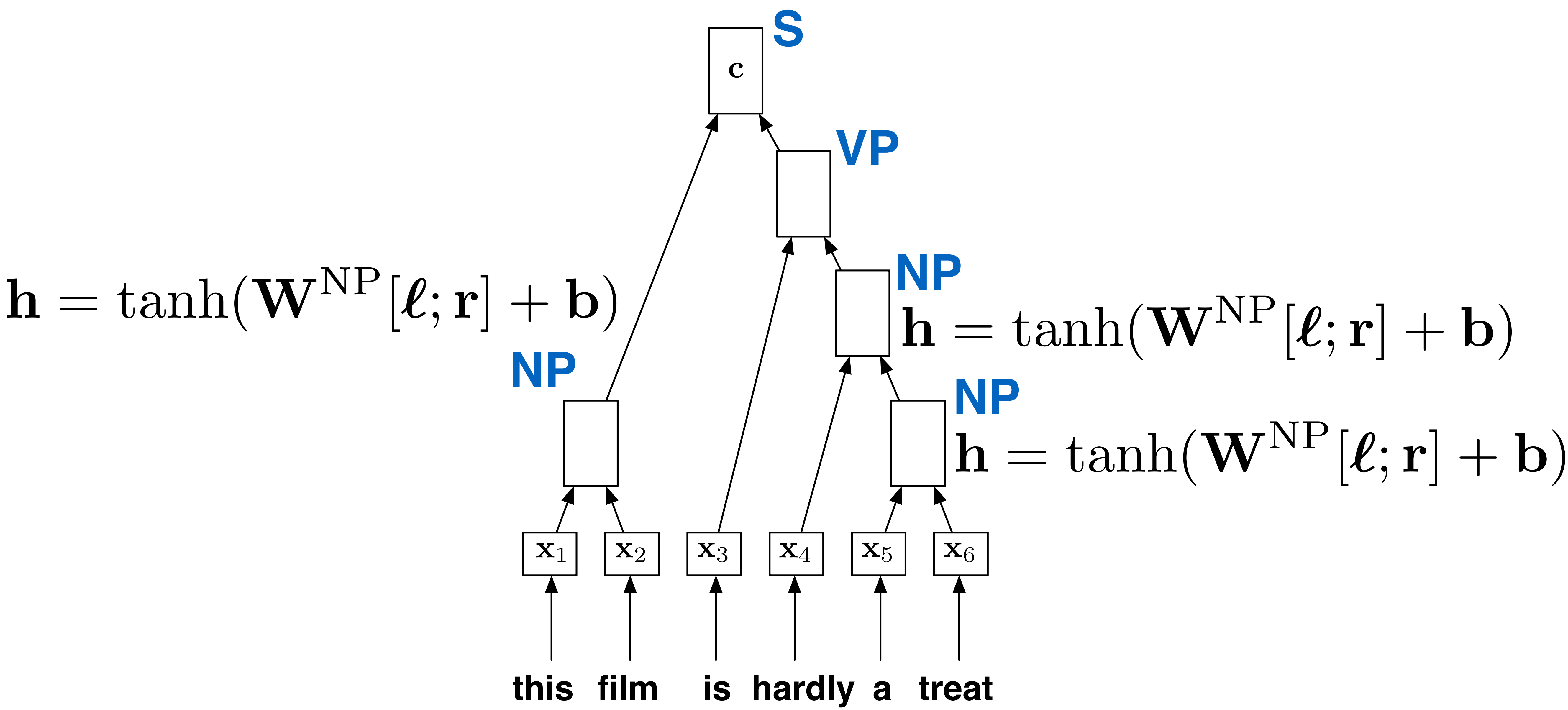


$$\mathbf{h} = \tanh(\mathbf{W}^{\mathrm{NP}}[\boldsymbol{\ell};\mathbf{r}] + \mathbf{b})$$

“Syntactic untying”

Representing a sentence

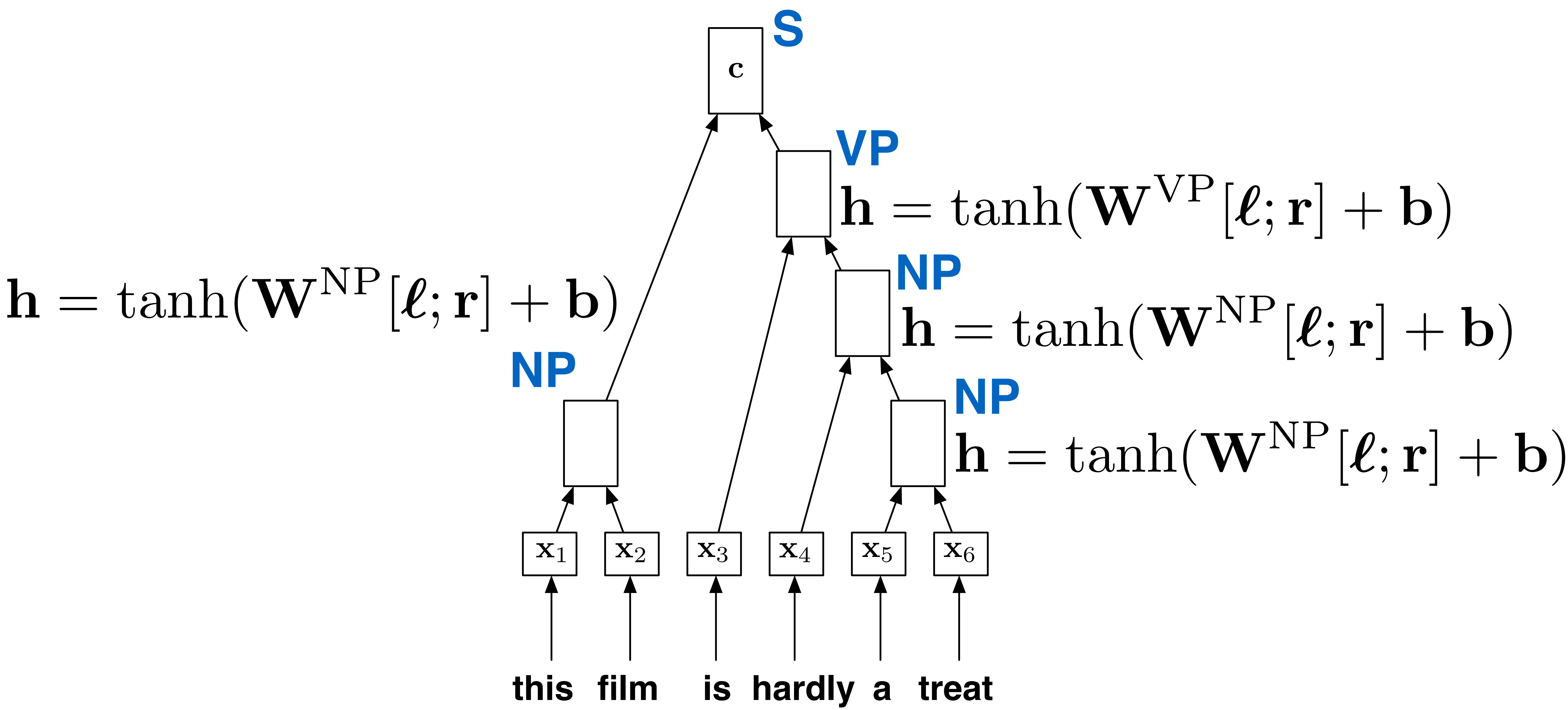
Recursive Neural Network



"Syntactic untying"

Representing a sentence

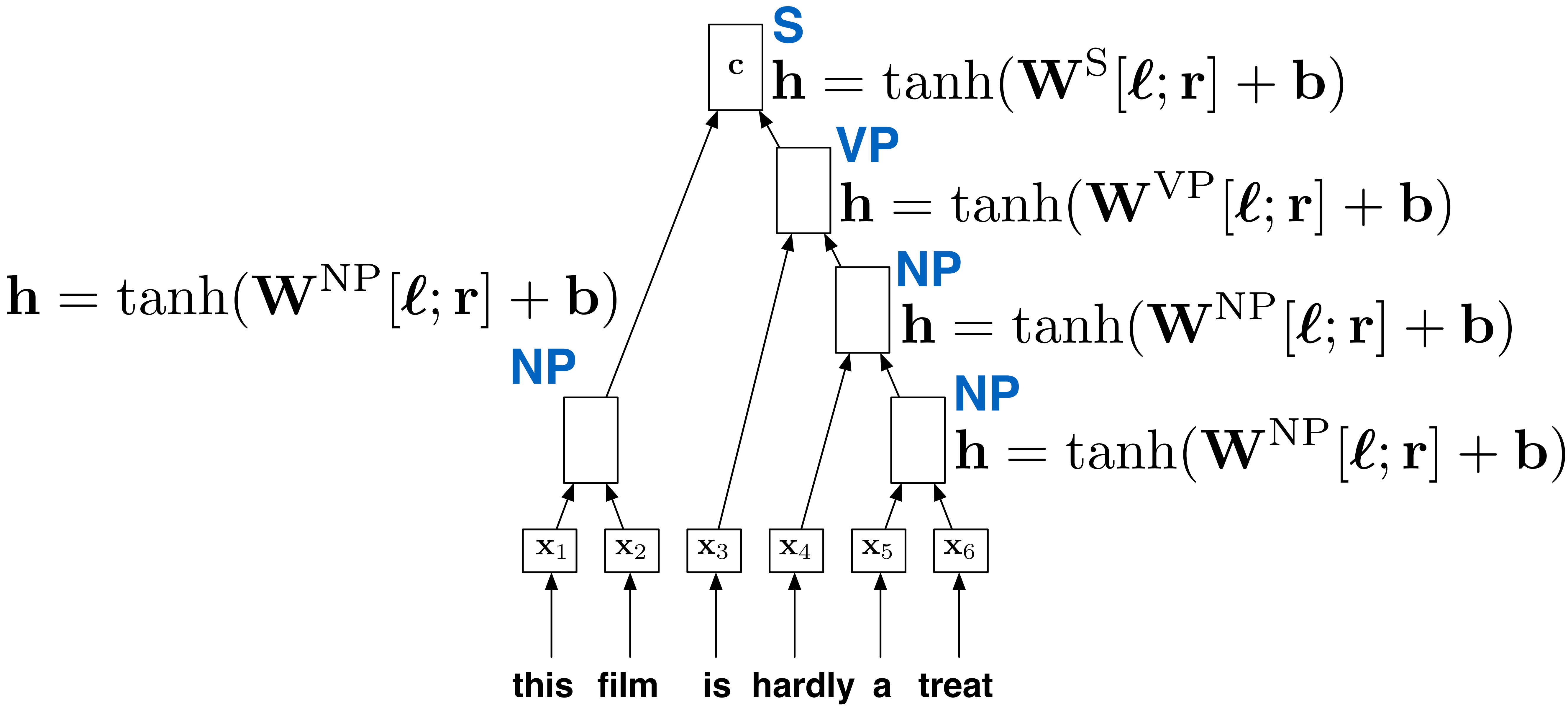
Recursive Neural Network



"Syntactic untying"

Representing a sentence

Recursive Neural Network

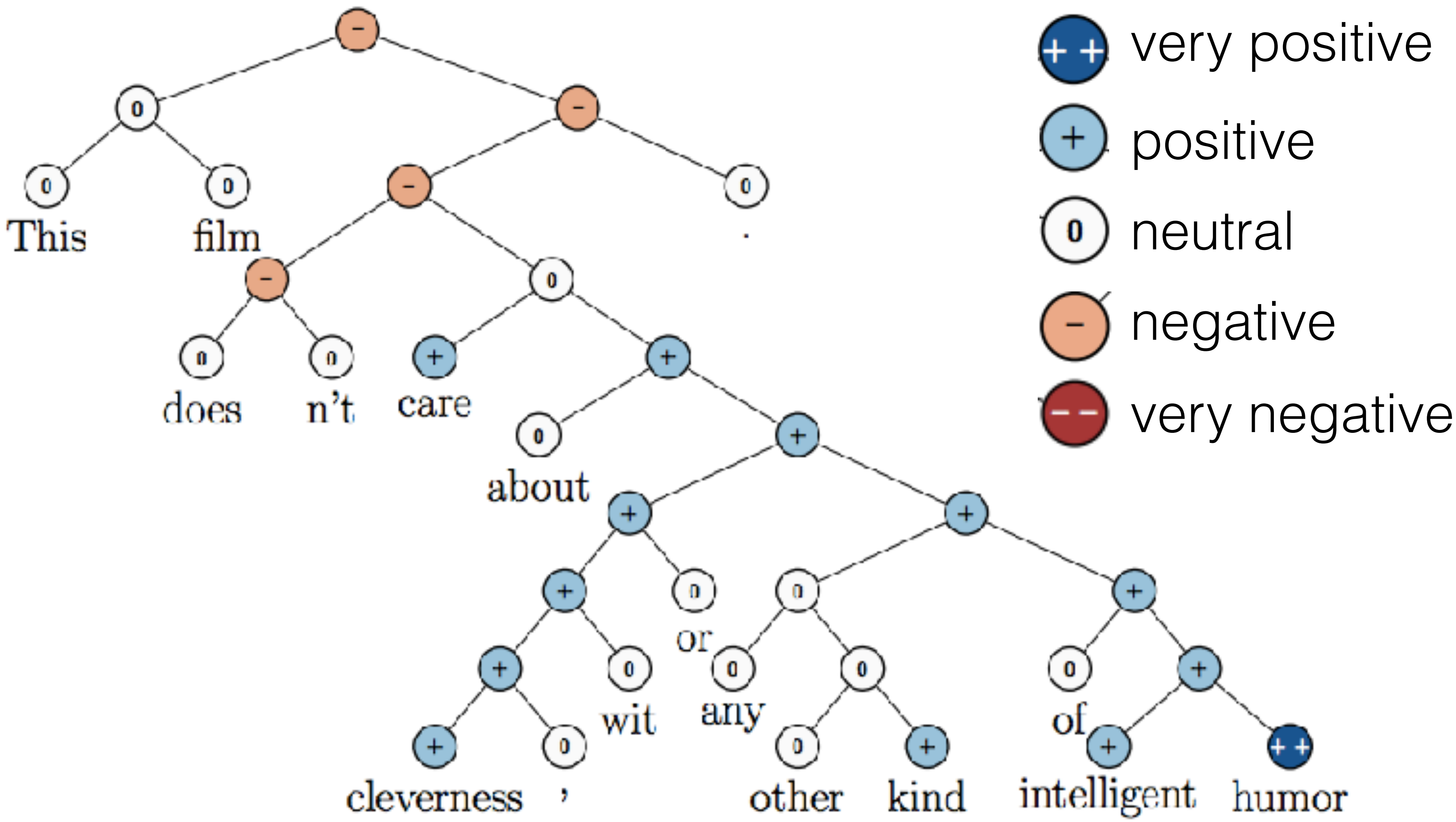


“Syntactic untying”

Representing Sentences

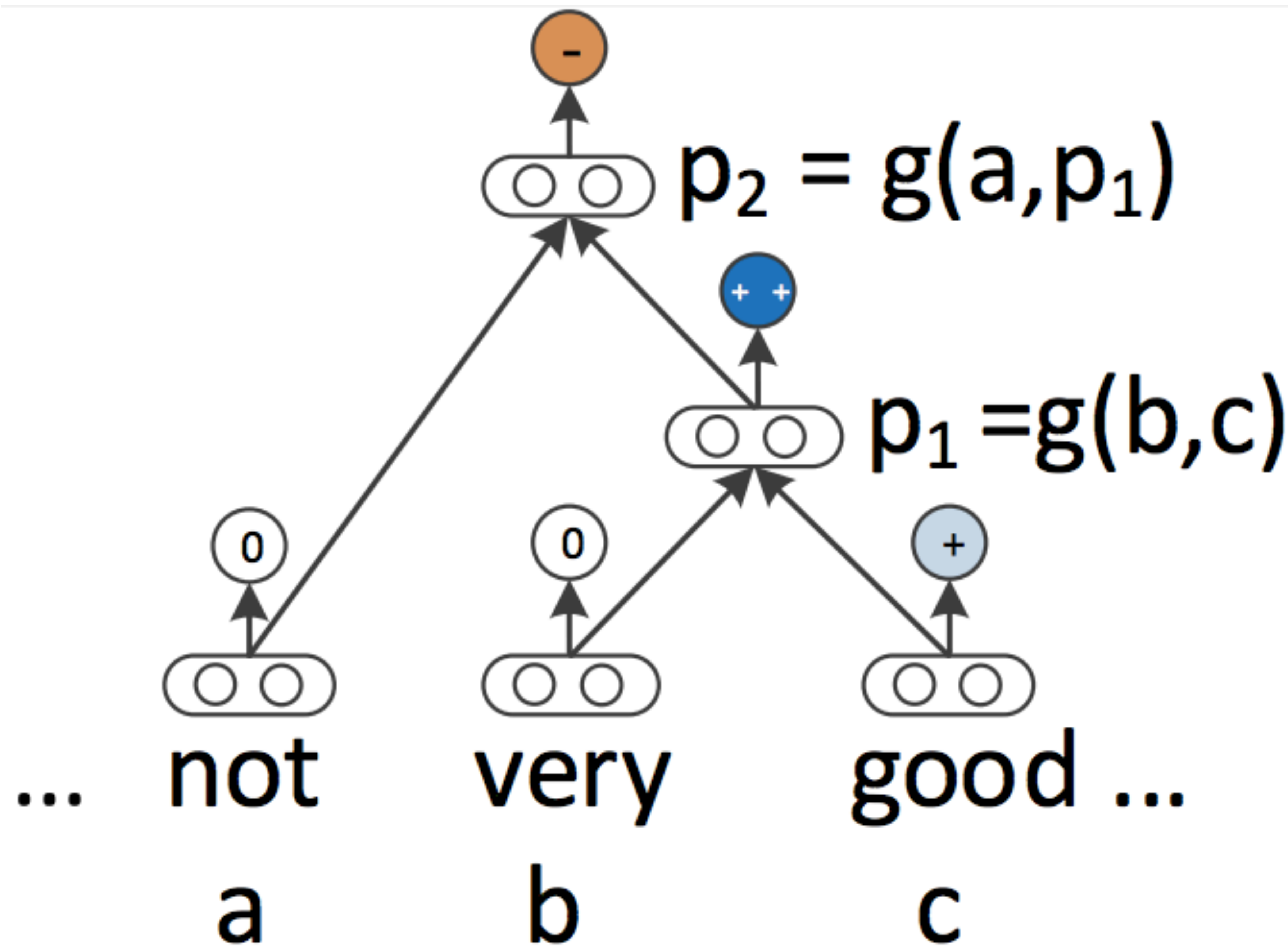
- Bag of words/n-grams
- Convolutional neural network
- Recurrent neural network
- Recursive neural network
- **In all of these, we can train by backpropagating through the "composition function"**

Stanford Sentiment Treebank



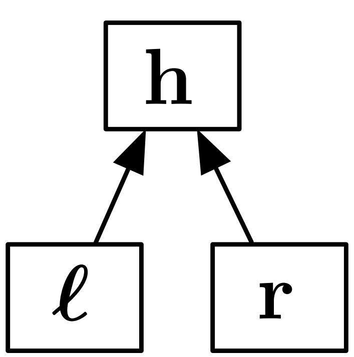
Socher et al. (2013, EMNLP)

Internal Supervision



Some Results

	all
Bigram Naive Bayes	83.1
RecNN (RNTN form)	**85.4**

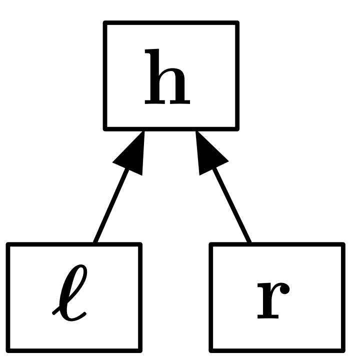


$$\mathbf{h} = \tanh\left(\mathbf{V}\mathrm{vec}([\boldsymbol{\ell};\mathbf{r}] \otimes [\boldsymbol{\ell};\mathbf{r}]) + \mathbf{W}[\boldsymbol{\ell};\mathbf{r}] + \mathbf{b}\right)$$

Outer product

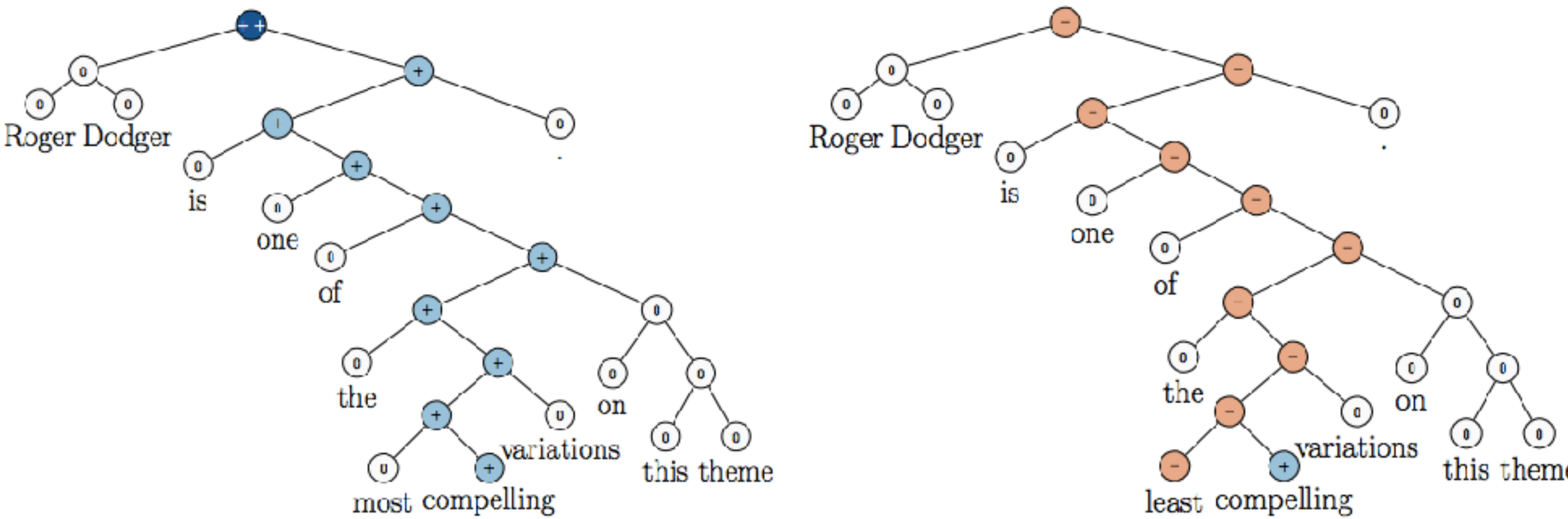
Some Results

	all	"not good"	"not terrible"
Bigram Naive Bayes	83.1	19.0	27.3
RecNN (RNTN form)	**85.4**	**71.4**	**81.8**



$$\mathbf{h} = \tanh\left(\mathbf{V}\,\mathrm{vec}([\boldsymbol{\ell};\mathbf{r}] \underbrace{\otimes}_{\text{Outer product}} [\boldsymbol{\ell};\mathbf{r}]) + \mathbf{W}[\boldsymbol{\ell};\mathbf{r}] + \mathbf{b}\right)$$

Some Predictions



Predictions by RNTN variant of RecNN.

Many Extensions

- Various cell definitions, e.g., (matrix, vector) pairs, higher order tensors
- Improved gradient dynamics using tree cells defined in terms of LSTM updates with gating instead of RNN. **Exercise**: *generalize the definition of a sequential LSTM to the tree case. Check the paper.*
- *n*-ary children
- "Inside outside" networks provide an analogue to bidirectional RNNs (lecture from a few weeks ago)
- Dependency syntax rather than "phrase structure" syntax
- Applications to programming languages, visual scene analysis—anywhere you can get trees, you can apply RecNNs

Recursive vs. Recurrent

- **Advantages**
  - Meaning decomposes roughly according to the syntax of a sentence (and we have good tools for obtaining syntax trees for sentences) — **better inductive bias**
  - Shorter gradient paths on average ($\log_2(n)$ in the best case)
  - Internal supervision of the node representations ("auxiliary objectives") is sometimes available
- **Disadvantages**
  - We need parse trees
  - Trees tend to be right-branching—gradients still have a long way to go!
  - More difficult to batch than RNNs

Topics

- Recursive neural networks for sentence representation
- Recurrent neural network grammars and parsing
- Word representations by looking inside words (words have structure too!)
- Analysis of neural networks with linguistic concepts

Where do trees come from?

An alternative to RNN LMs

Recurrent Neural Net Grammars

- Generate **symbols** sequentially using an **RNN**

An alternative to RNN LMs
Recurrent Neural Net Grammars

- Generate **symbols** sequentially using an **RNN**
- Add some **control symbols** to rewrite the history occasionally
  - Occasionally **compress** a sequence into a **constituent**
  - RNN predicts next terminal/control symbol based on the history of compressed elements and non-compressed terminals

An alternative to RNN LMs
Recurrent Neural Net Grammars

- Generate **symbols** sequentially using an **RNN**
- Add some **control symbols** to rewrite the history occasionally
  - Occasionally **compress** a sequence into a **constituent**
  - RNN predicts next terminal/control symbol based on the history of compressed elements and non-compressed terminals
- This is a **top-down, left-to-right generation** of a tree+sequence

Example derivation

The hungry cat meows loudly

stack	action	probability

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$

stack	action	probability
	NT(S**)**	$p(\text{NT}(\text{S}) \mid \text{TOP})$
(S		

stack	action	probability
	NT(S**)**	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT(**NP**)**	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$

stack	action	probability
	NT(S**)**	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT(**NP**)**	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP		

stack	action	probability
	NT(S**)**	$p(\textsc{nt}(\text{S}) \mid \textsc{top})$
(S	**NT(**NP**)**	$p(\textsc{nt}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN(***The***)**	$p(\textsc{gen}(\textit{The}) \mid (\text{S}, (\text{NP})$

stack	action	probability
	NT(S**)**	$p(\textsc{nt}(\text{S}) \mid \textsc{top})$
(S	**NT(**NP**)**	$p(\textsc{nt}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN(***The***)**	$p(\textsc{gen}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*		

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT(NP)**	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN(*The*)**	$p(\textsc{gen}(\mathit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN(*hungry*)**	$p(\textsc{gen}(\mathit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \mathit{The})$

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT(NP)**	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN(*The*)**	$p(\textsc{gen}(\mathit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN(*hungry*)**	$p(\textsc{gen}(\mathit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \mathit{The})$
(S (NP *The* *hungry*		

stack	action	probability
	NT(S)	$p(\textsc{nt}(\text{S}) \mid \textsc{top})$
(S	**NT**(NP)	$p(\textsc{nt}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN**(*The*)	$p(\textsc{gen}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\textsc{gen}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\textsc{gen}(\textit{cat}) \mid \ldots)$

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT**(NP)	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN**(*The*)	$p(\textsc{gen}(\mathit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\textsc{gen}(\mathit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \mathit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\textsc{gen}(\mathit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*		

stack	action	probability
	NT(S)	$p(\text{NT}(\text{S}) \mid \text{TOP})$
(S	NT(NP)	$p(\text{NT}(\text{NP}) \mid (\text{S})$
(S (NP	GEN(*The*)	$p(\text{GEN}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	GEN(*hungry*)	$p(\text{GEN}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	GEN(*cat*)	$p(\text{GEN}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	REDUCE	$p(\text{REDUCE} \mid \ldots)$

stack	action	probability
	NT(S)	$p(\text{NT}(\text{S}) \mid \text{TOP})$
(S	**NT**(NP)	$p(\text{NT}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN**(*The*)	$p(\text{GEN}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\text{GEN}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\text{GEN}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\text{REDUCE} \mid \ldots)$
(S (NP *The* *hungry* *cat*)		

stack	action	probability
	NT(S)	$p(\textsc{nt}(\text{S}) \mid \textsc{top})$
(S	NT(NP)	$p(\textsc{nt}(\text{NP}) \mid (\text{S})$
(S (NP	GEN(*The*)	$p(\textsc{gen}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	GEN(*hungry*)	$p(\textsc{gen}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	GEN(*cat*)	$p(\textsc{gen}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	REDUCE	$p(\textsc{reduce} \mid \ldots)$
(S (NP *The* *hungry* *cat*)		
(S (NP *The hungry cat*)		

Compress "The hungry cat" into a single composite symbol

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT**(NP)	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN**(*The*)	$p(\textsc{gen}(\mathit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\textsc{gen}(\mathit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \mathit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\textsc{gen}(\mathit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\textsc{reduce} \mid \ldots)$
(S (NP *The hungry cat*)		

stack	action	probability
	NT(S)	$p(\textsc{nt}(\text{S}) \mid \textsc{top})$
(S	**NT(NP)**	$p(\textsc{nt}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN(*The*)**	$p(\textsc{gen}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	**GEN(*hungry*)**	$p(\textsc{gen}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	**GEN(*cat*)**	$p(\textsc{gen}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\textsc{reduce} \mid \ldots)$
(S (NP *The hungry cat*)	**NT(VP)**	$p(\textsc{nt}(\text{VP}) \mid (\text{S}, (\text{NP} \;\; \textit{The hungry cat})$

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT**(NP)	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN**(*The*)	$p(\textsc{gen}(\mathit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\textsc{gen}(\mathit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \mathit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\textsc{gen}(\mathit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\textsc{reduce} \mid \ldots)$
(S (NP *The hungry cat*)	**NT**(VP)	$p(\textsc{nt}(\mathrm{VP}) \mid (\mathrm{S}, (\mathrm{NP}\ \mathit{The\ hungry\ cat}))$
(S (NP *The hungry cat*) (VP		

stack	action	probability
	NT(S)	$p(\textsc{nt}(\mathrm{S}) \mid \textsc{top})$
(S	**NT**(NP)	$p(\textsc{nt}(\mathrm{NP}) \mid (\mathrm{S})$
(S (NP	**GEN**(*The*)	$p(\textsc{gen}(\textit{The}) \mid (\mathrm{S}, (\mathrm{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\textsc{gen}(\textit{hungry}) \mid (\mathrm{S}, (\mathrm{NP}, \textit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\textsc{gen}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\textsc{reduce} \mid \ldots)$
(S (NP *The hungry cat*)	**NT**(VP)	$p(\textsc{nt}(\mathrm{VP}) \mid (\mathrm{S}, (\mathrm{NP}\ \textit{The hungry cat})$
(S (NP *The hungry cat*) (VP	**GEN**(*meows*)	

stack	action	probability
	NT(S)	$p(\text{NT}(\text{S}) \mid \text{TOP})$
(S	**NT**(NP)	$p(\text{NT}(\text{NP}) \mid (\text{S})$
(S (NP	**GEN**(*The*)	$p(\text{GEN}(\textit{The}) \mid (\text{S}, (\text{NP})$
(S (NP *The*	**GEN**(*hungry*)	$p(\text{GEN}(\textit{hungry}) \mid (\text{S}, (\text{NP}, \textit{The})$
(S (NP *The* *hungry*	**GEN**(*cat*)	$p(\text{GEN}(\textit{cat}) \mid \ldots)$
(S (NP *The* *hungry* *cat*	**REDUCE**	$p(\text{REDUCE} \mid \ldots)$
(S (NP *The hungry cat*)	**NT**(VP)	$p(\text{NT}(\text{VP}) \mid (\text{S}, (\text{NP}\ \textit{The hungry cat})$
(S (NP *The hungry cat*) (VP	**GEN**(*meows*)	
(S (NP *The hungry cat*) (VP *meows*	**REDUCE**	
(S (NP *The hungry cat*) (VP *meows*)	**GEN**(.)	
(S (NP *The hungry cat*) (VP *meows*) .	**REDUCE**	
(S (NP *The hungry cat*) (VP *meows*) .)		

Some things you can (easily) prove

- Valid (tree, string) pairs are in bijection to valid sequences of actions (specifically, the DFS, left-to-right traversal of the trees)

- Every stack configuration perfectly encodes the complete history of actions.

- Therefore, the probability decomposition is justified by the chain rule, i.e.

$$
\begin{aligned}
p(\boldsymbol{x}, \boldsymbol{y}) &= p(actions(\boldsymbol{x}, \boldsymbol{y})) && \text{(prop 1)} \\
p(actions(\boldsymbol{x}, \boldsymbol{y})) &= \prod_i p(a_i \mid \boldsymbol{a}_{<i}) && \text{(chain rule)} \\
&= \prod_i p(a_i \mid stack(\boldsymbol{a}_{<i})) && \text{(prop 2)}
\end{aligned}
$$

Modeling the next action

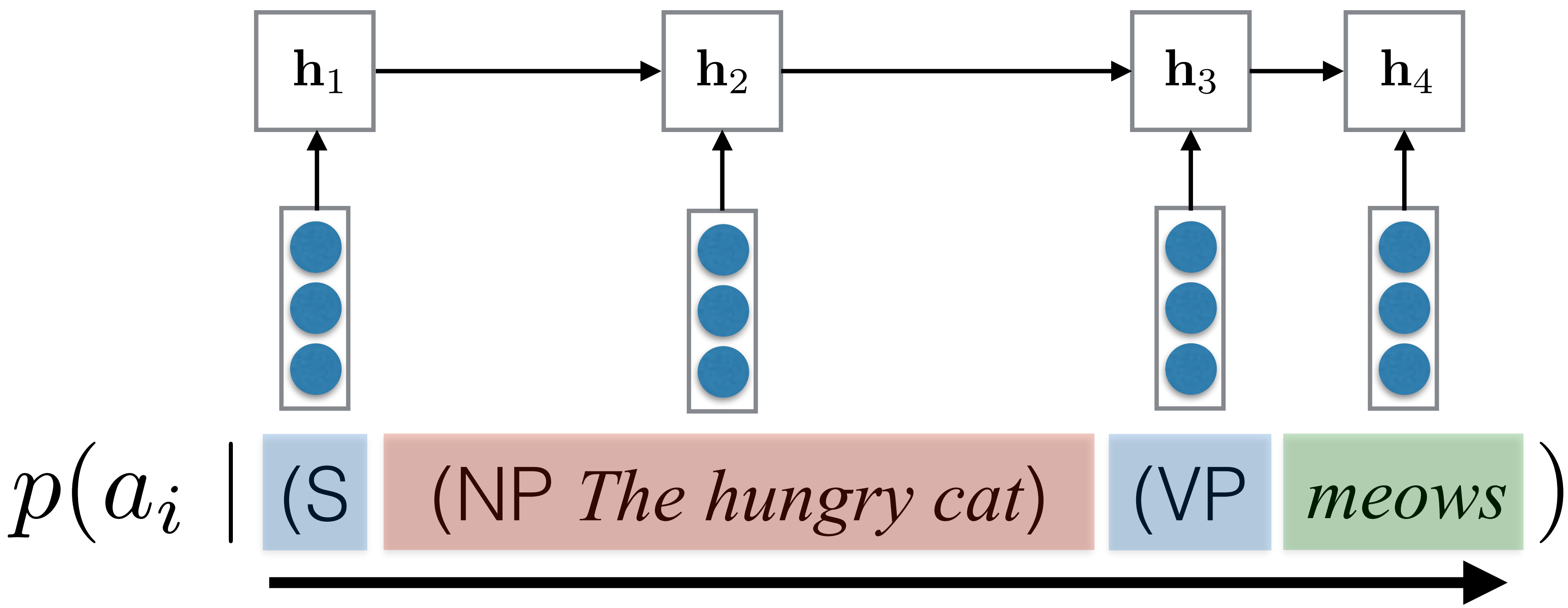
$$p(a_i \mid \text{(S} \;\; \text{(NP } \textit{The hungry cat}\text{)} \;\; \text{(VP} \;\; \textit{meows}\;)$$

Modeling the next action

$p(a_i \mid$ (S (NP *The hungry cat*) (VP *meows* $)$

1. unbounded depth

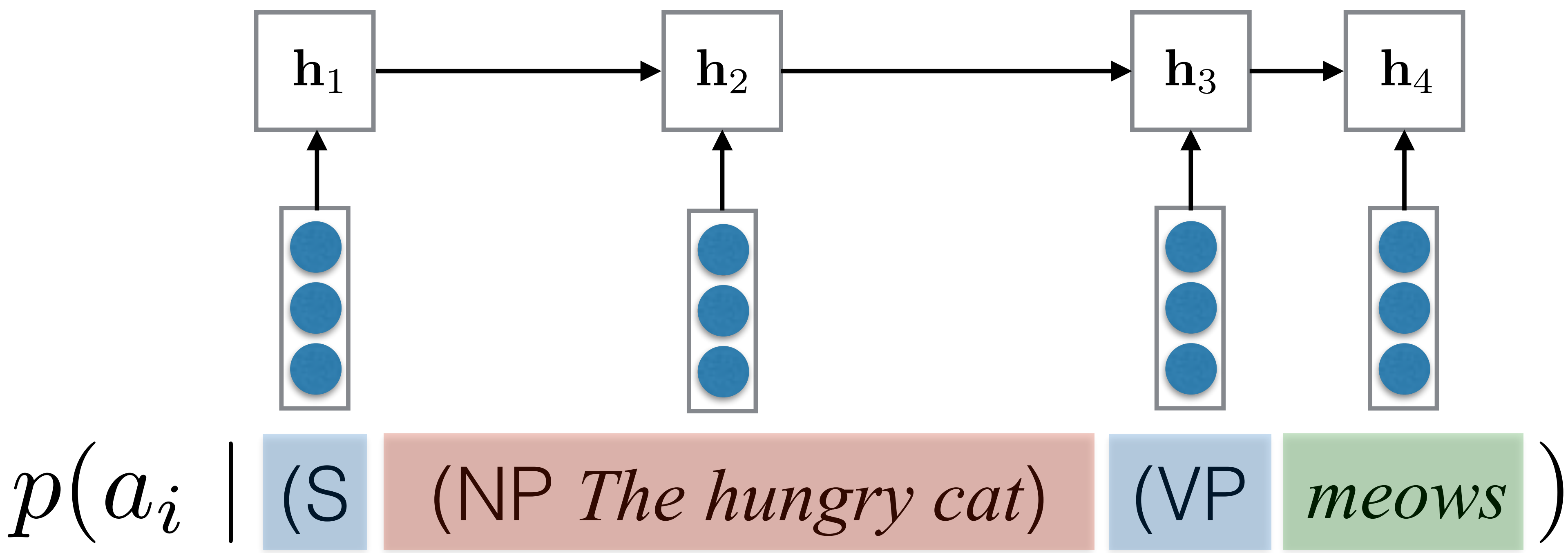
Modeling the next action



1. unbounded depth

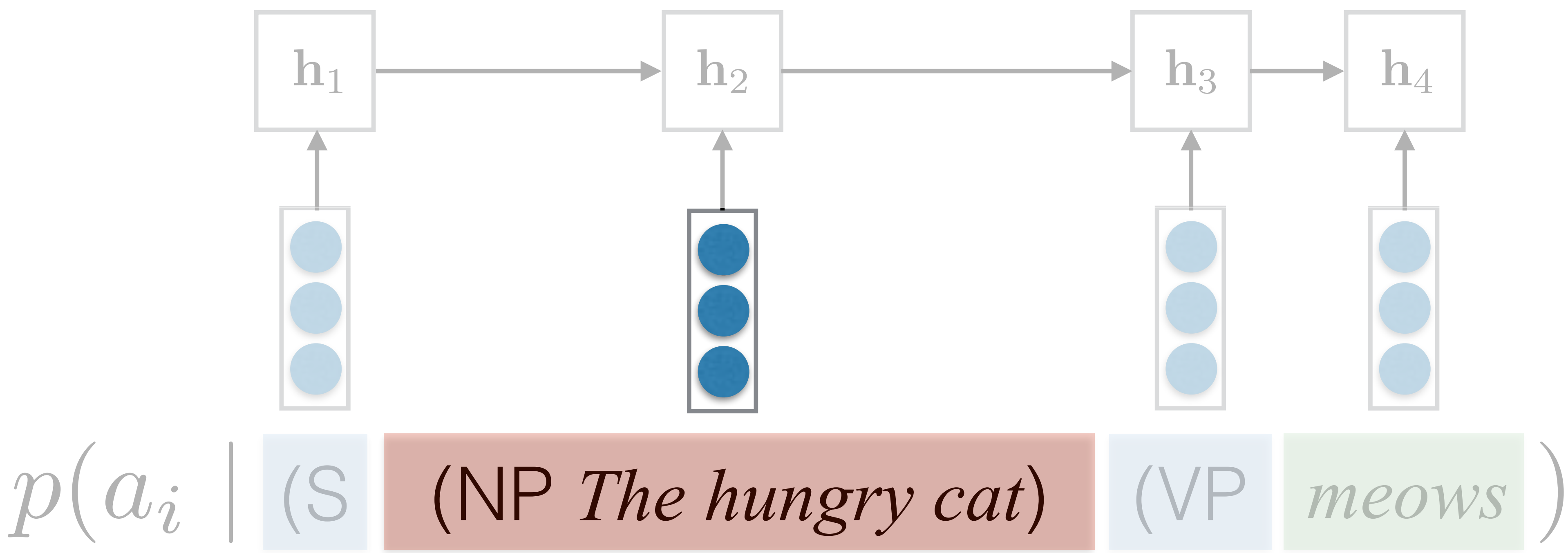
1. Unbounded depth → recurrent neural nets

Modeling the next action



1. Unbounded depth → recurrent neural nets

Modeling the next action



2. arbitrarily complex trees

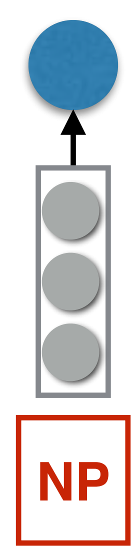
1. Unbounded depth → recurrent neural nets
2. Arbitrarily complex trees → recursive neural nets

Syntactic composition

Need representation for: (NP *The hungry cat*)

Syntactic composition

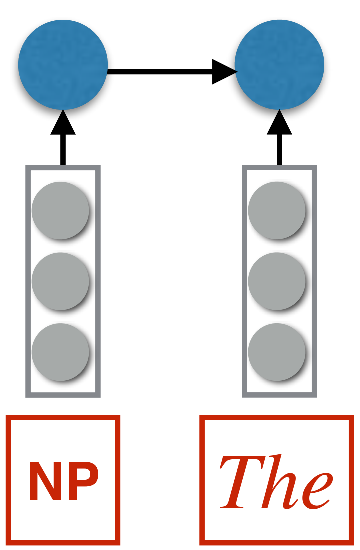
Need representation for: (NP *The hungry cat*)



What head type?

Syntactic composition

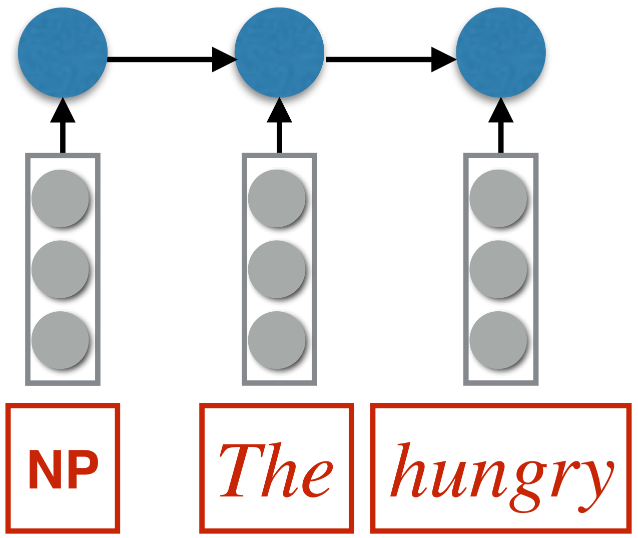
Need representation for: (NP *The hungry cat*)



What head type?

Syntactic composition

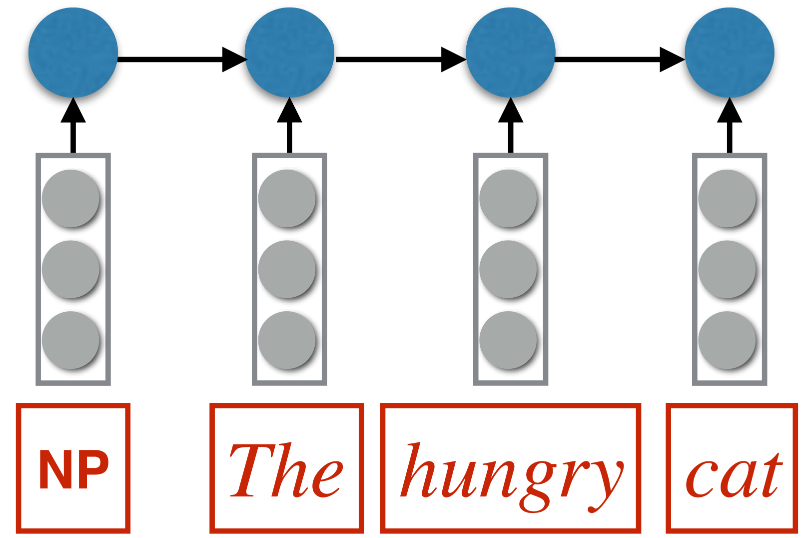
Need representation for: (NP *The hungry cat*)



What head type?

Syntactic composition

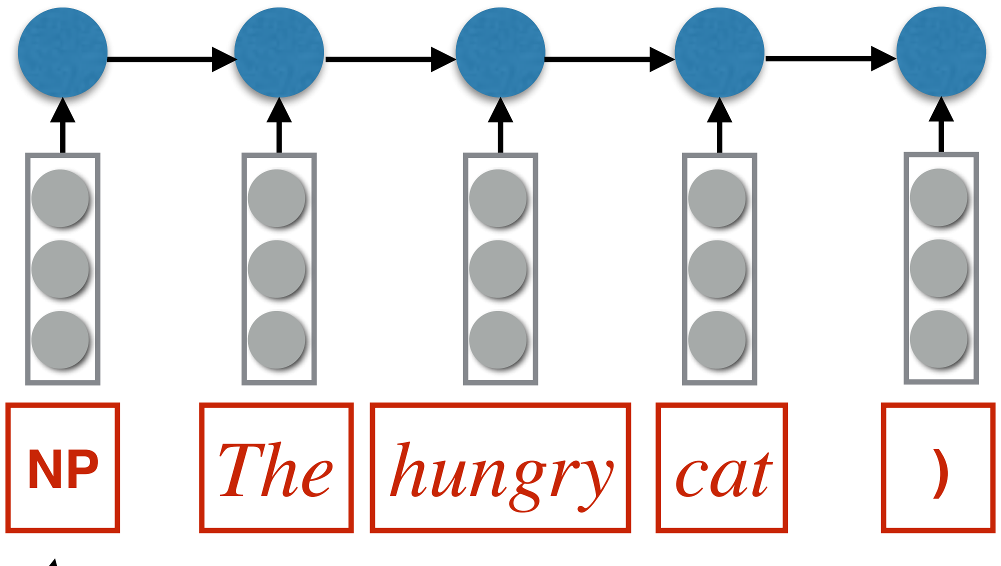
Need representation for: (NP *The hungry cat*)



What head type?

Syntactic composition

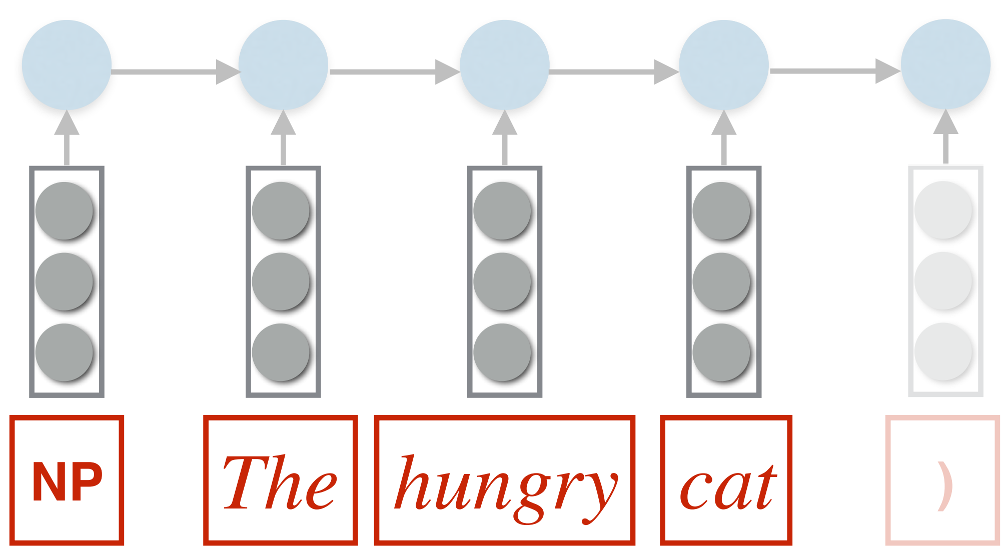
Need representation for: (NP *The hungry cat*)



What head type?

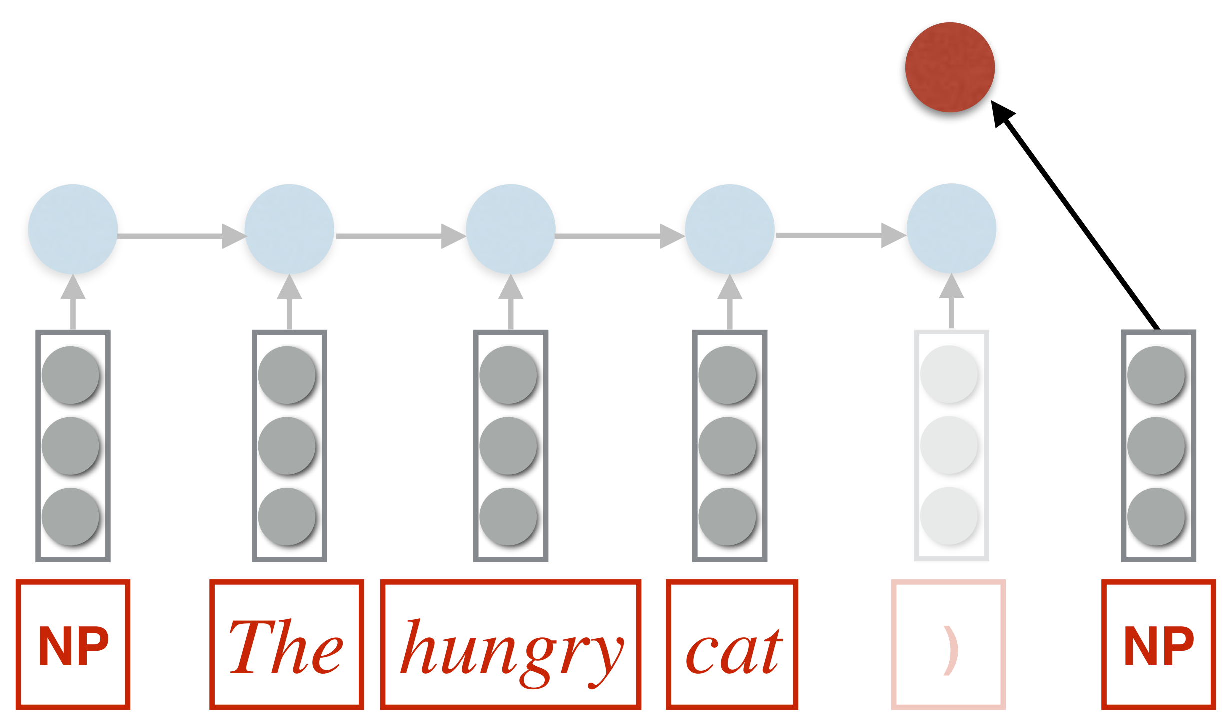
Syntactic composition

Need representation for: (NP *The hungry cat*)



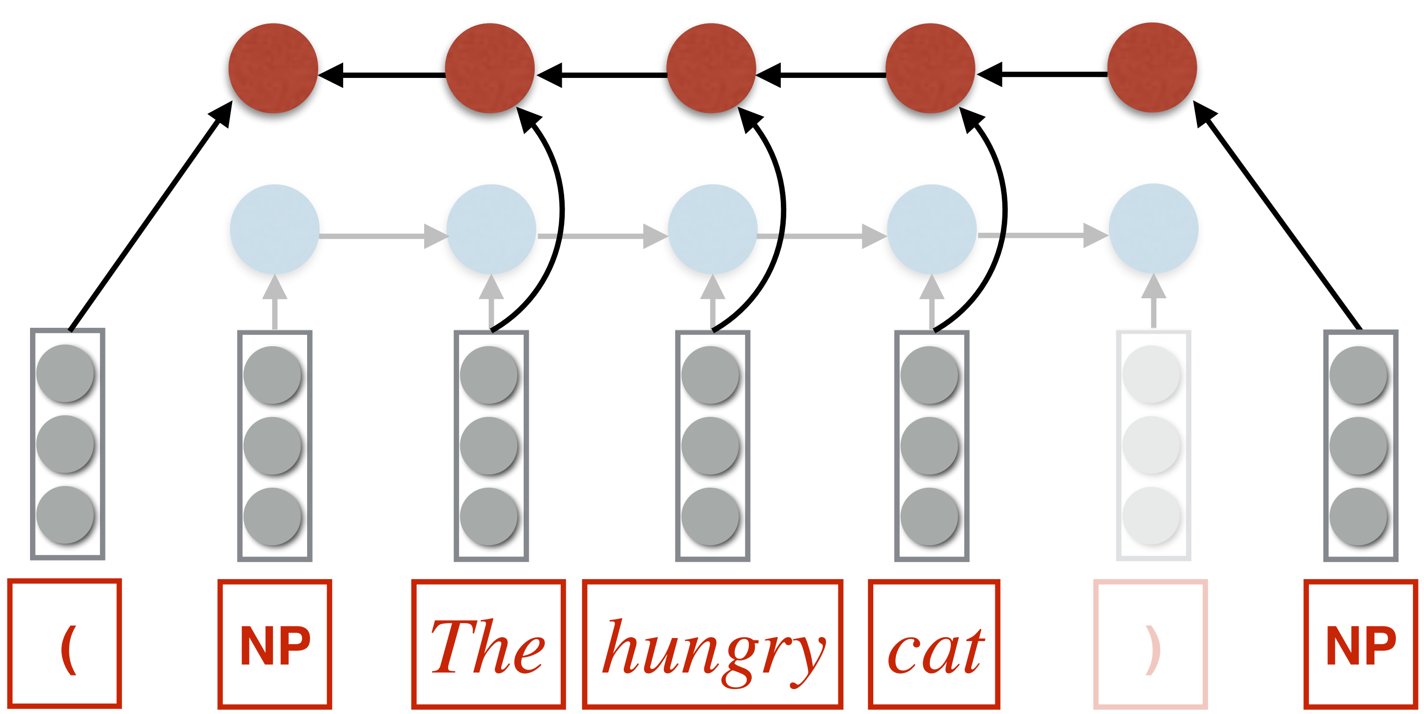
Syntactic composition

Need representation for: (NP *The hungry cat*)



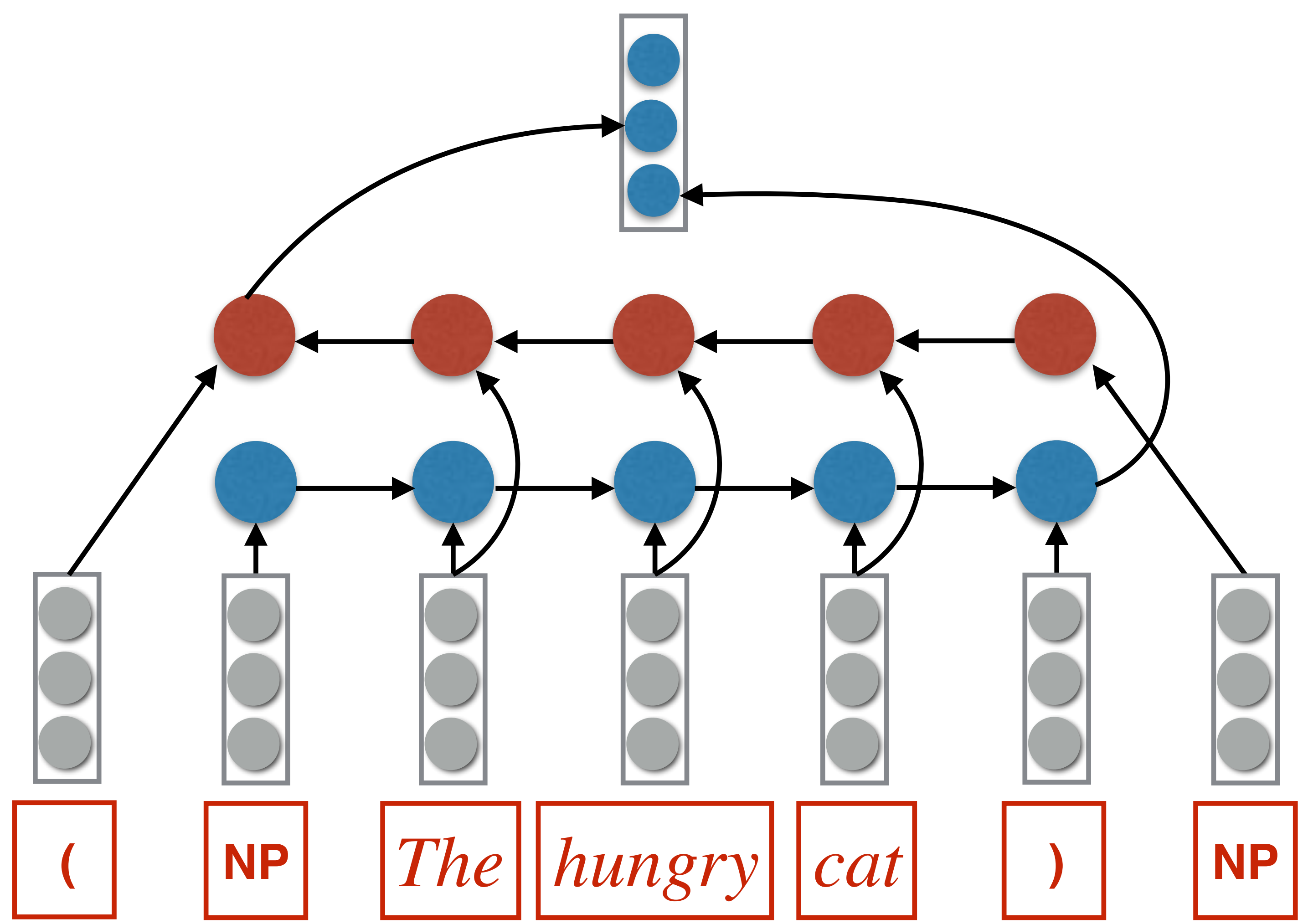
Syntactic composition

Need representation for: (NP *The hungry cat*)



Syntactic composition

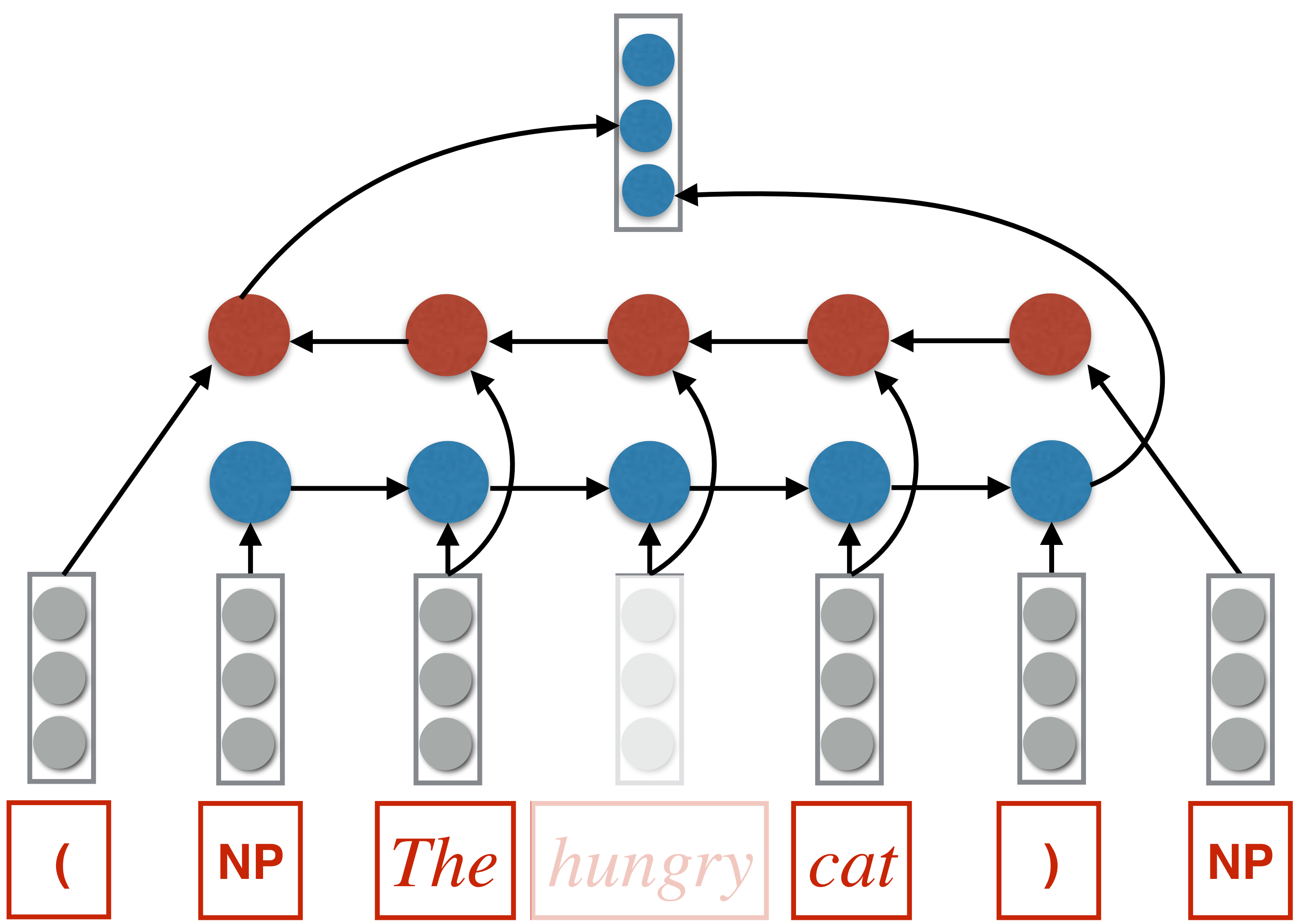
Need representation for: (NP *The hungry cat*)



Syntactic composition
Recursion

Need representation for: (NP *The hungry cat*)

(NP *The* (ADJP *very hungry*) *cat*)

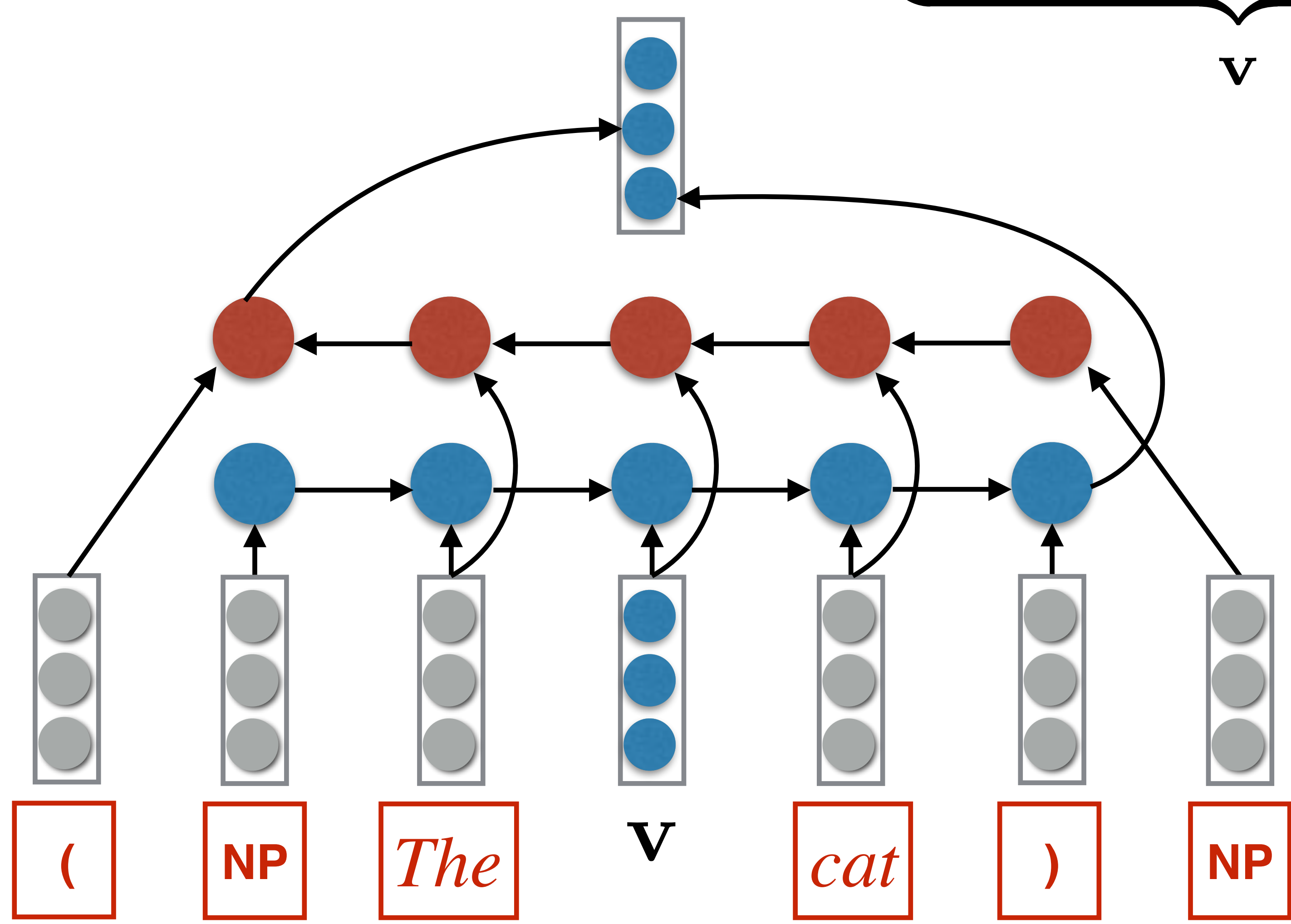


Syntactic composition
Recursion

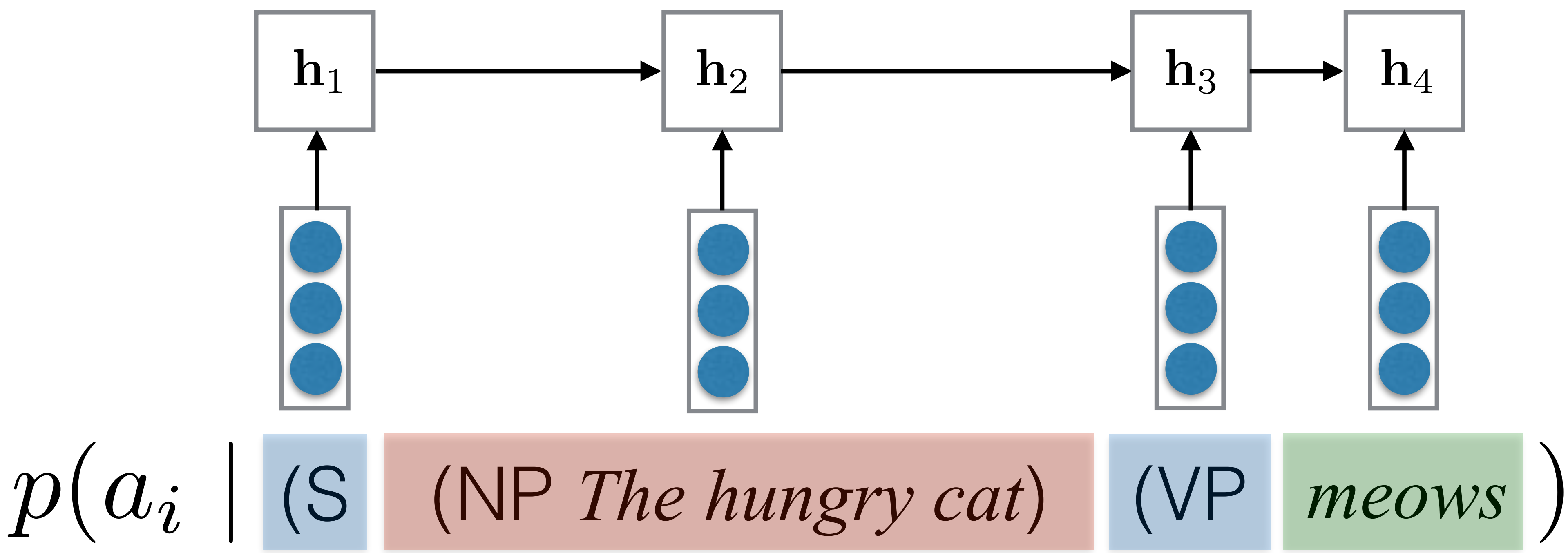
Need representation for: (NP *The hungry cat*)

(NP *The* (ADJP *very hungry*) *cat*)

$\underbrace{\text{(ADJP } \textit{very hungry}\text{)}}_{\mathbf{v}}$

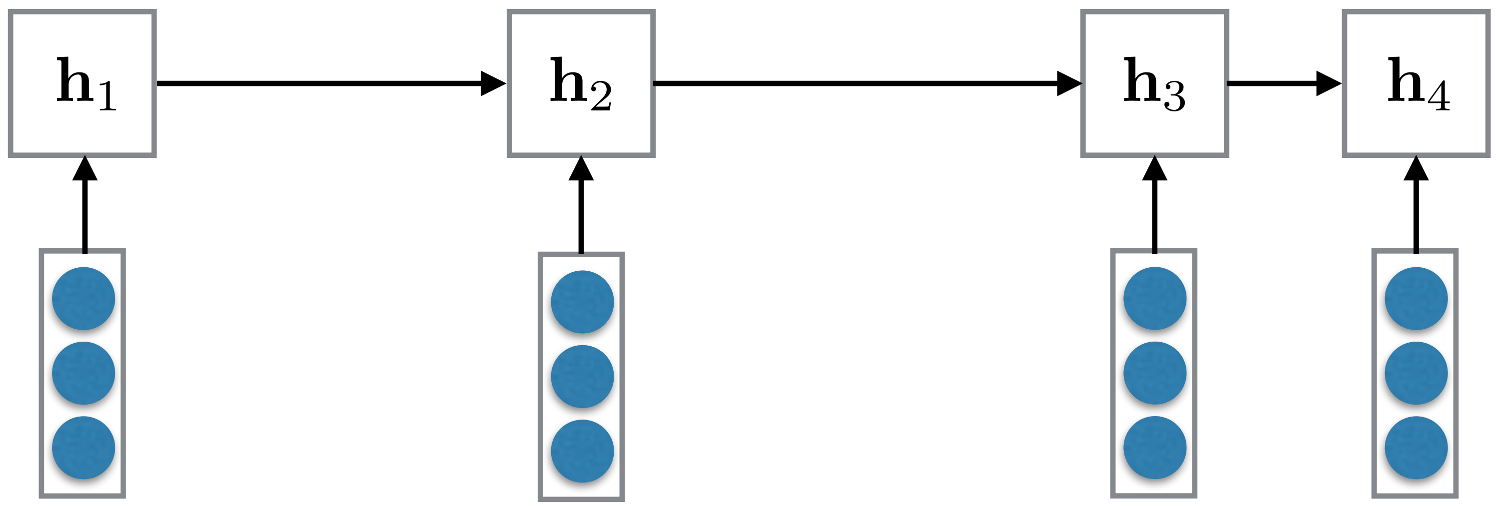


Modeling the next action



1. Unbounded depth → recurrent neural nets
2. Arbitrarily complex trees → recursive neural nets

Modeling the next action



$p(a_i \mid$ (S (NP *The hungry cat*) (VP *meows* $) \sim$ **REDUCE**

1. Unbounded depth → recurrent neural nets
2. Arbitrarily complex trees → recursive neural nets

Modeling the next action

$$p(a_i \mid \text{(S (NP } \textit{The hungry cat}\text{) (VP } \textit{meows}) \sim \textbf{REDUCE}$$

$$p(a_{i+1} \mid \text{(S (NP } \textit{The hungry cat}\text{) (VP } \textit{meows}\text{)})$$

1. Unbounded depth → recurrent neural nets
2. Arbitrarily complex trees → recursive neural nets

Modeling the next action

$$p(a_i \mid \text{(S} \;\; \text{(NP } \textit{The hungry cat}\text{)} \;\; \text{(VP} \;\; \textit{meows} \;) \sim \textbf{REDUCE}$$

$$p(a_{i+1} \mid \text{(S} \;\; \text{(NP } \textit{The hungry cat}\text{)} \;\; \text{(VP } \textit{meows}\text{)} \;)$$

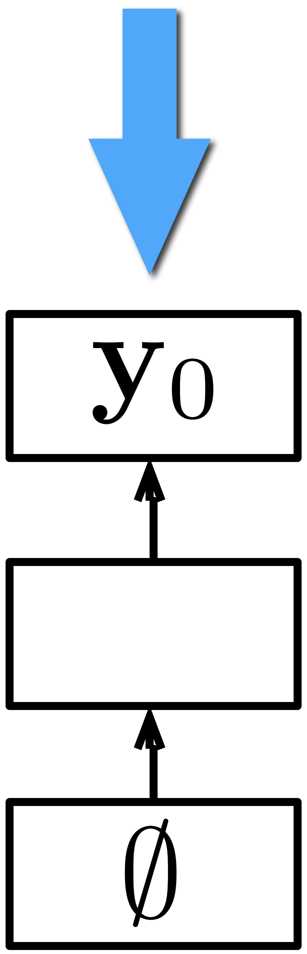
3. limited updates

1. Unbounded depth → recurrent neural nets
2. Arbitrarily complex trees → recursive neural nets
3. Limited updates to state → stack RNNs

Stack RNNs
Operation

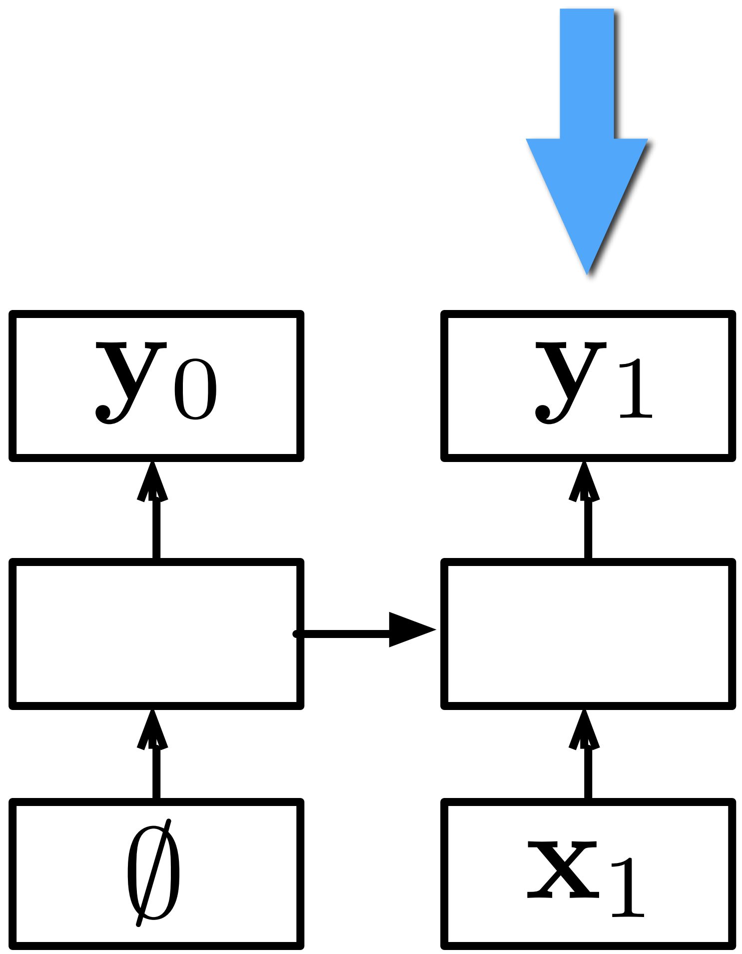
- Augment RNN with a **stack pointer**
- Two constant-time operations
  - **Push** - read input, add to top of stack
  - **Pop** - move stack pointer back
- A **summary** of stack contents is obtained by accessing the output of the RNN at location of the stack pointer

Stack RNNs
Operation



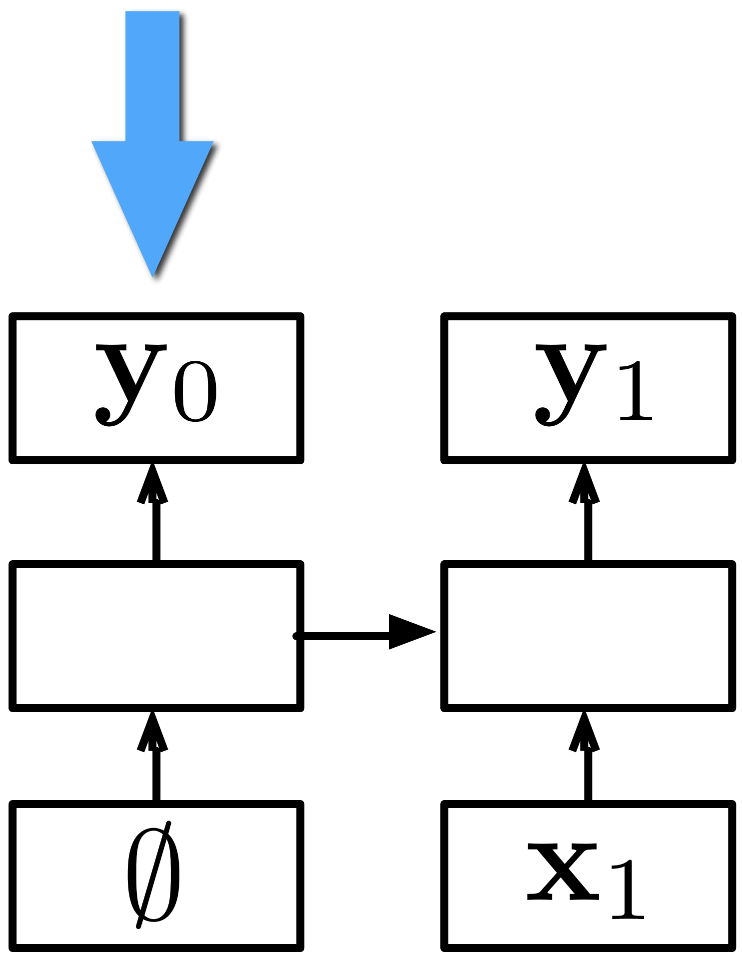
PUSH

Stack RNNs
Operation

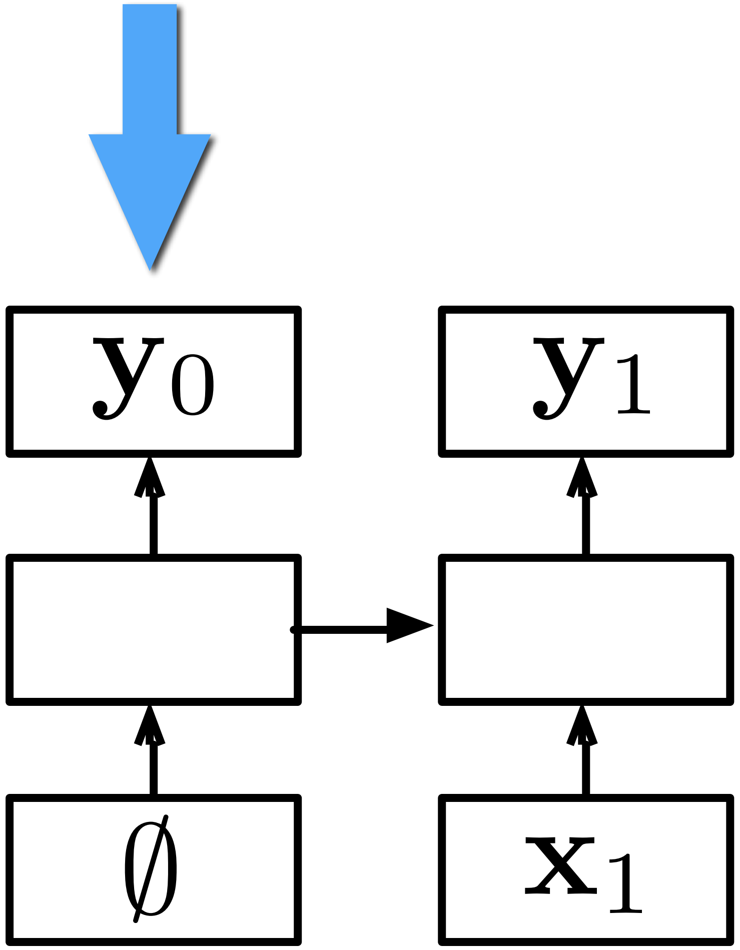


POP

Stack RNNs
Operation

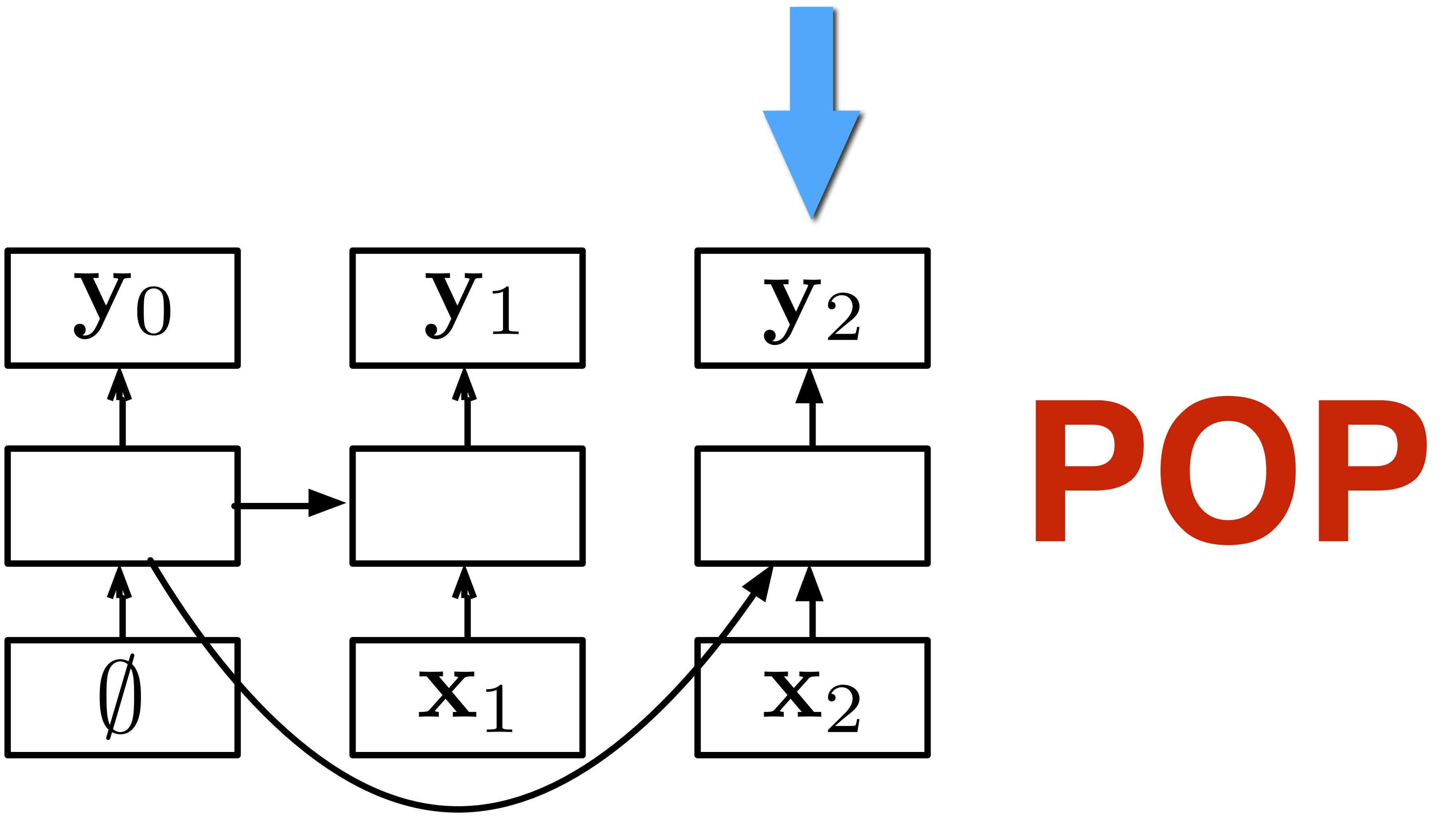


Stack RNNs
Operation

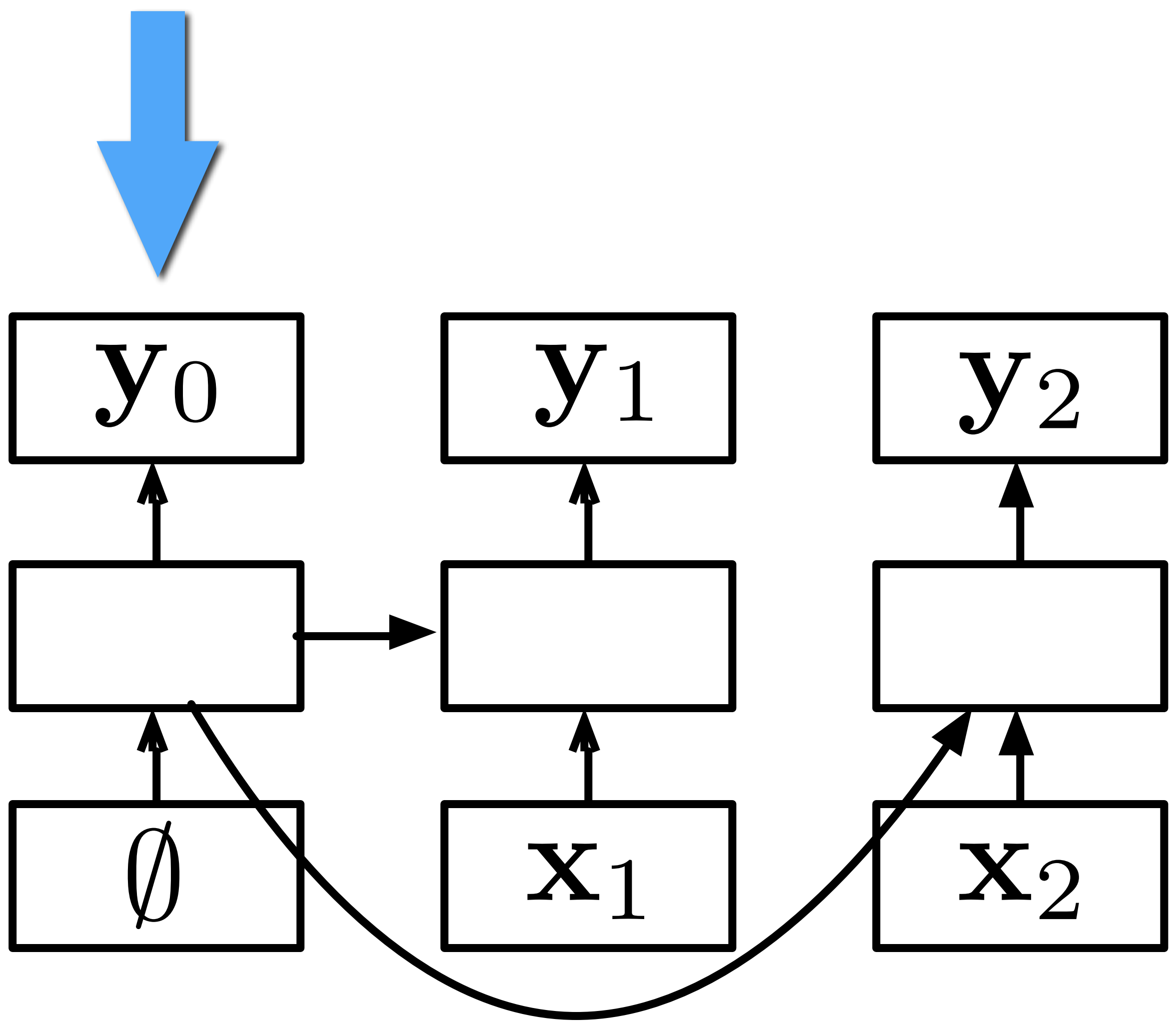


PUSH

Stack RNNs
Operation

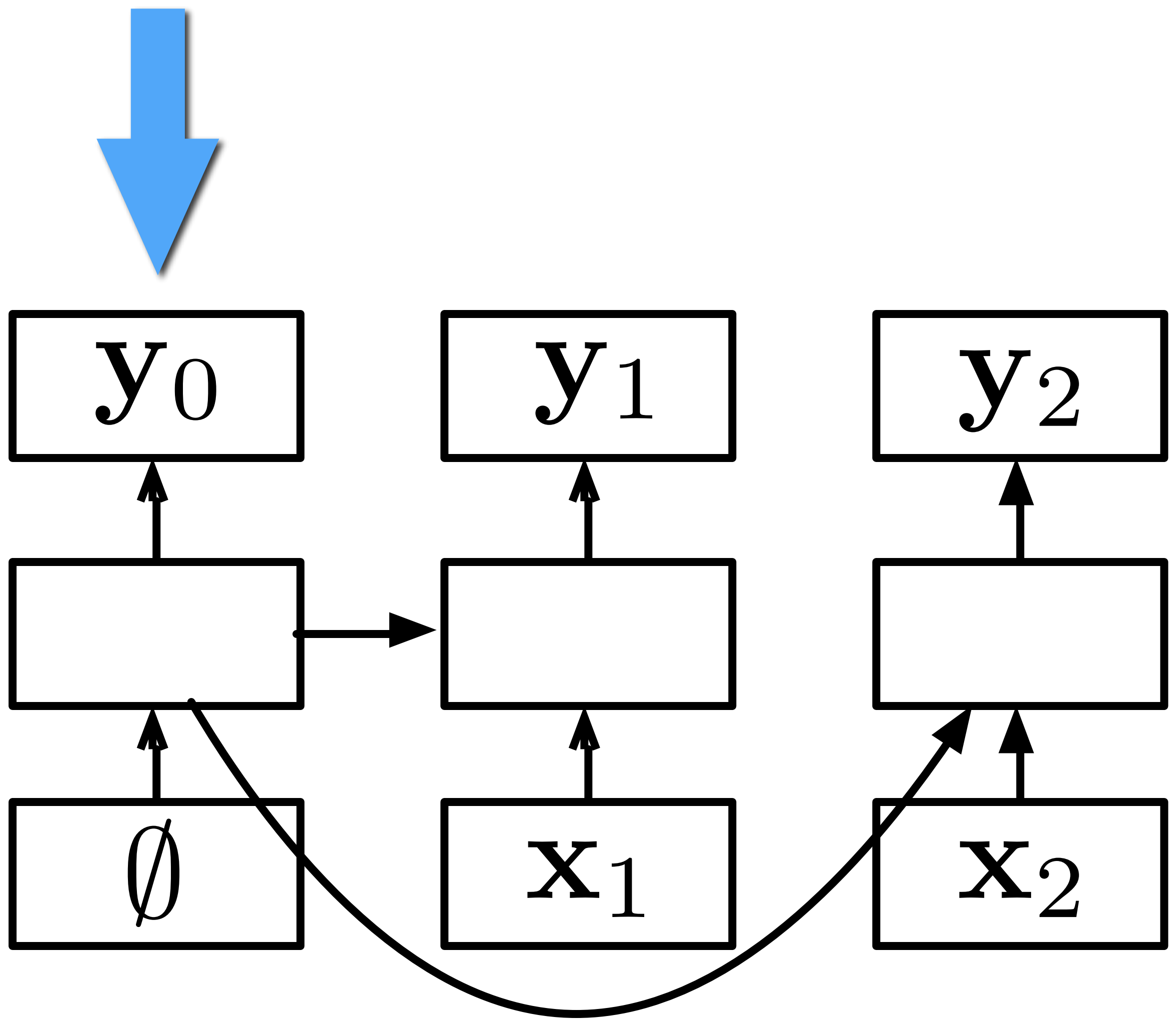


Stack RNNs
Operation



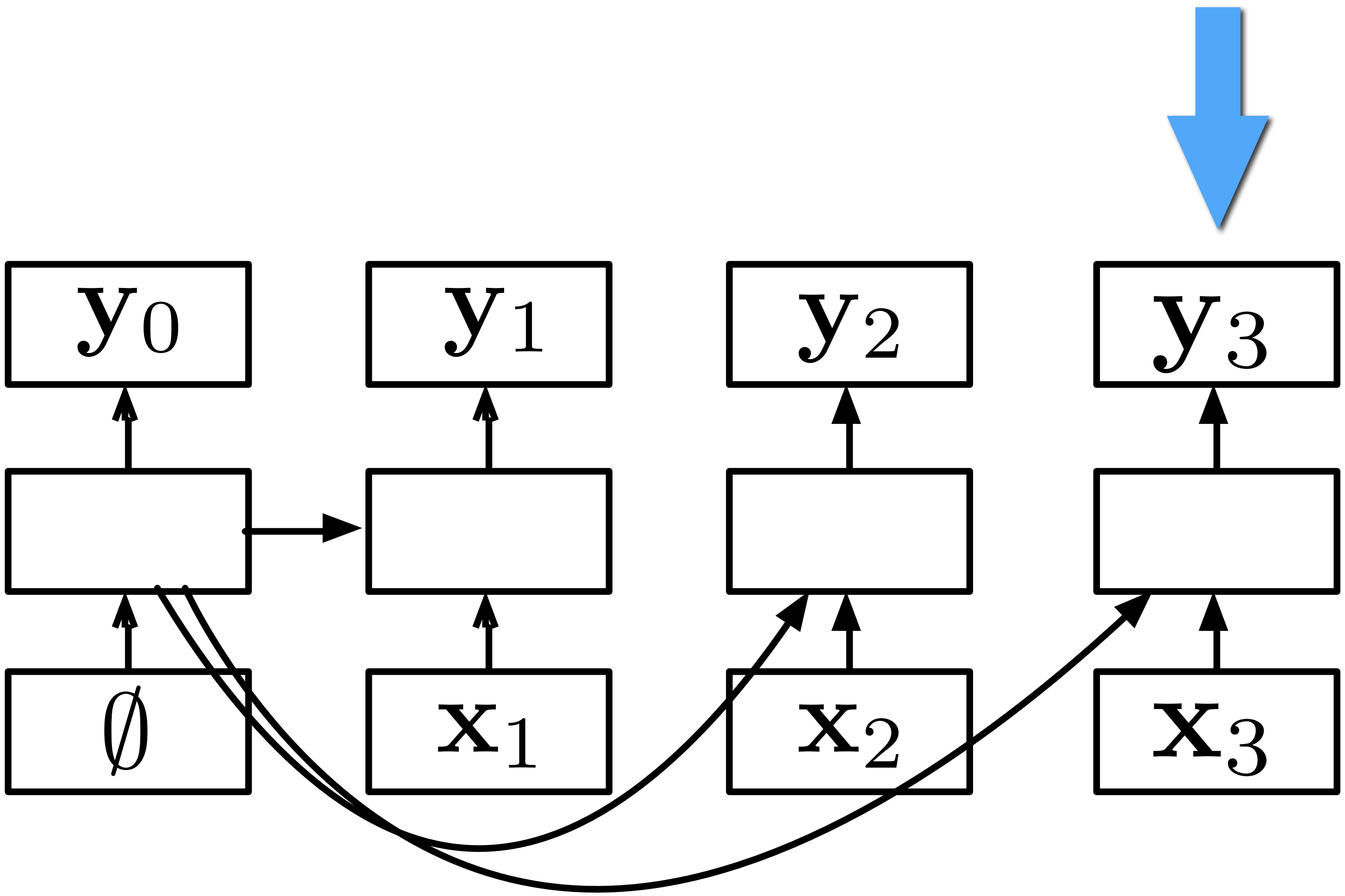
Stack RNNs

Operation



PUSH

Stack RNNs
Operation



RNNGs
Inductive bias?

- What inductive biases do RNNGs exhibit?
- If we accept the following two propositions
  - RNNs have recency biases
  - Syntactic composition learns to represent trees by their heads
- Then we can say that they have a bias for **syntactic recency** rather than **sequential recency**
- Not a perfect model, but maybe a **better** model

“talk”

(S (NP *The talk* (SBAR *I did* ***not*** *give*)) (VP *appealed* (PP *to* …

Parameter estimation

- **Generative**
  - Jointly model sentence **x** and its tree **y**
  - Trained using gold standard trees (here: from a tree bank) to minimize cross-entropy
  - We call this joint distribution *p*(**x**,**y**)

To parse (find a tree for ***x***): we need to compute

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} p(\boldsymbol{y} \mid \boldsymbol{x})$$

$$= \arg\max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})}$$ def. conditional prob.

$$= \arg\max_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} p(\boldsymbol{x}, \boldsymbol{y})$$ denominator is constant

Parameter estimation

- **Generative**
  - Jointly model sentence **x** and its tree **y**
  - Trained using gold standard trees (here: from a tree bank) to minimize cross-entropy
  - We call this joint distribution $p(\mathbf{x},\mathbf{y})$
- **Discriminative**
  - Given a sentence **x**, predict the sequence to of actions **y** necessary to build its parse tree - *the full sentence* ***x*** *is observable*
  - Instead of **GEN**, use **SHIFT**
  - We call this conditional distribution $q(\mathbf{y} \mid \mathbf{x})$

To parse: simply use beam search to find the best sequence.

English PTB (Parsing)

	Type	F1
Petrov and Klein (2007)	Gen	90.1
Shindo et al (2012) Single model	Gen	91.1
Vinyals et al (2015) PTB only	Disc	90.5
Shindo et al (2012) Ensemble	Gen	92.4
Vinyals et al (2015) Semisupervised	Disc+SemiSup	92.8
Discriminative **PTB only**	Disc	91.7
Generative **PTB only**	Gen	**93.6**

English PTB (Parsing)

	Type	F1
Petrov and Klein (2007)	Gen	90.1
Shindo et al (2012) Single model	Gen	91.1
Vinyals et al (2015) PTB only	Disc	90.5
Shindo et al (2012) Ensemble	Gen	92.4
Vinyals et al (2015) Semisupervised	Disc+SemiSup	92.8
***Discriminative* PTB only**	Disc	91.7
***Generative* PTB only**	Gen	**93.6**

English PTB (Parsing)

	Type	F1
Petrov and Klein (2007)	Gen	90.1
Shindo et al (2012) Single model	Gen	91.1
Vinyals et al (2015) PTB only	Disc	90.5
Shindo et al (2012) Ensemble	Gen	92.4
Vinyals et al (2015) Semisupervised	Disc+SemiS up	92.8
***Discriminative* PTB only**	Disc	91.7
***Generative* PTB only**	Gen	**93.6**

English Language Modeling

	Perplexity
5-gram IKN	169.3
LSTM + Dropout	113.4
Generative (approx.)	**102.4**

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{x}}} p(\boldsymbol{x}, \boldsymbol{y})$$

Transition-based parsing

- Build trees by pushing words ("**shift**") onto a stack and combing elements at the top of the stack into a syntactic constituent ("**reduce**")

- ***Given current stack and buffer of unprocessed words, what action should the algorithm take?***

- Widely used

    - Good accuracy

    - $O(n)$ runtime [much faster than other parsing algos]

Stack	Buffer	Action
	I saw her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw		

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT

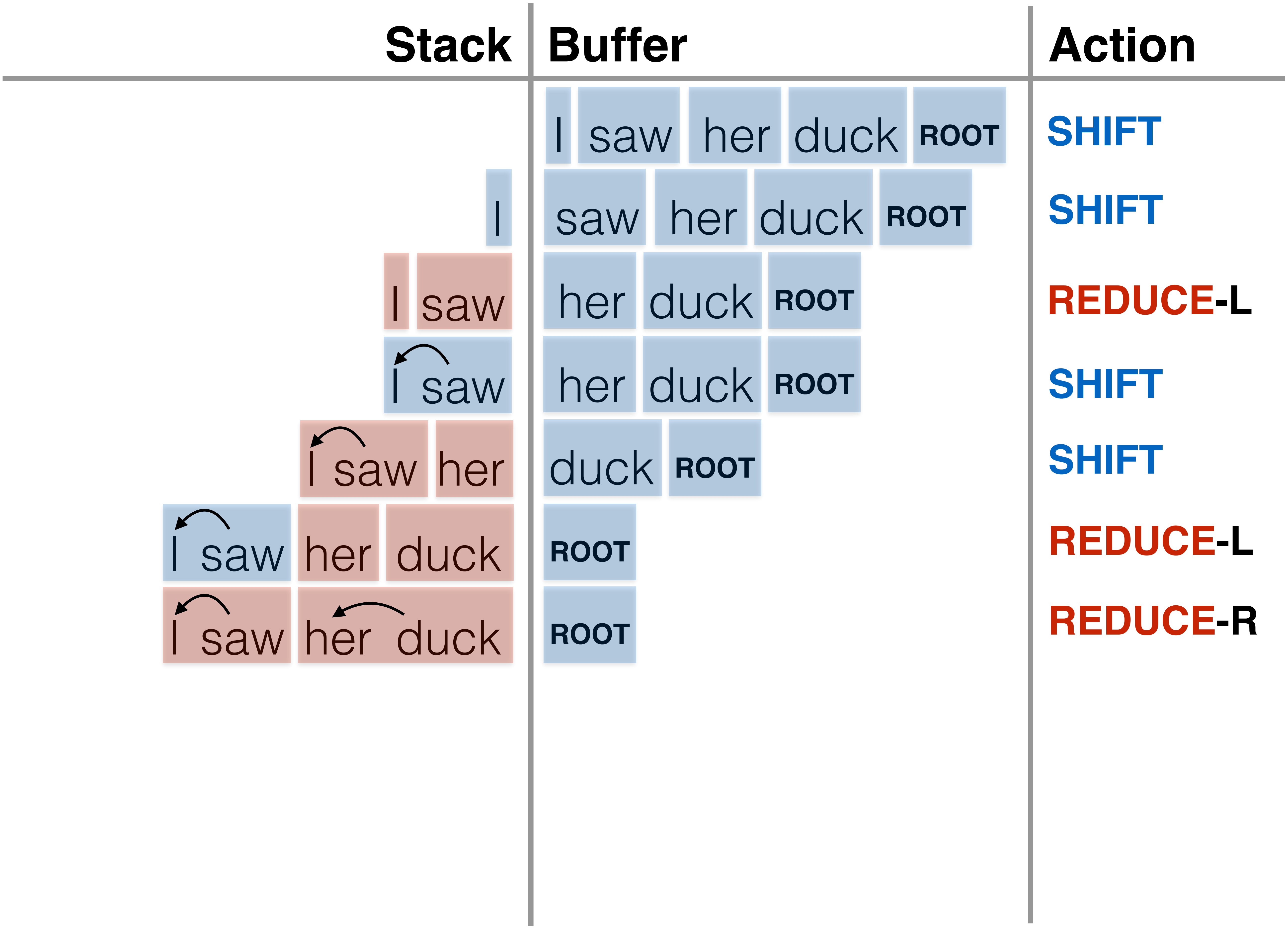
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L

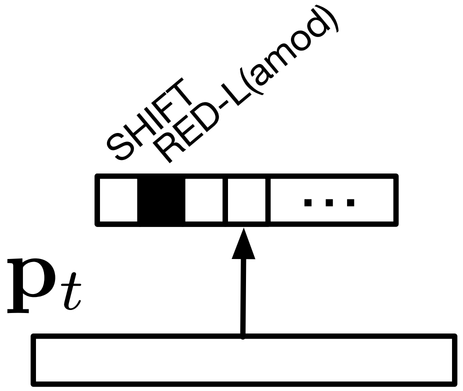
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L
I saw her duck	ROOT	



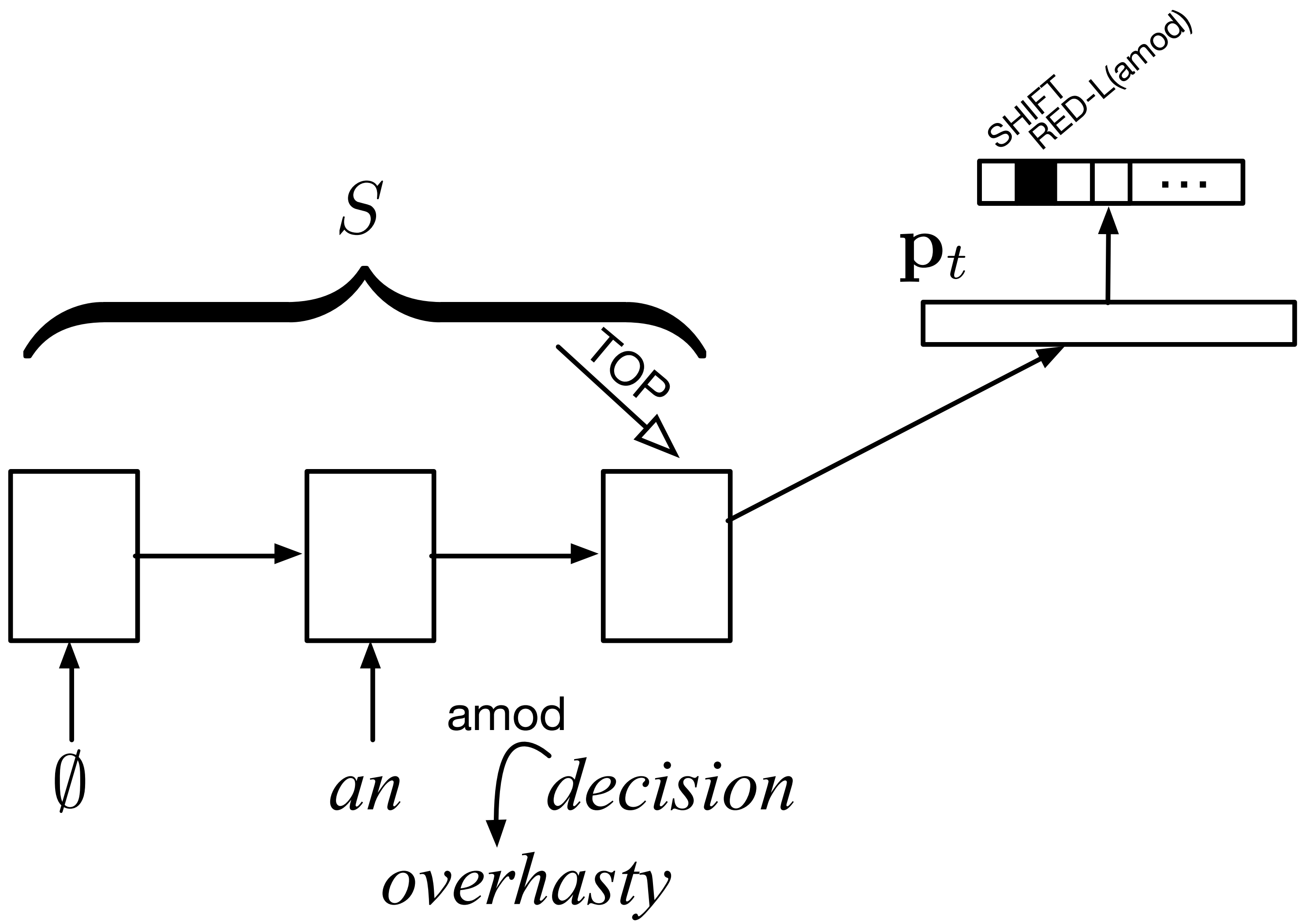
Stack
Buffer
Action
I saw her duck ROOT
SHIFT
I
saw her duck ROOT
SHIFT
I saw
her duck ROOT
REDUCE-L
I saw
her duck ROOT
SHIFT
I saw her
duck ROOT
SHIFT
I saw her duck
ROOT
REDUCE-L
I saw her duck
ROOT
REDUCE-R

Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L
I saw her duck	ROOT	REDUCE-R
I saw her duck		

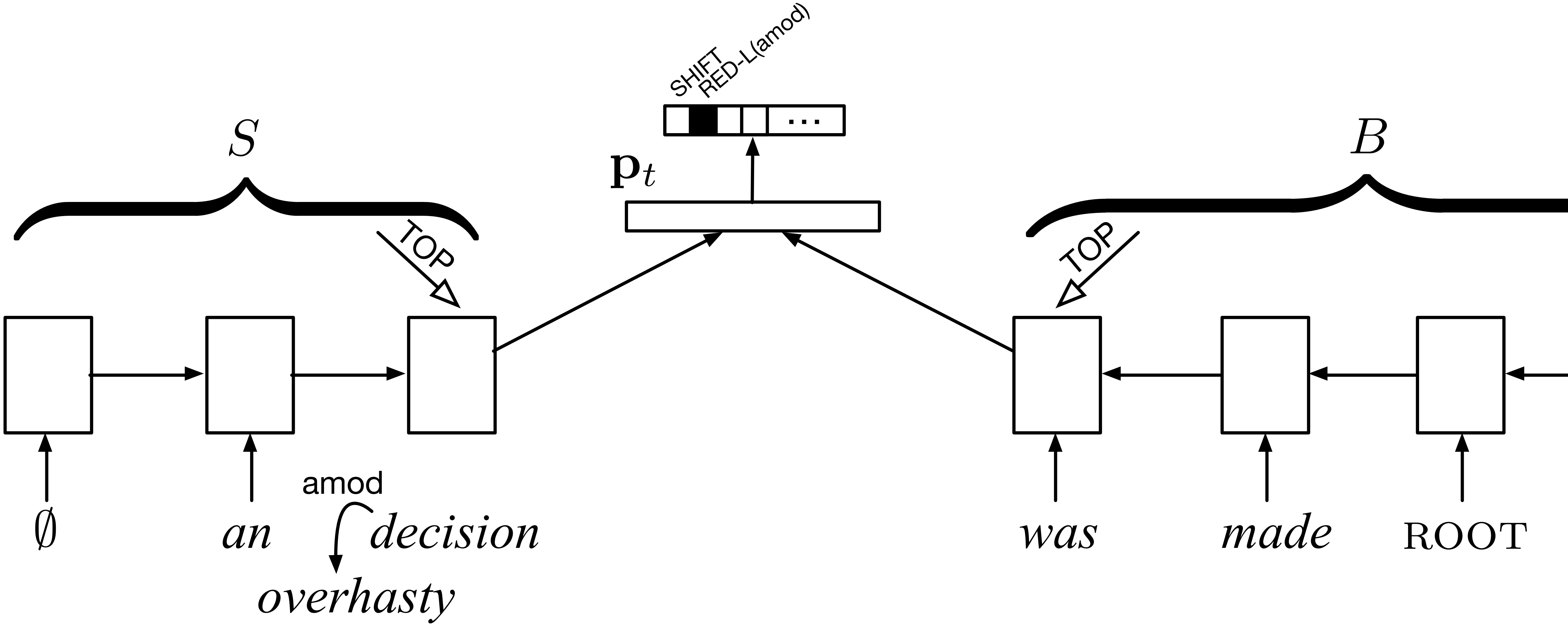
Stack	Buffer	Action
	I saw her duck ROOT	SHIFT
I	saw her duck ROOT	SHIFT
I saw	her duck ROOT	REDUCE-L
I saw	her duck ROOT	SHIFT
I saw her	duck ROOT	SHIFT
I saw her duck	ROOT	REDUCE-L
I saw her duck	ROOT	REDUCE-R
I saw her duck	ROOT	SHIFT
I saw her duck ROOT		REDUCE-R
I saw her duck ROOT		

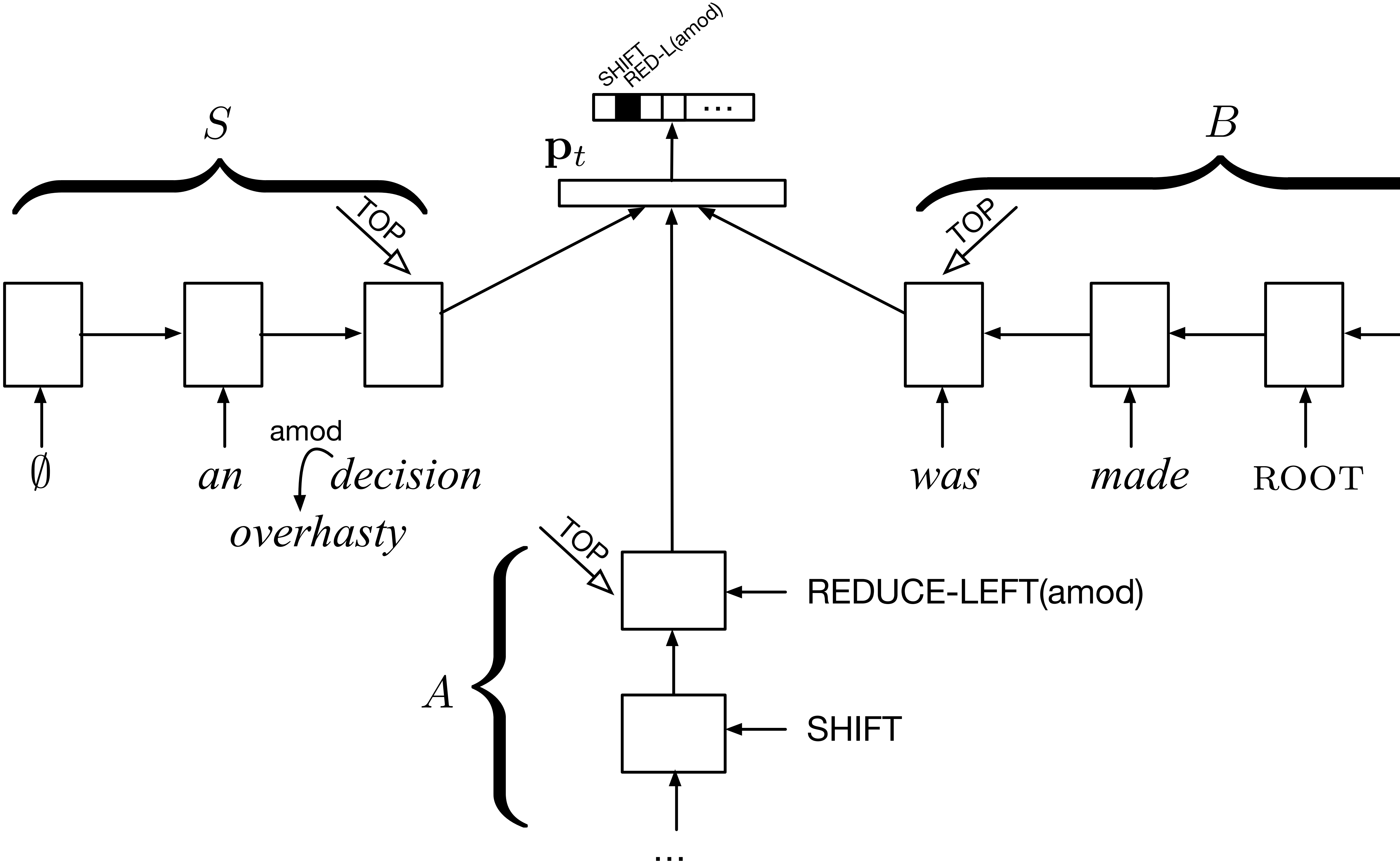


SHIFT
RED-L(amod)
...
$\mathbf{p}_t$



SHIFT
RED-L(amod)
...
$\mathbf{p}_t$
S
TOP
$\emptyset$
an
amod
decision
overhasty


SHIFT
RED-L(amod)
$\mathbf{p}_t$
S
B
TOP
TOP
amod
∅
an
decision
overhasty
was
made
ROOT



SHIFT
RED-L(amod)
...
$\mathbf{p}_t$
S
B
TOP
TOP
TOP
∅
an
amod
decision
overhasty
was
made
ROOT
A
REDUCE-LEFT(amod)
SHIFT
...

Some Results

	Accuracy
Feed forward NN	**93.1**
Stack LSTM	91.8

Dyer et al. (2015, ACL)

Topics

- Recursive neural networks for sentence representation
- Recurrent neural network grammars and parsing
- Word representations by looking inside words (words have structure too!)
- Analysis of neural networks with linguistic concepts

Word representation

ARBITRARINESS (de Saussure, 1916)

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

---

car

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

car   Auto   voiture   xe hơi

 ọkọ ayọkẹlẹ   koloi   sakyanan

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

car   Auto   voiture   xe hơi

   ọkọ ayọkẹlẹ   koloi   sakyanan

Is it reasonable to compose characters into “meanings”?

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

---

car   Auto   voiture   xe hơi

   ọkọ ayọkẹlẹ   koloi   sakyanan

OPPORTUNITY

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

car   Auto   voiture   xe hơi

   ọkọ ayọkẹlẹ   koloi   sakyanan

OPPORTUNITY

cool | cooooool | coooooooool

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar   cat − c + b = bat

car 

Auto
ọkọ ayọkẹlẹ

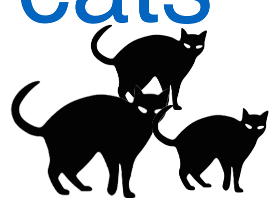
voiture
koloi

xe hơi
sakyanan

OPPORTUNITY

cool | cooooool | coooooooool

cat + s = cats

Word representation

ARBITRARINESS (de Saussure, 1916)

car − c + b = bar

cat − c + b = bat

car 

Auto

ọkọ ayọkẹlẹ

voiture

koloi

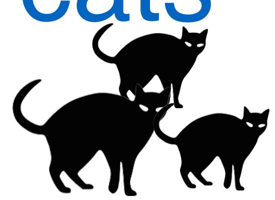
xe hơi

sakyanan

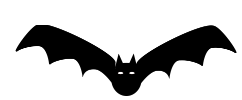
OPPORTUNITY

cool | cooooool | cooooooooool

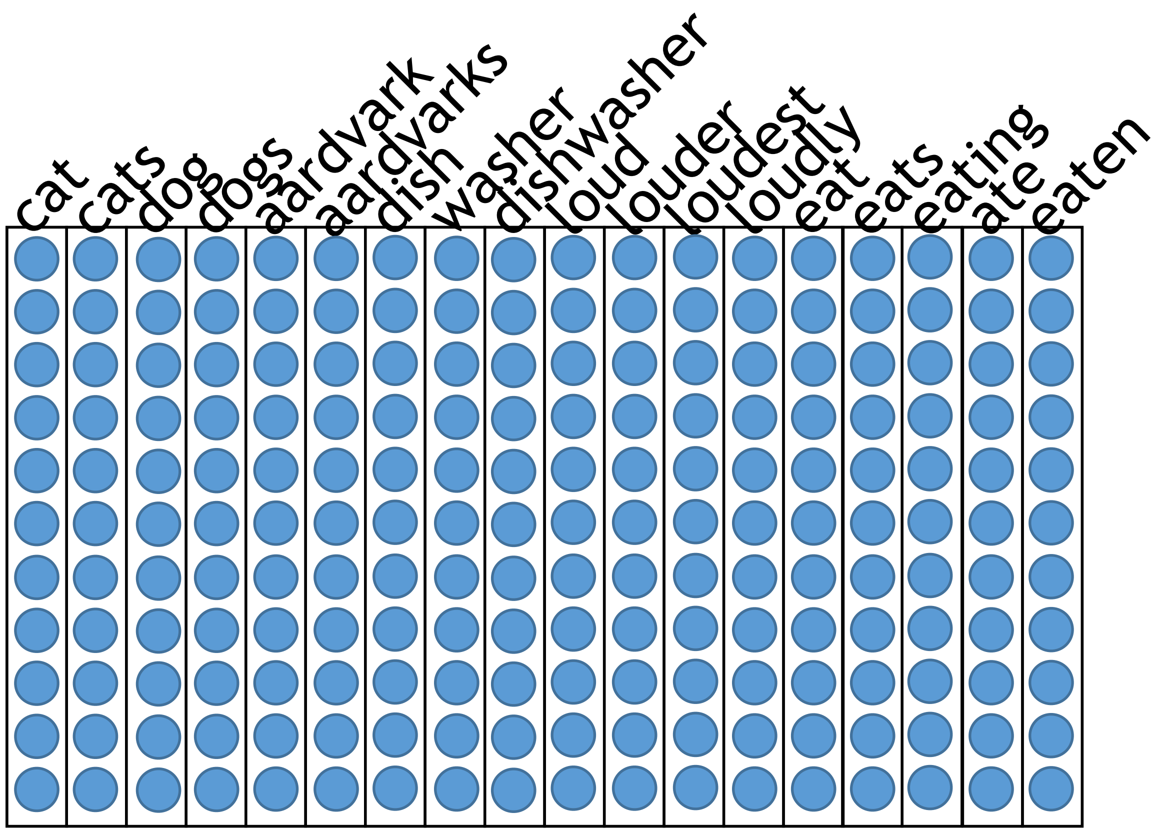
cat + s = cats

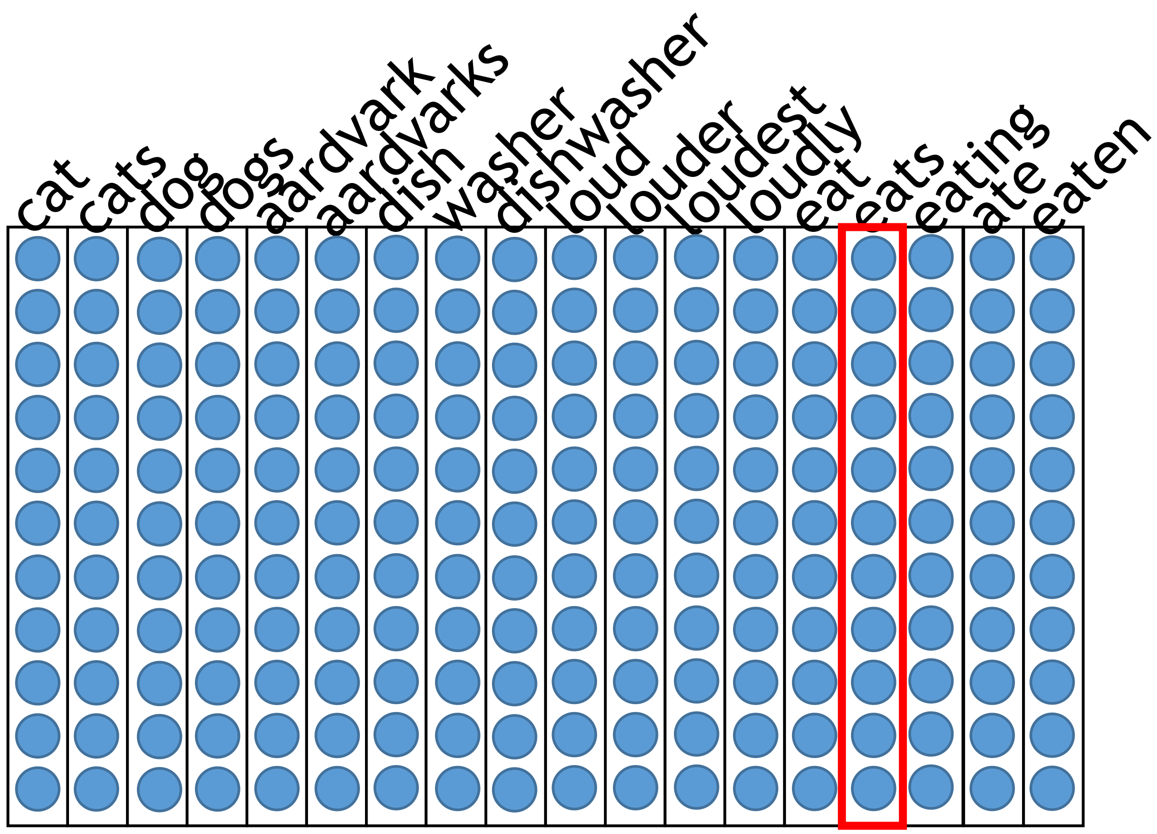
bat + s = bats

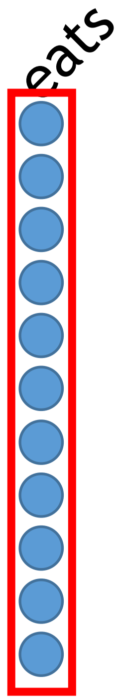
Words as structured objects



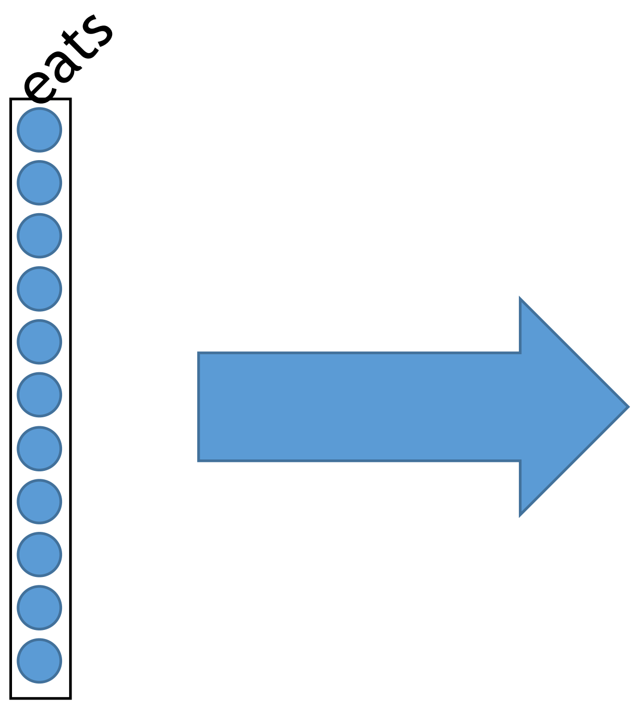
Words as structured objects



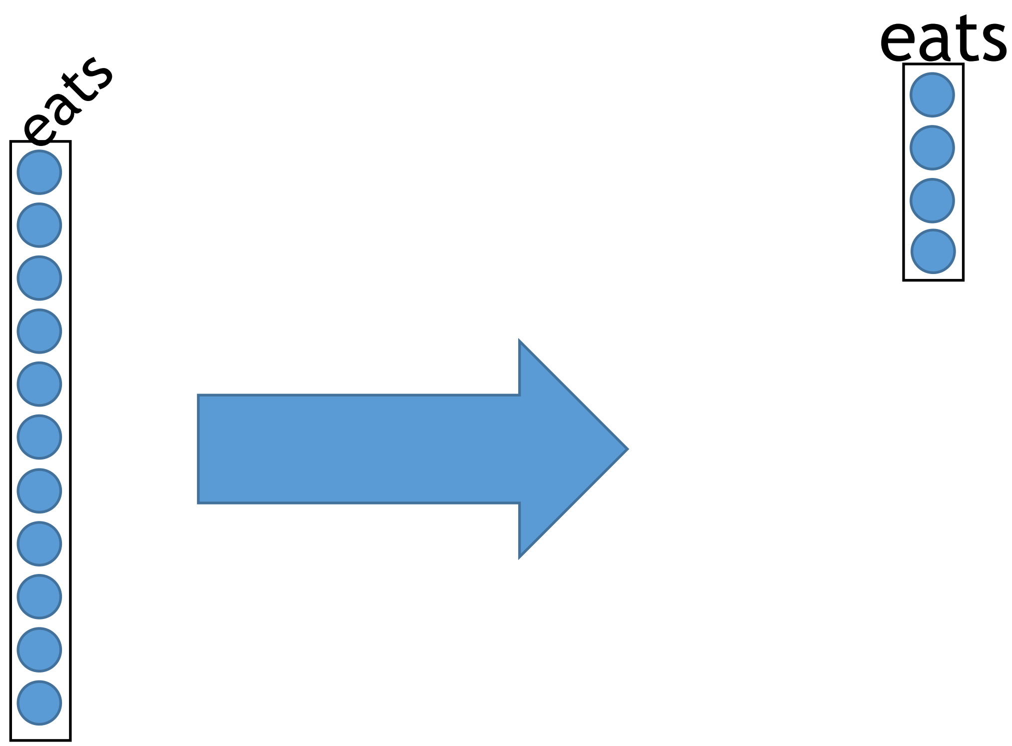
Words as structured objects



Words as structured objects

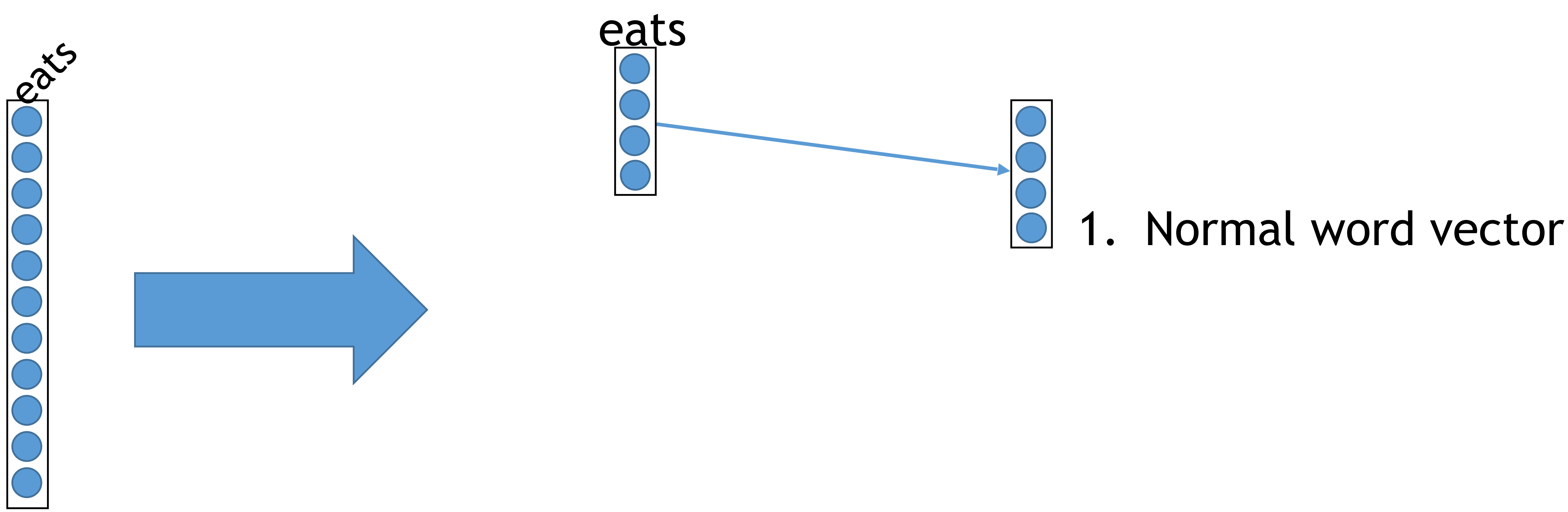


Words as structured objects

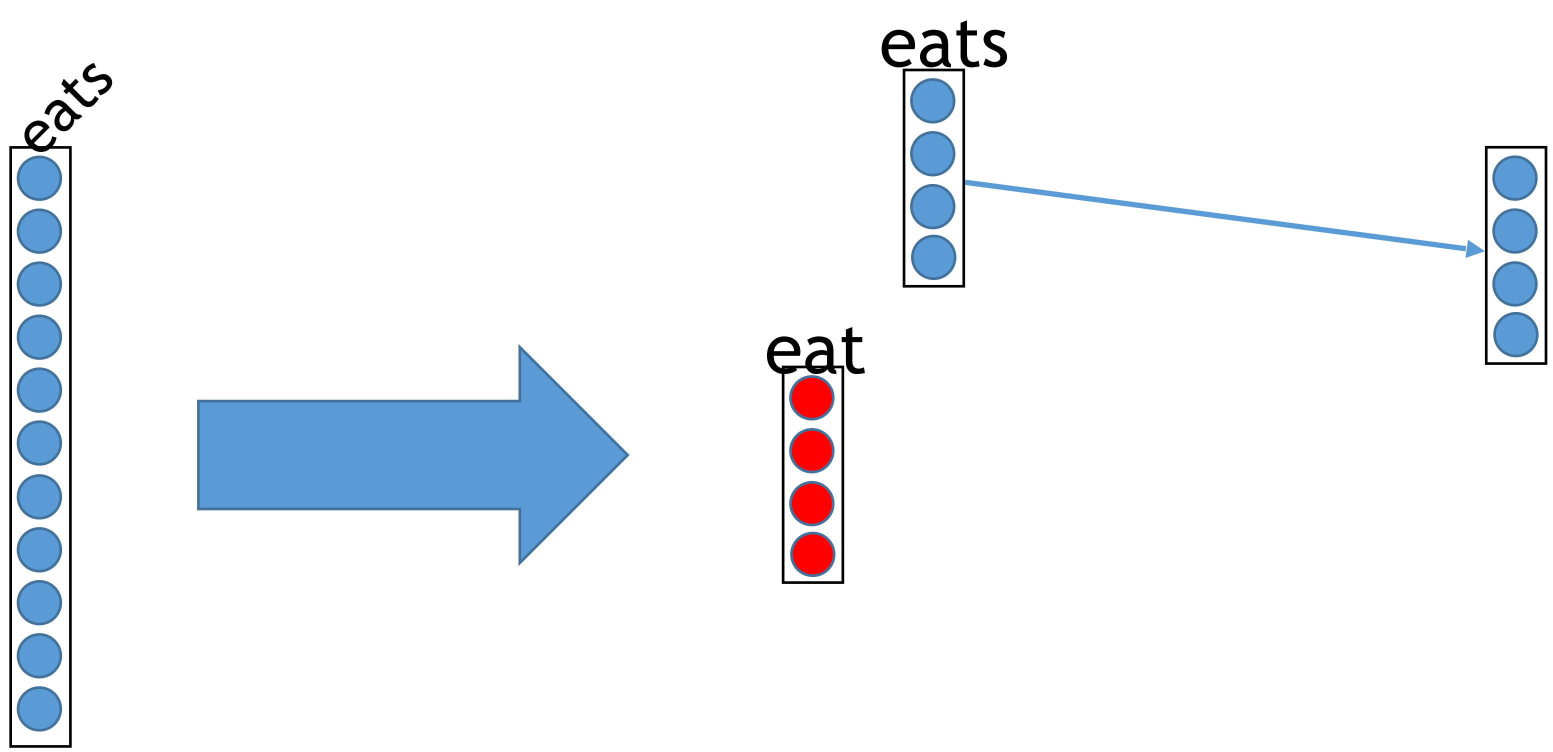


1. Normal word vector

Words as structured objects

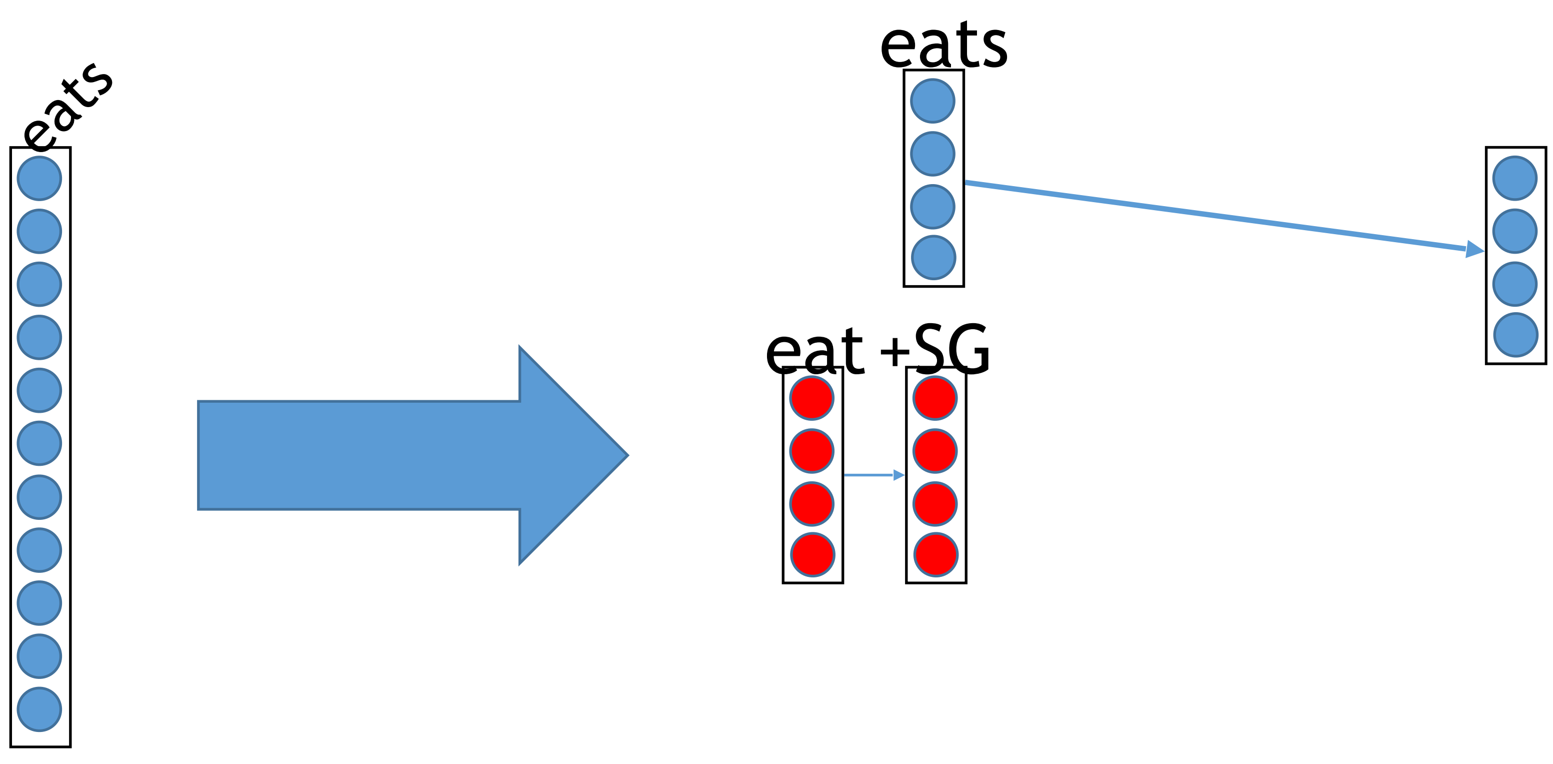


Words as structured objects



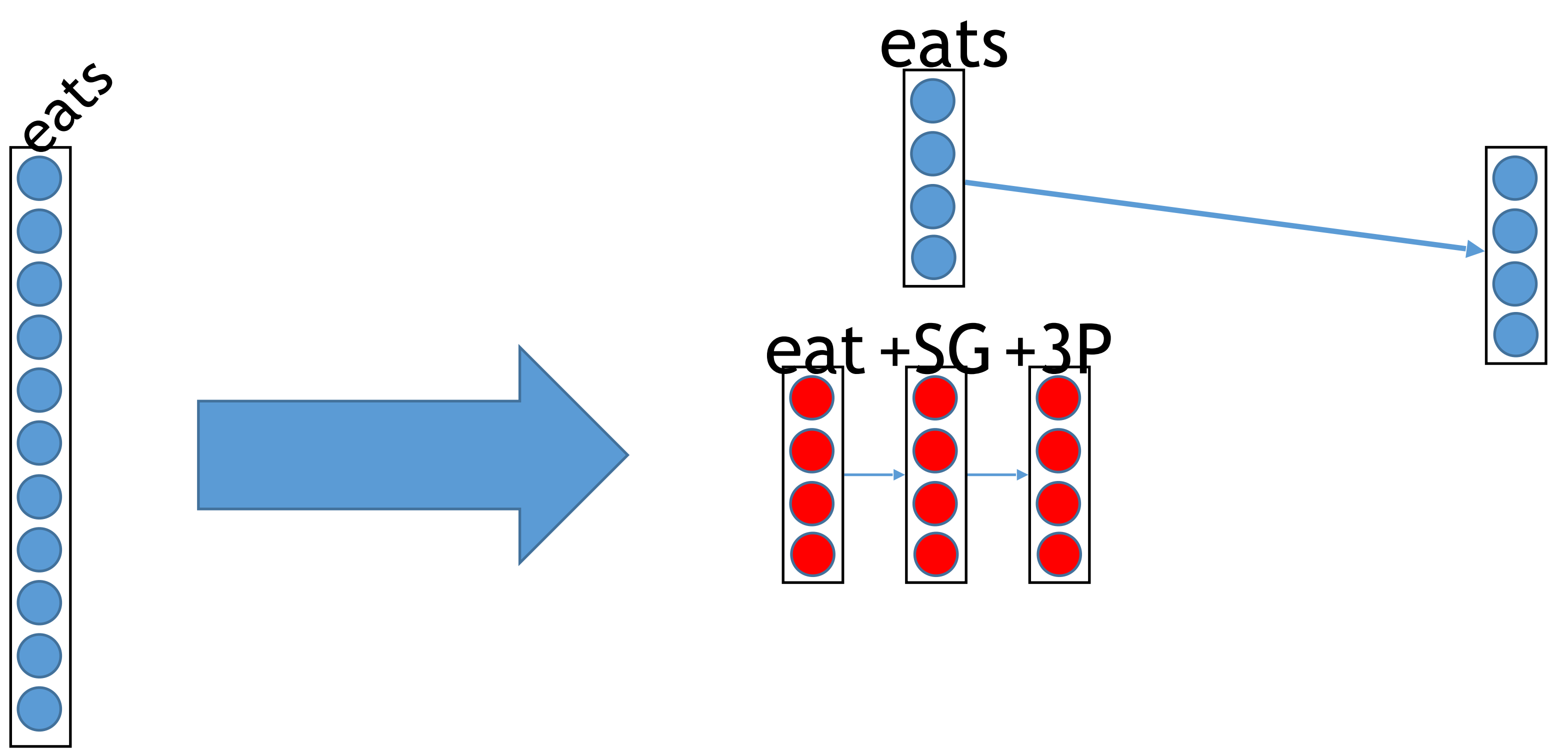
1. Normal word vector
2. Morphological word vector

Words as structured objects



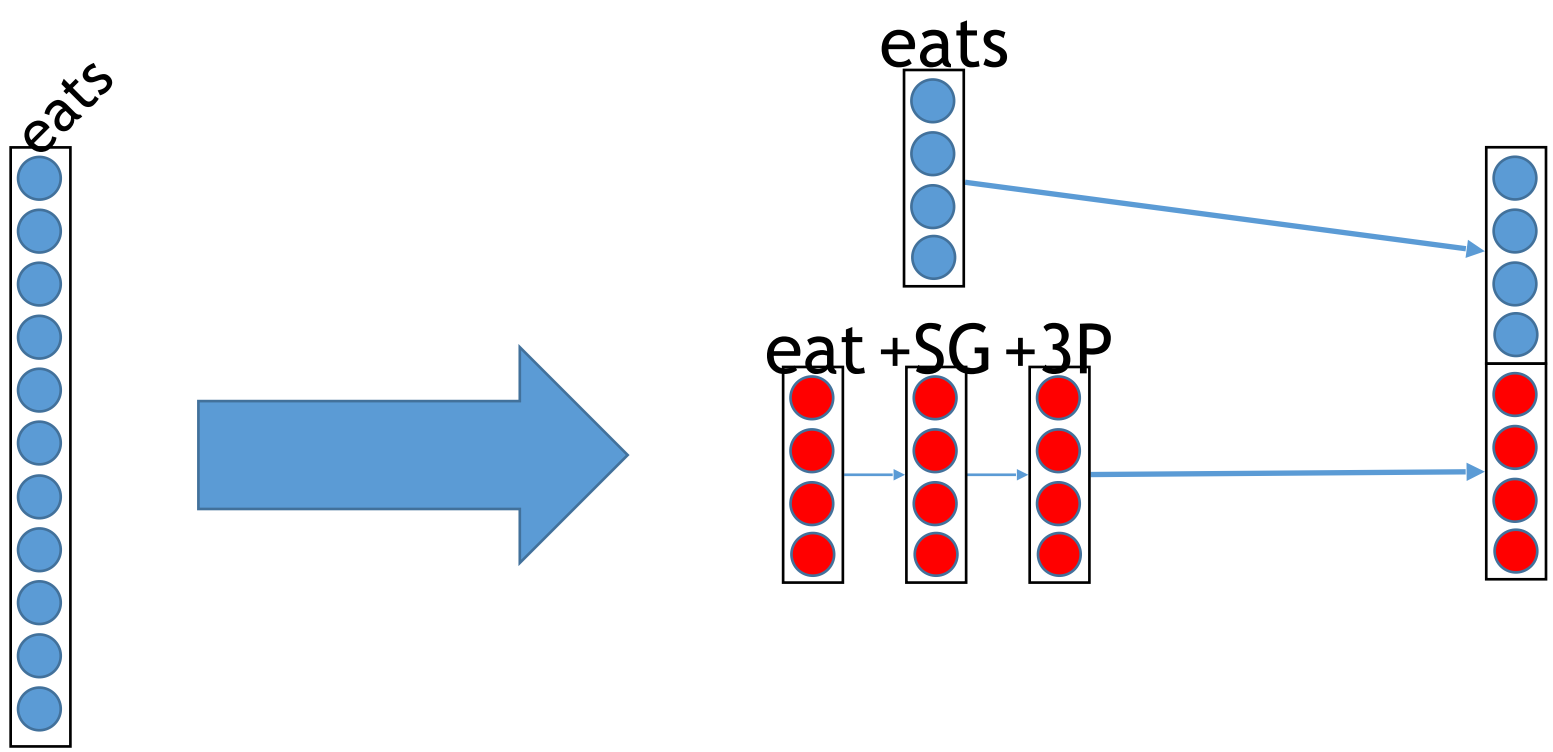
1. Normal word vector
2. Morphological word vector

Words as structured objects



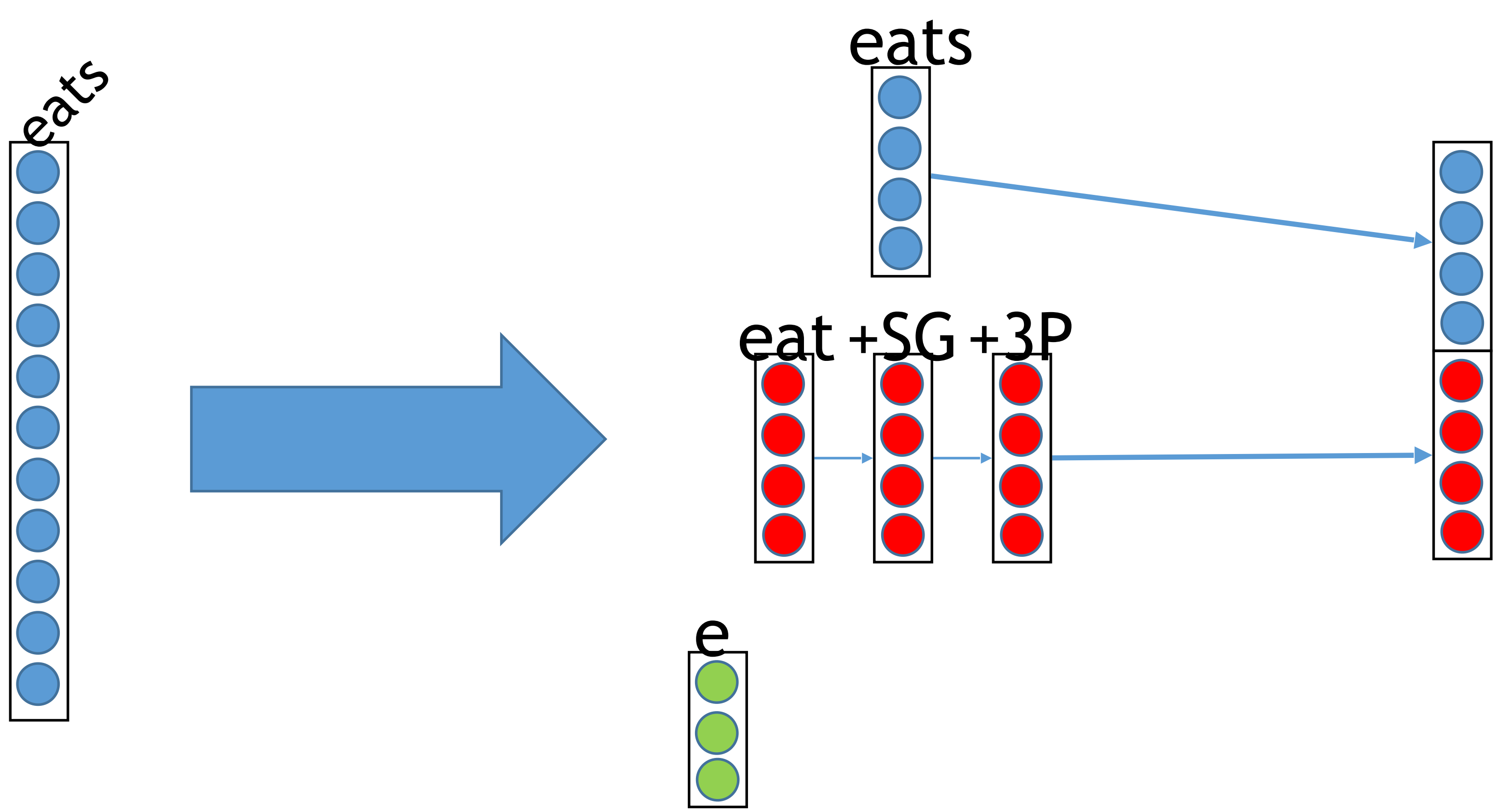
1. Normal word vector
2. Morphological word vector

Words as structured objects



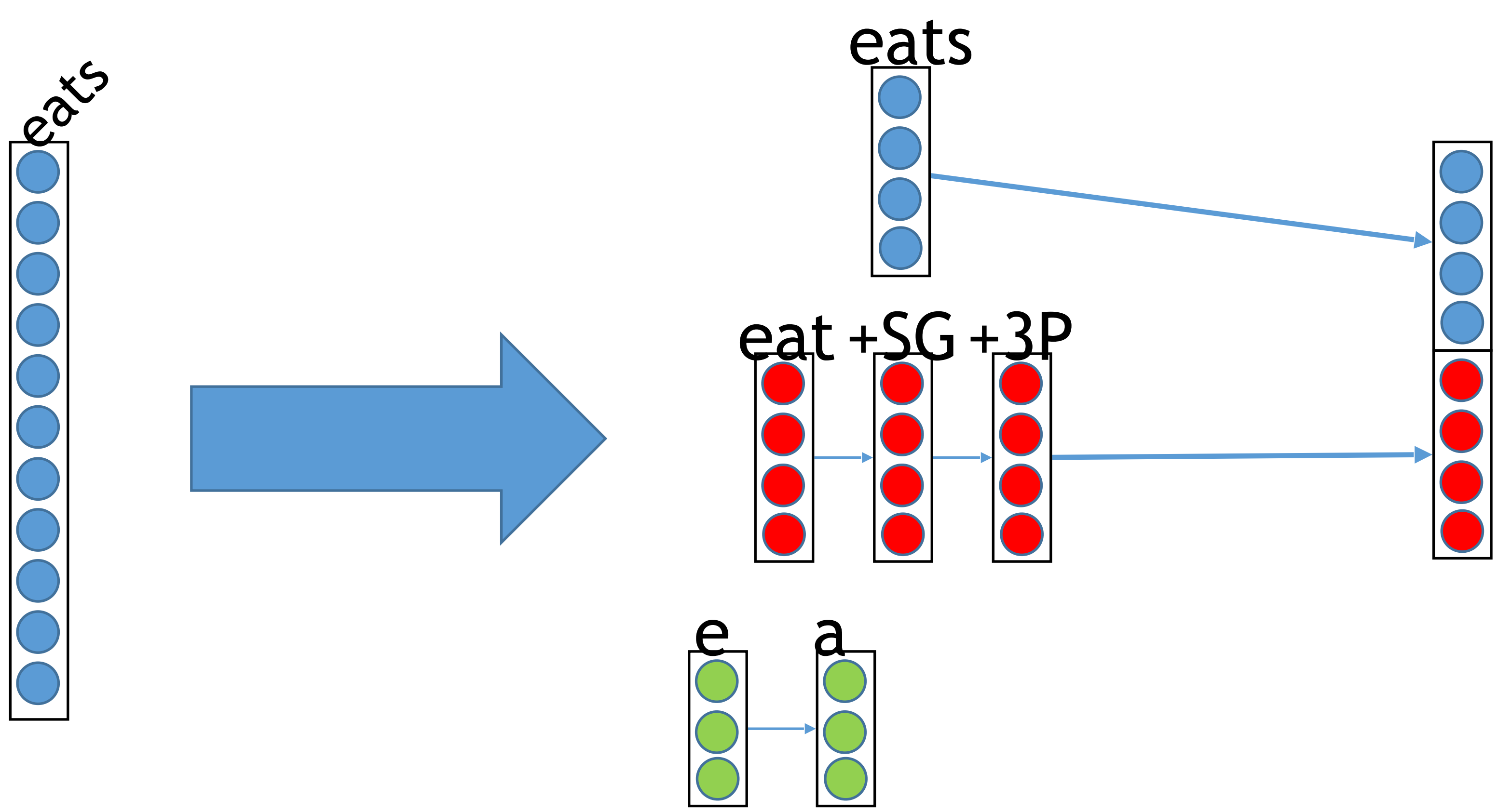
1. Normal word vector
2. Morphological word vector

Words as structured objects



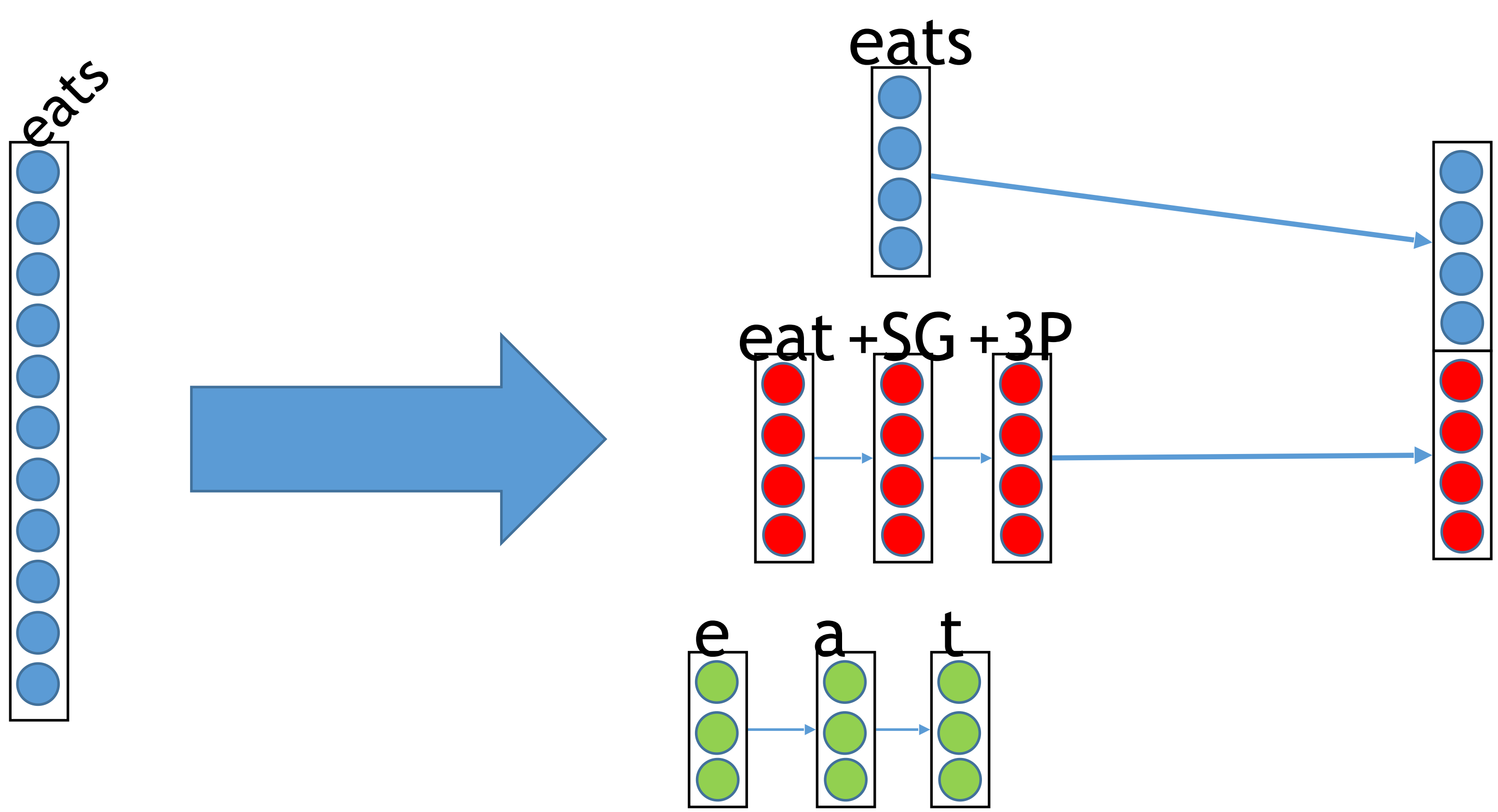
1. Normal word vector
2. Morphological word vector
3. Character-based word vector

Words as structured objects



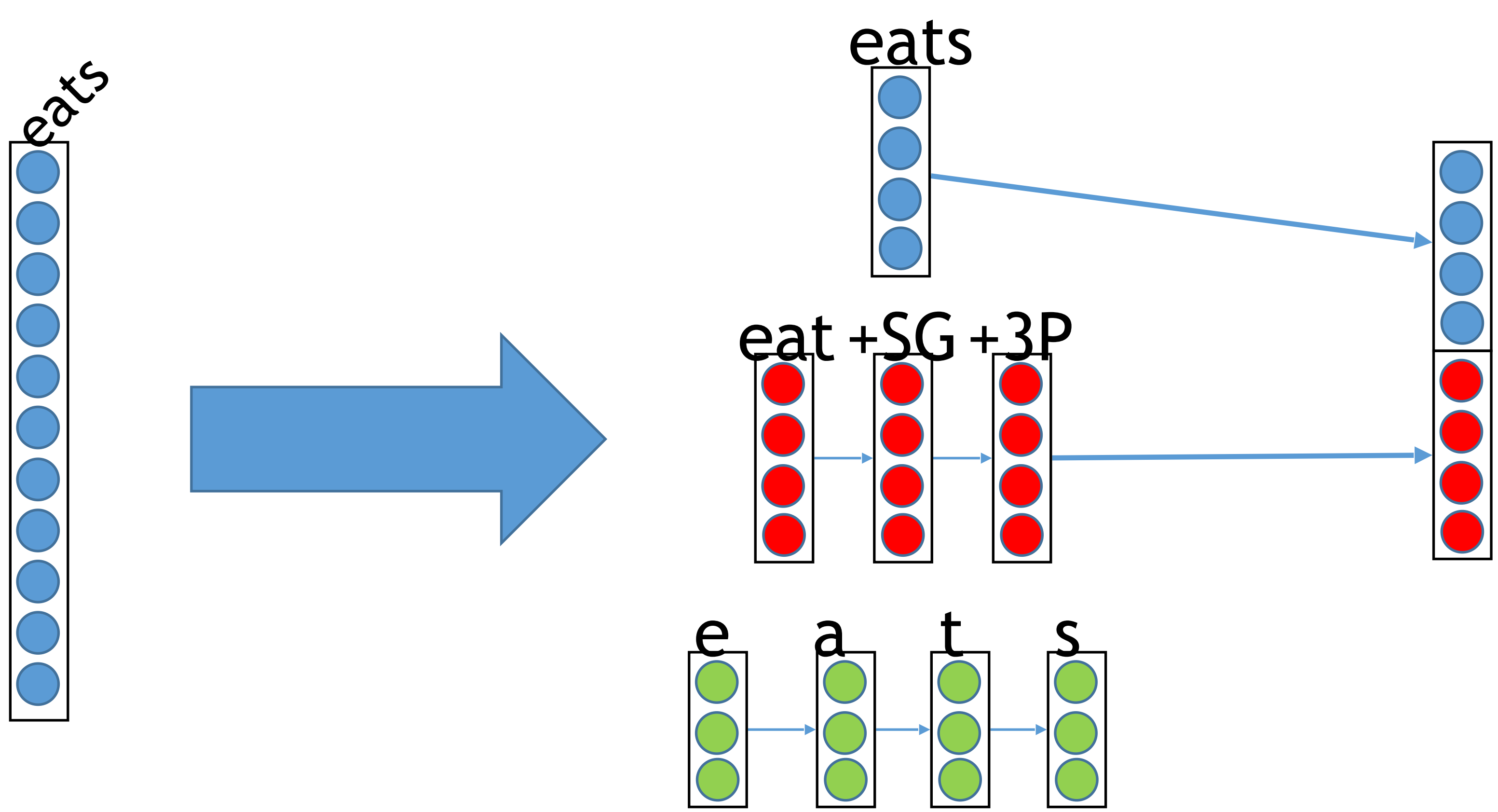
1. Normal word vector
2. Morphological word vector
3. Character-based word vector

Words as structured objects



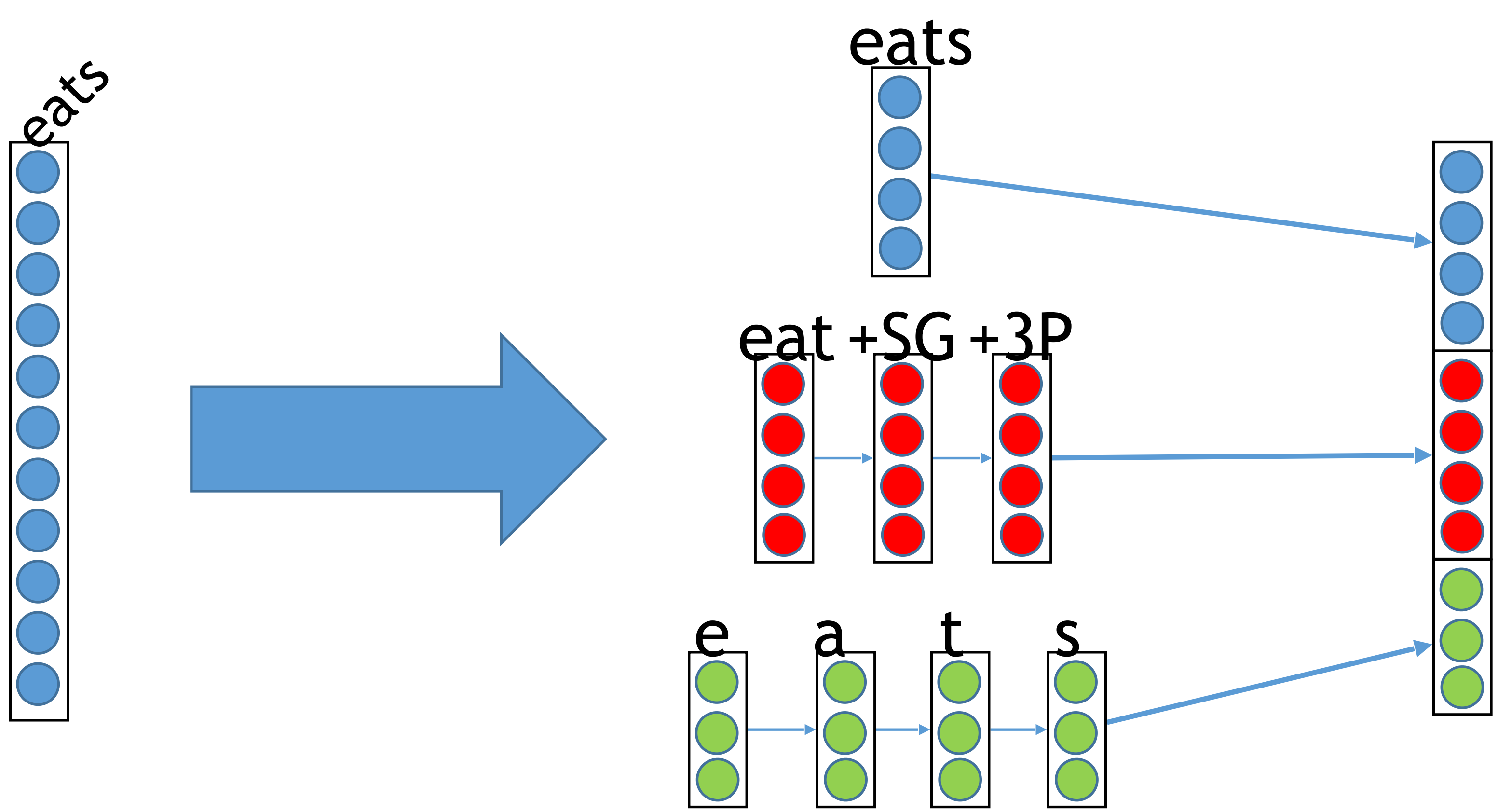
1. Normal word vector
2. Morphological word vector
3. Character-based word vector

Words as structured objects



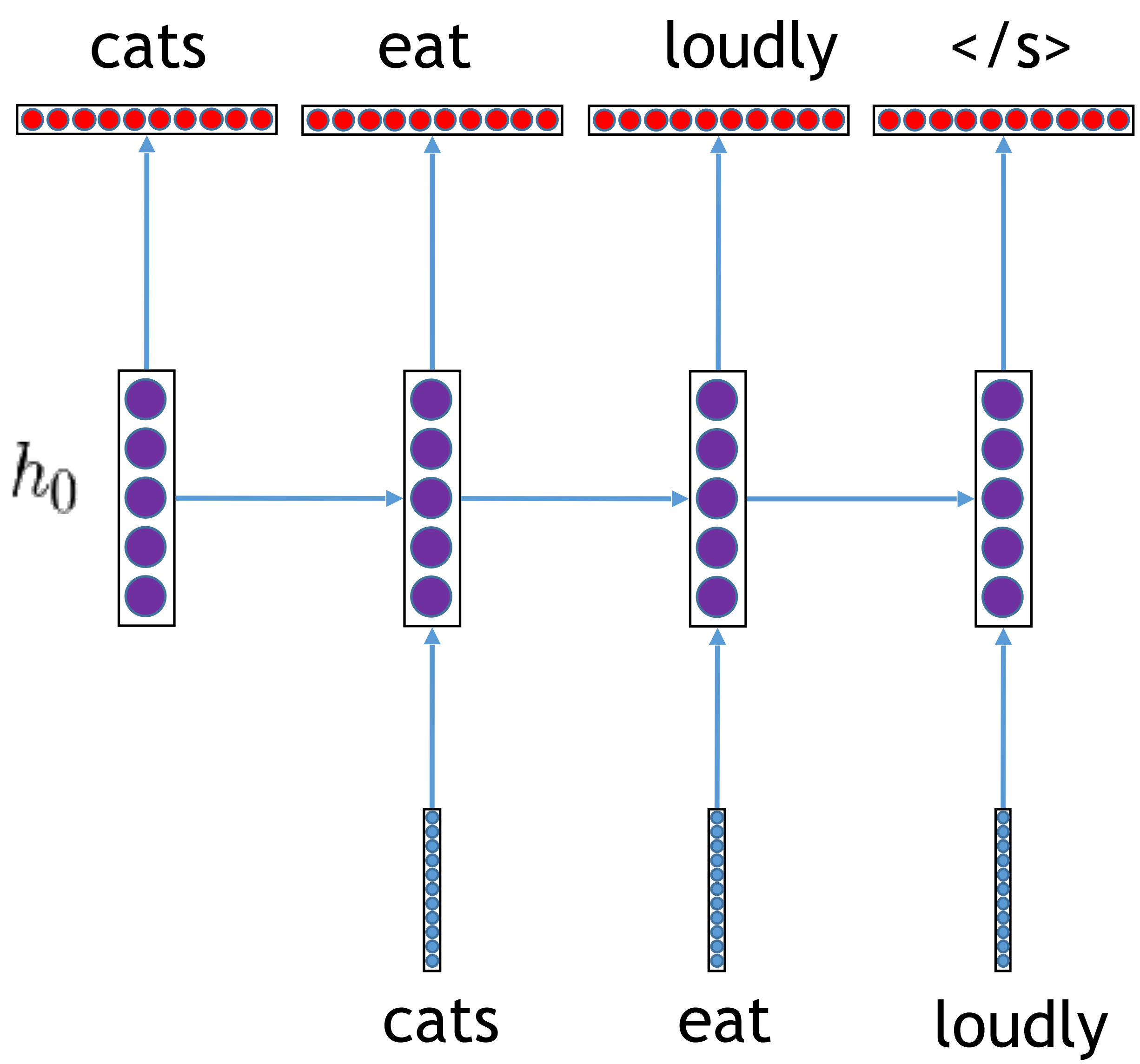
1. Normal word vector
2. Morphological word vector
3. Character-based word vector

Words as structured objects

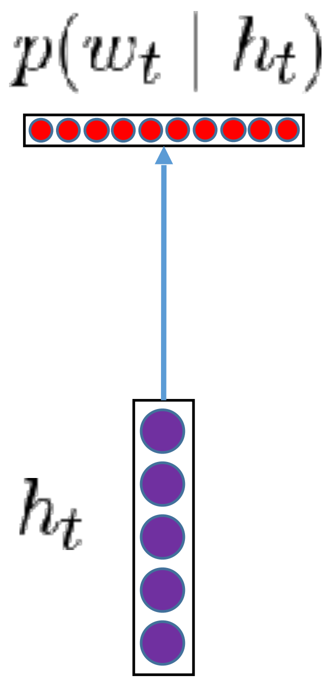


1. Normal word vector
2. Morphological word vector
3. Character-based word vector

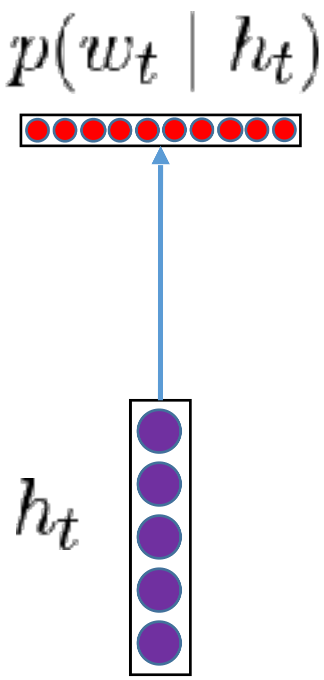
Generating new word forms



Generating new word forms

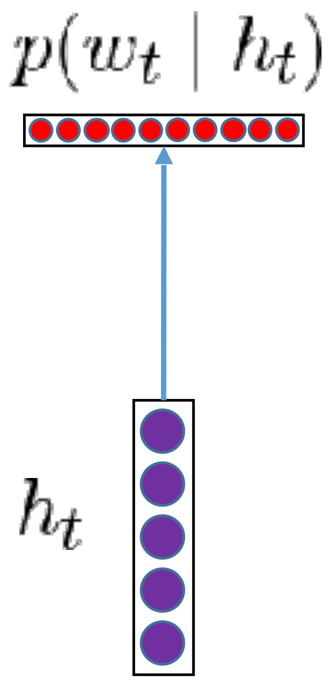


Generating new word forms



- Normally we model $p(w_t \mid h_t)$ directly.

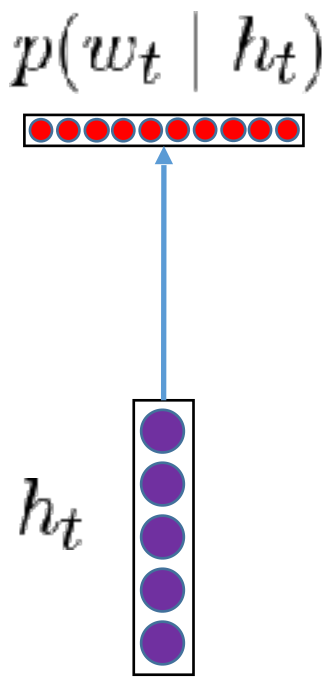
Generating new word forms



- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

Generating new word forms



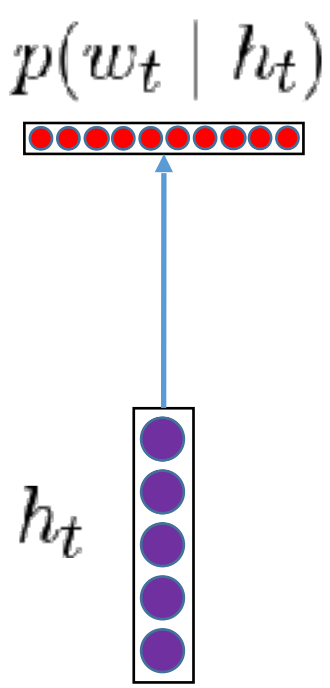
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms

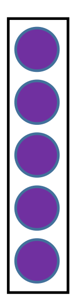
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\; p(m \mid h_t)$$

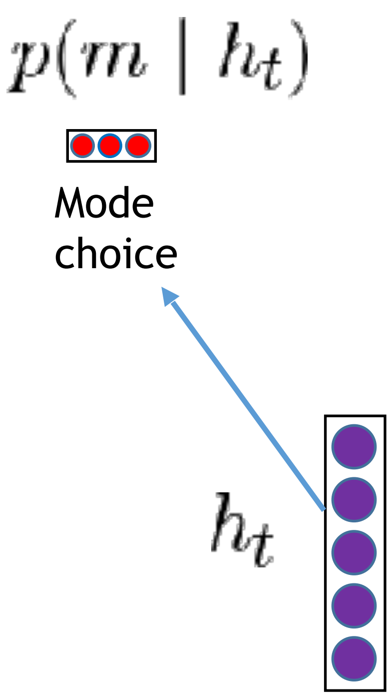
where *m* loops over three "generation modes":

- words
- morphemes
- characters

h_t



Generating new word forms



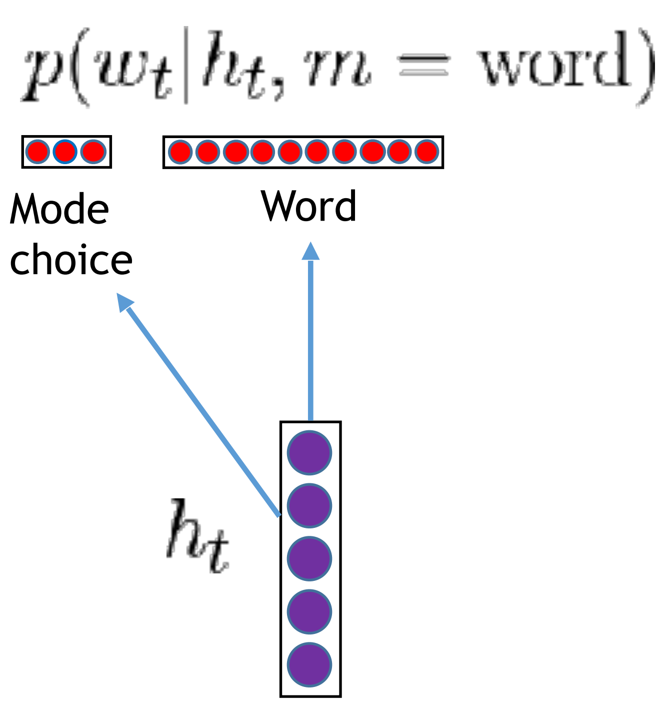
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



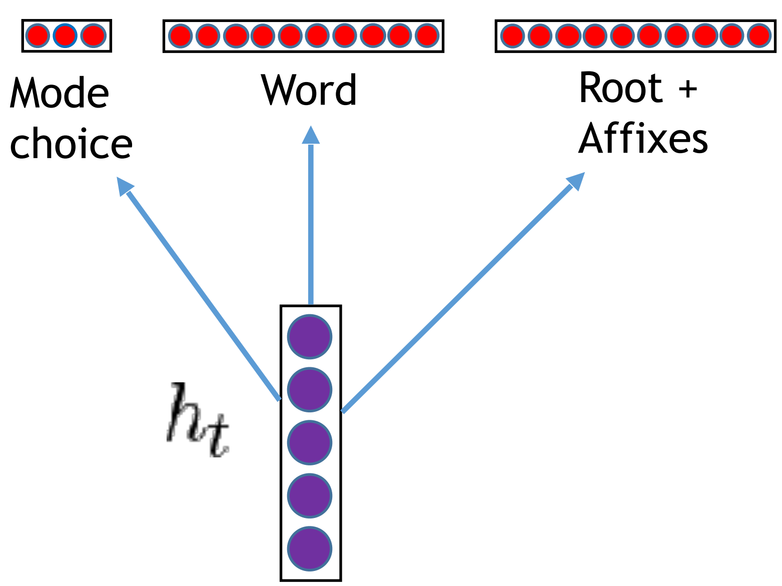
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



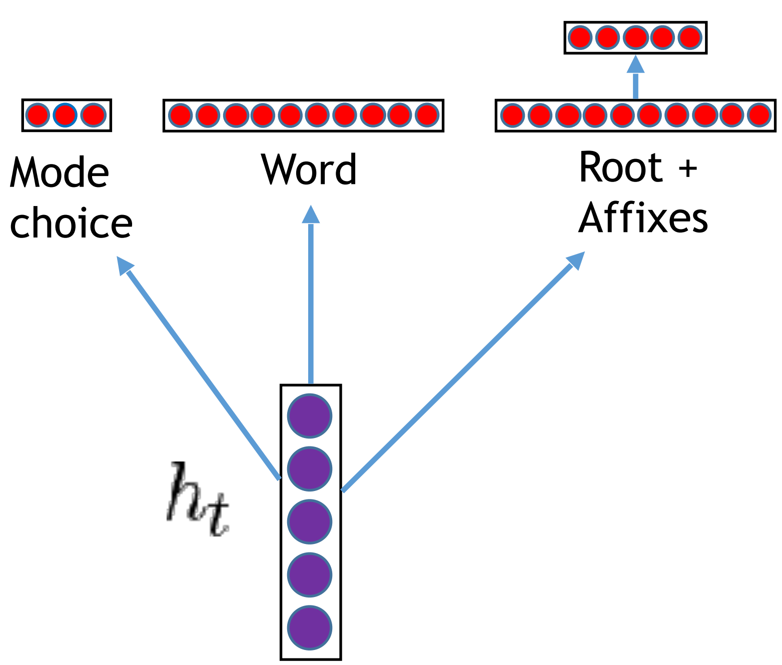
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\; p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



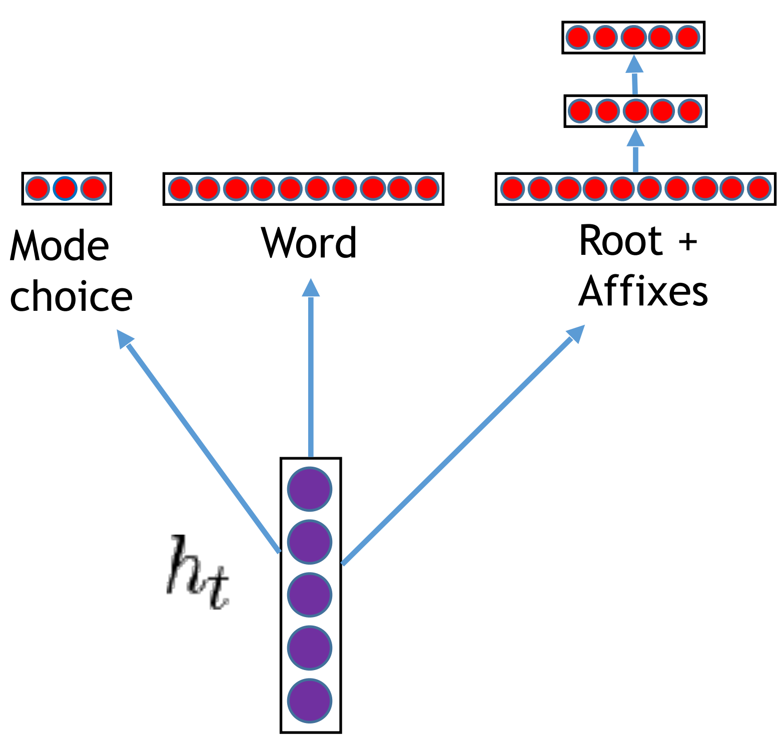
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\; p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

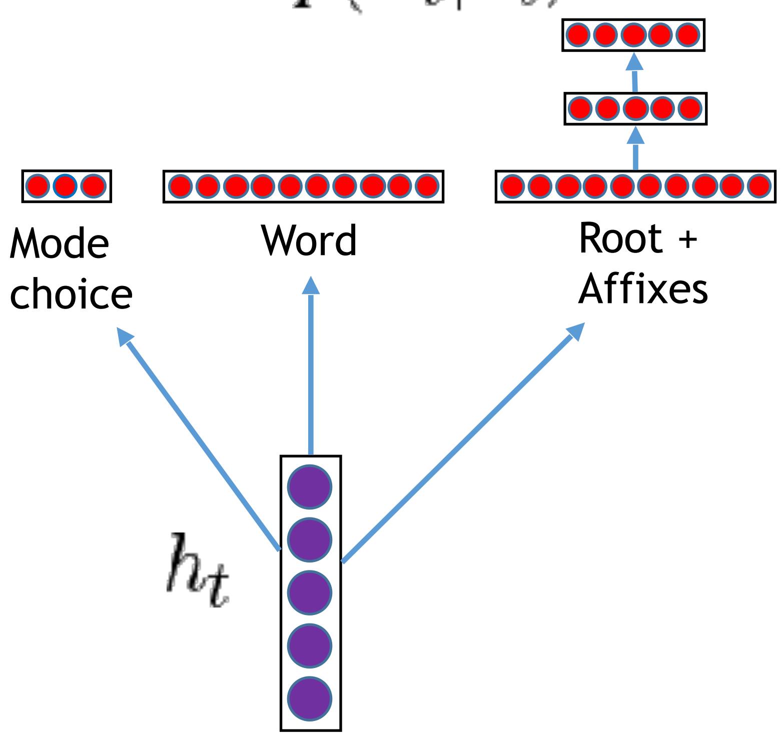
$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms

$$p(w_t | h_t, m = \mathrm{morph})$$



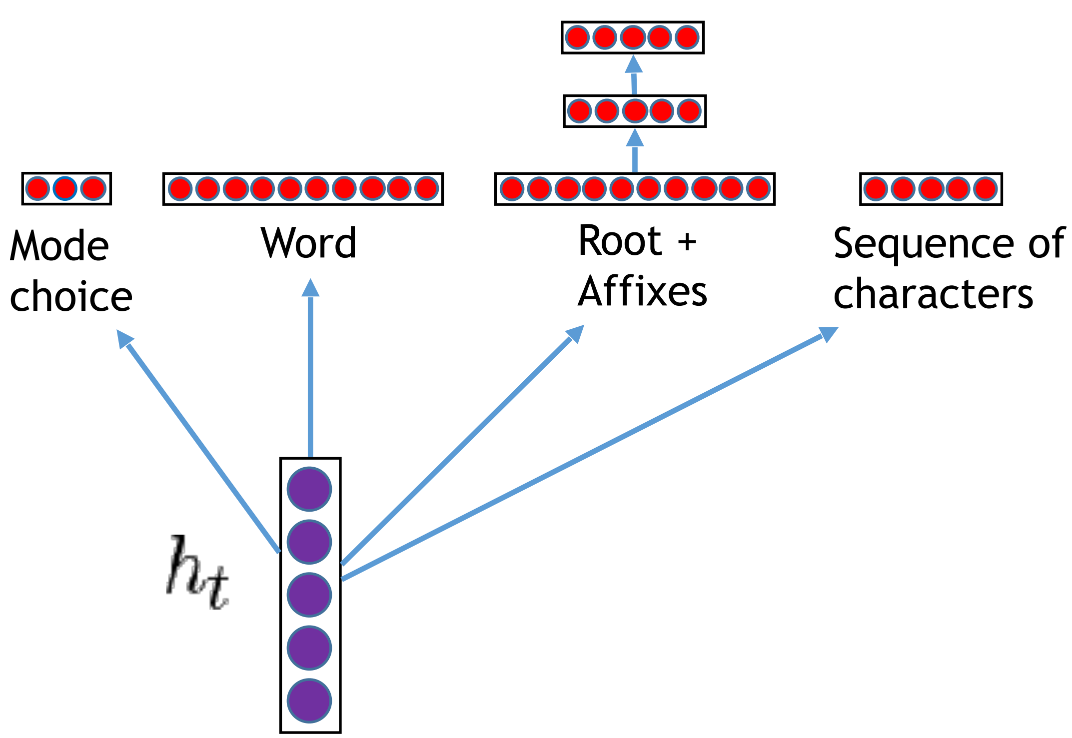
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



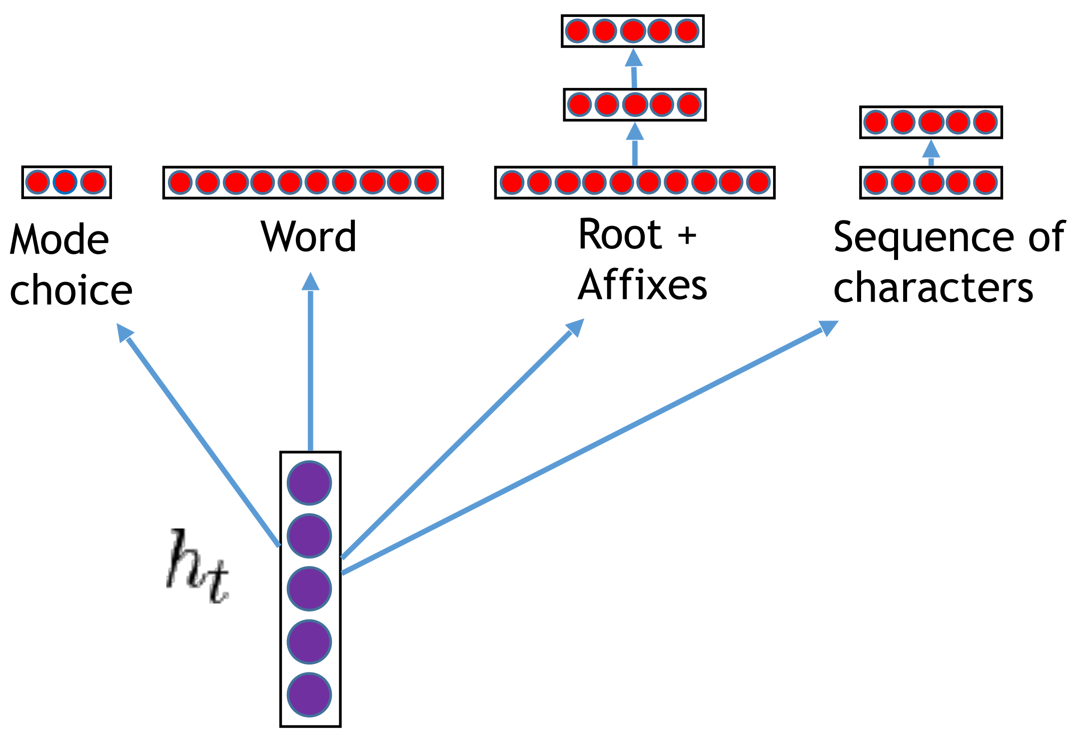
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



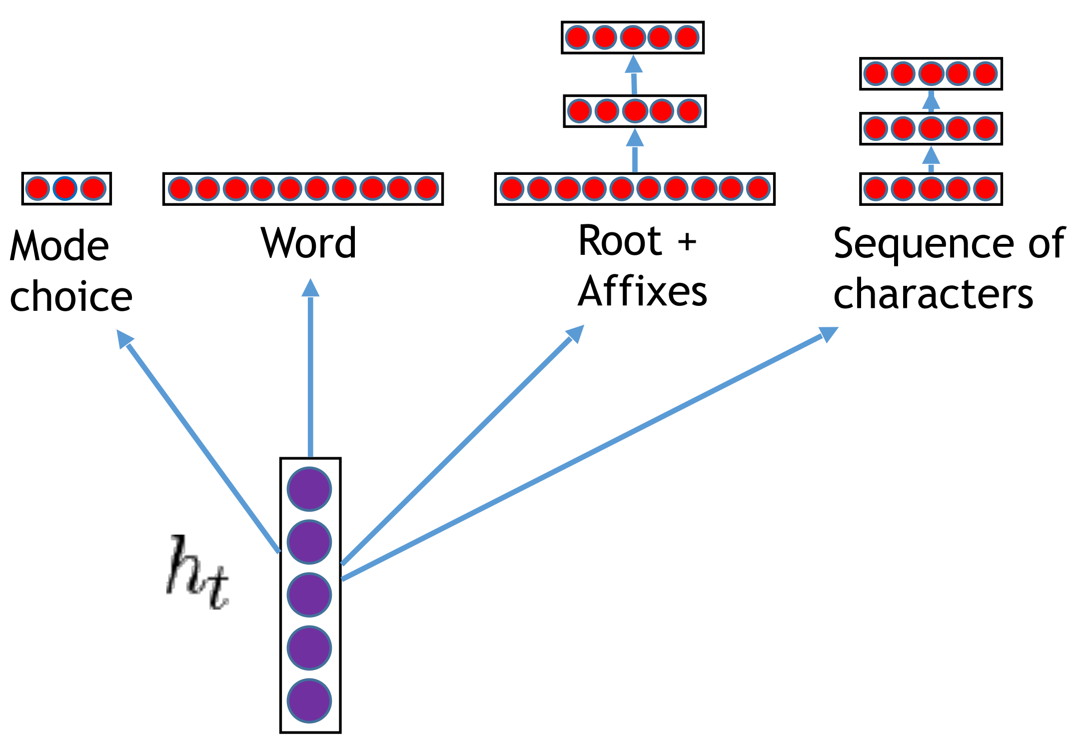
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



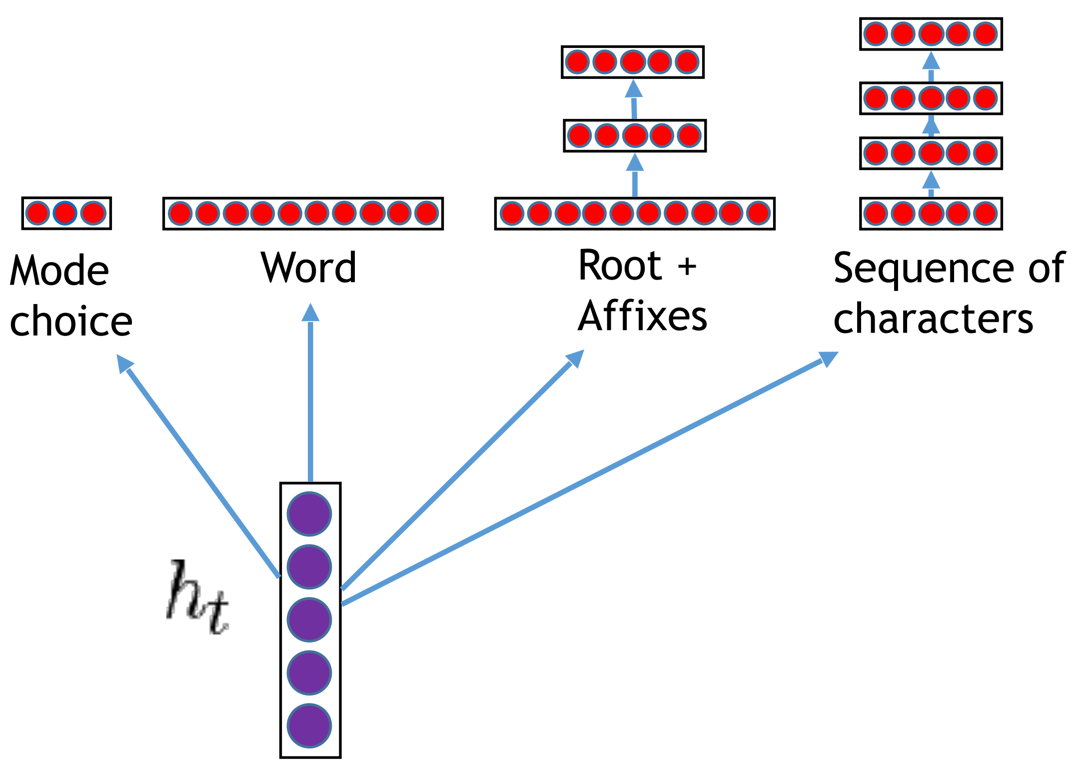
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



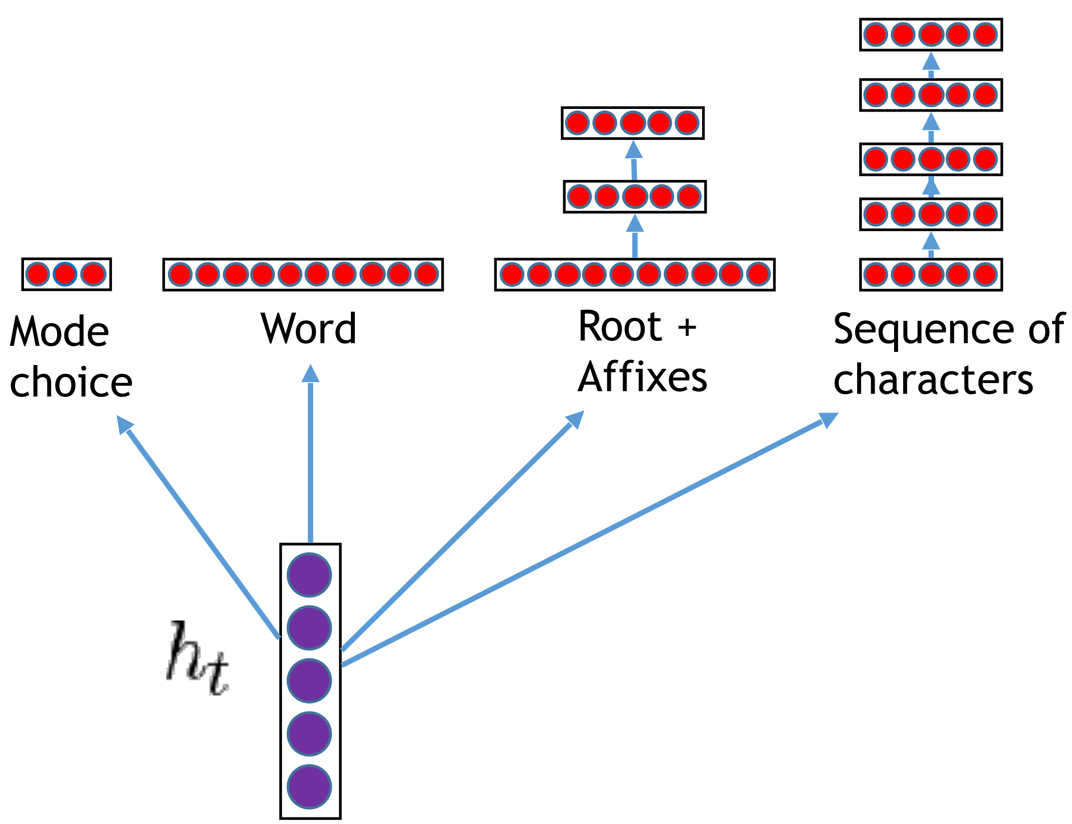
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



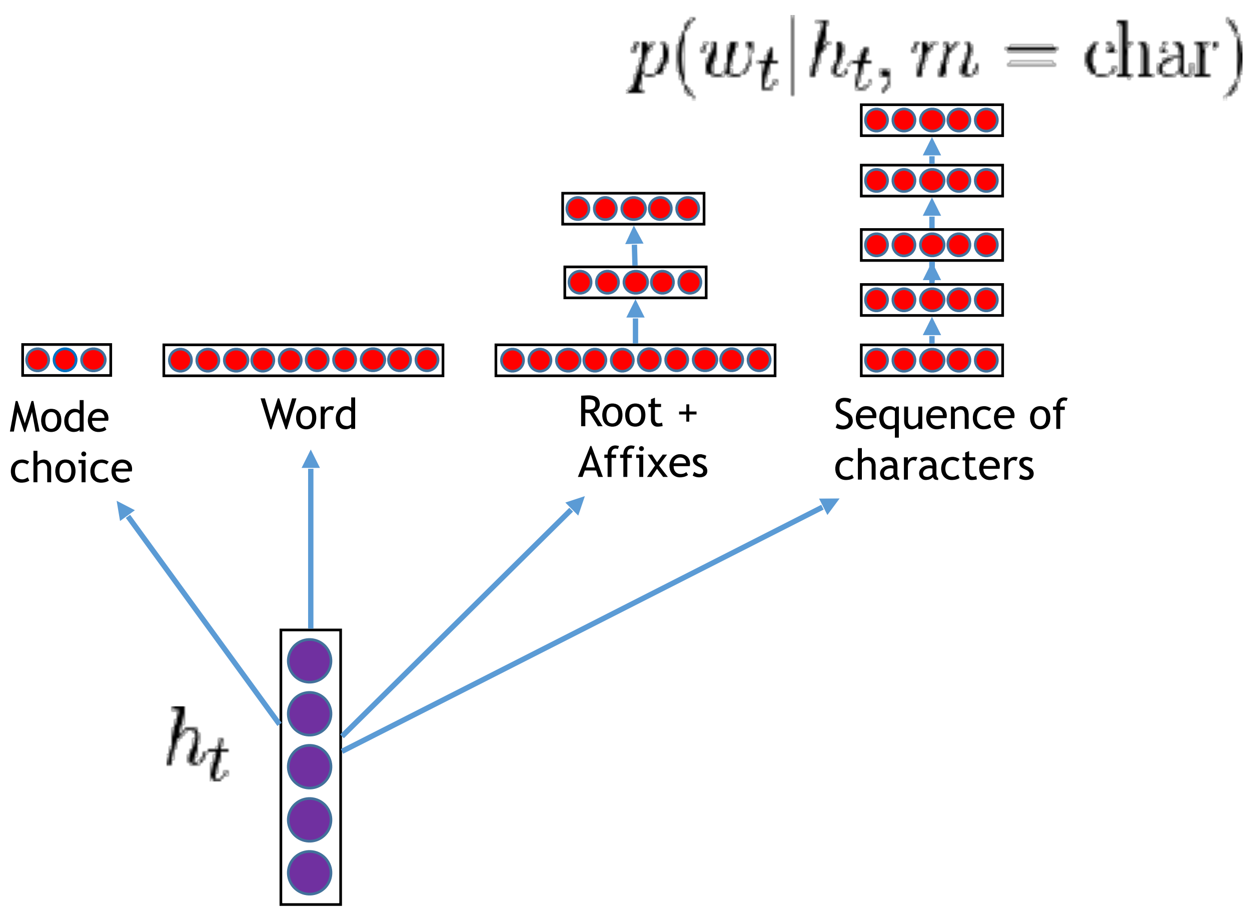
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



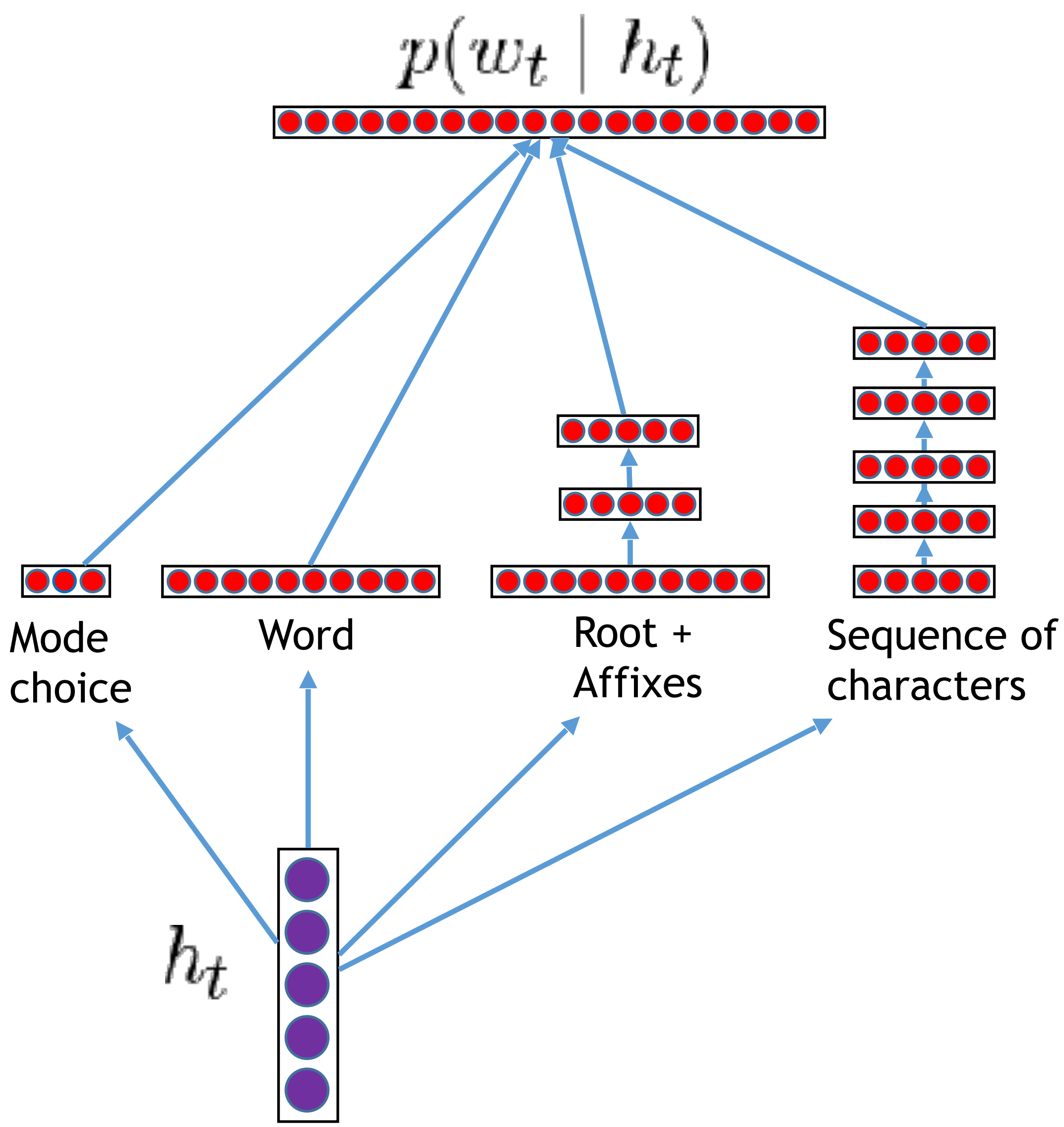
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\; p(m \mid h_t)$$

where *m* loops over three "generation modes":

- words
- morphemes
- characters

Generating new word forms



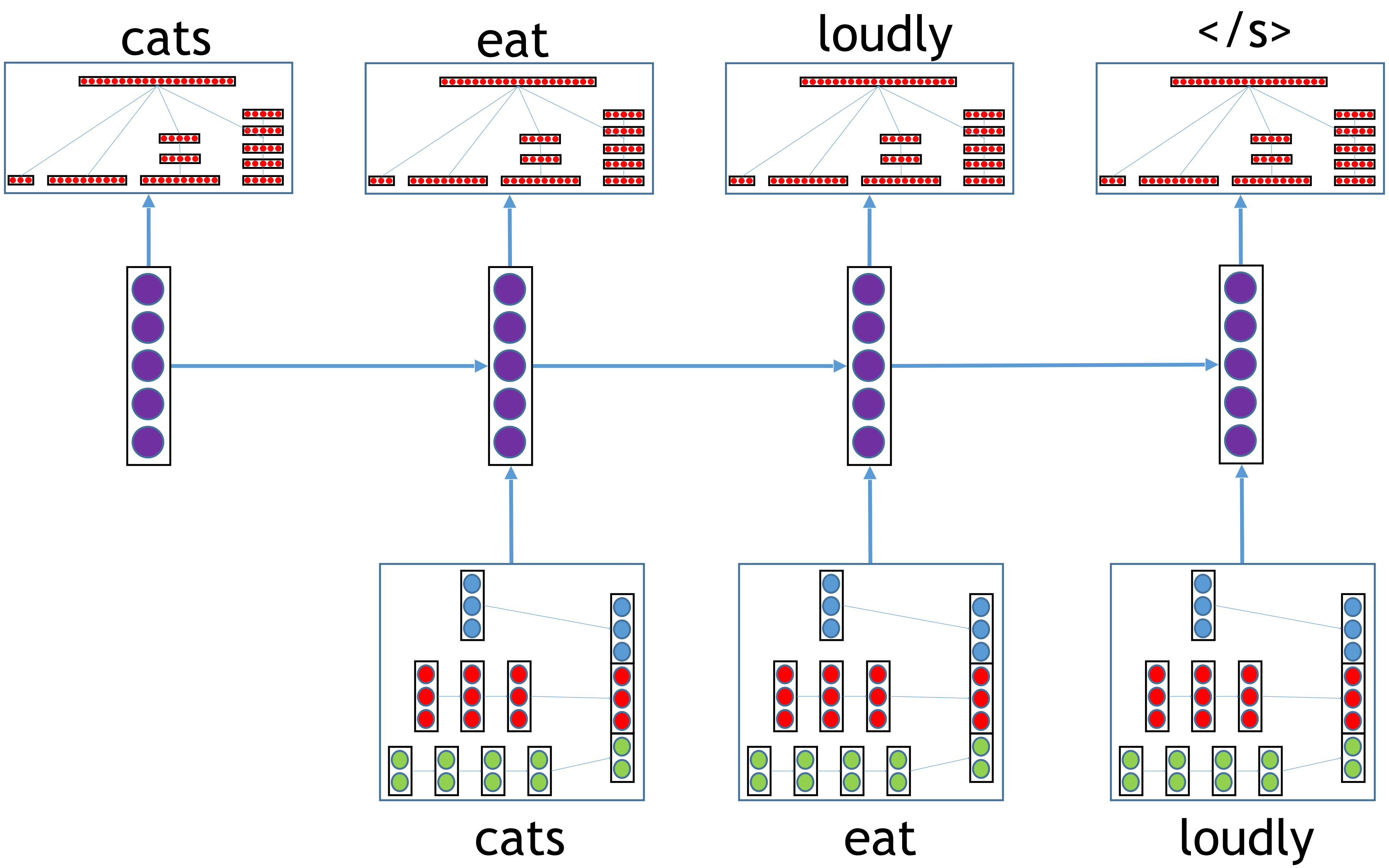
- Normally we model $p(w_t \mid h_t)$ directly.
- Instead let's model

$$p(w_t \mid h_t) = \sum_m p(w_t \mid h_t, m)\, p(m \mid h_t)$$

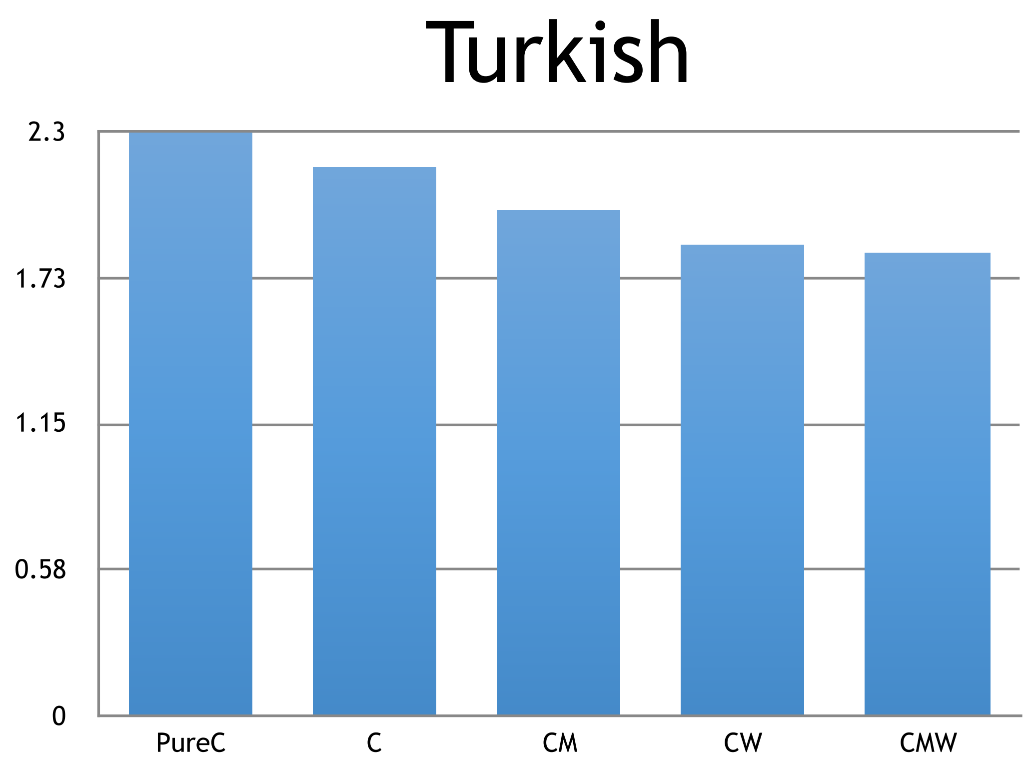
where *m* loops over three "generation modes":

- words
- morphemes
- characters

Putting it all together



Language modeling results



Lower is better.

Columns:

RNN predicts language as a sequences of characters

Compositional character model only

Character+morpheme model

Character+word embedding model

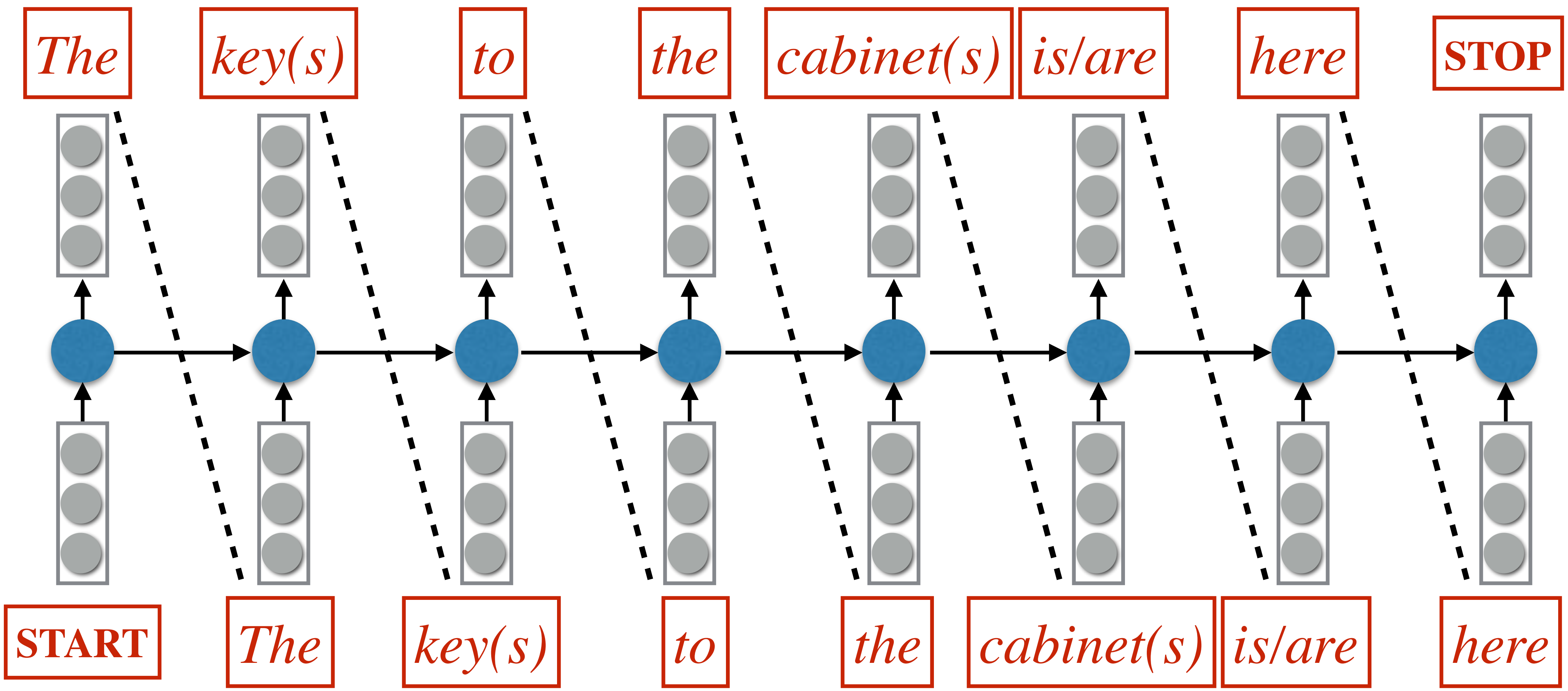
Character+morpheme+word embedding model

Topics

- Recursive neural networks for sentence representation
- Recurrent neural network grammars and parsing
- Word representations by looking inside words (words have structure too!)
- Analysis of neural networks with linguistic concepts

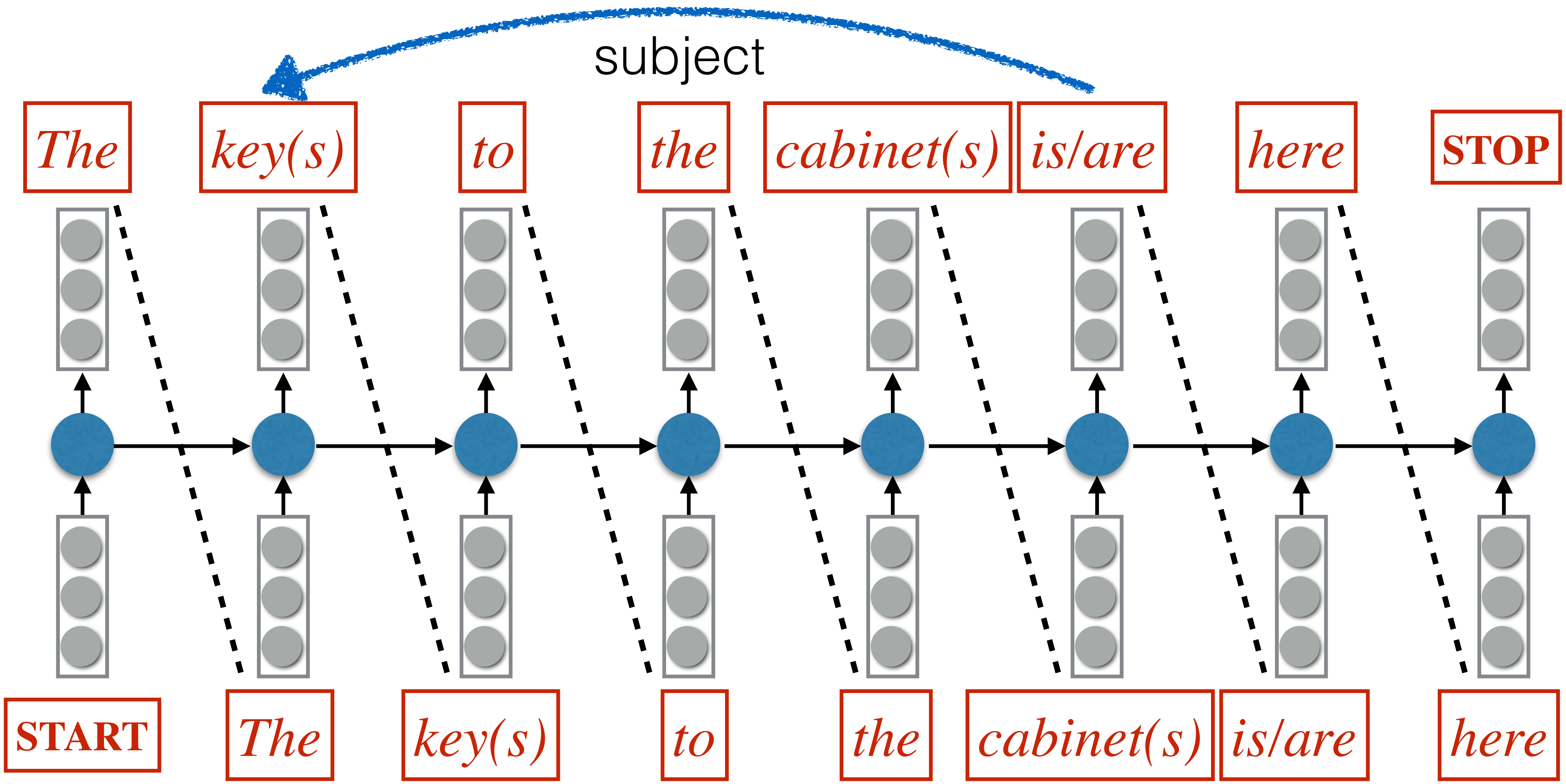
What do Neural Nets Learn about Linguistics?

What do Neural Nets Learn about Linguistics?



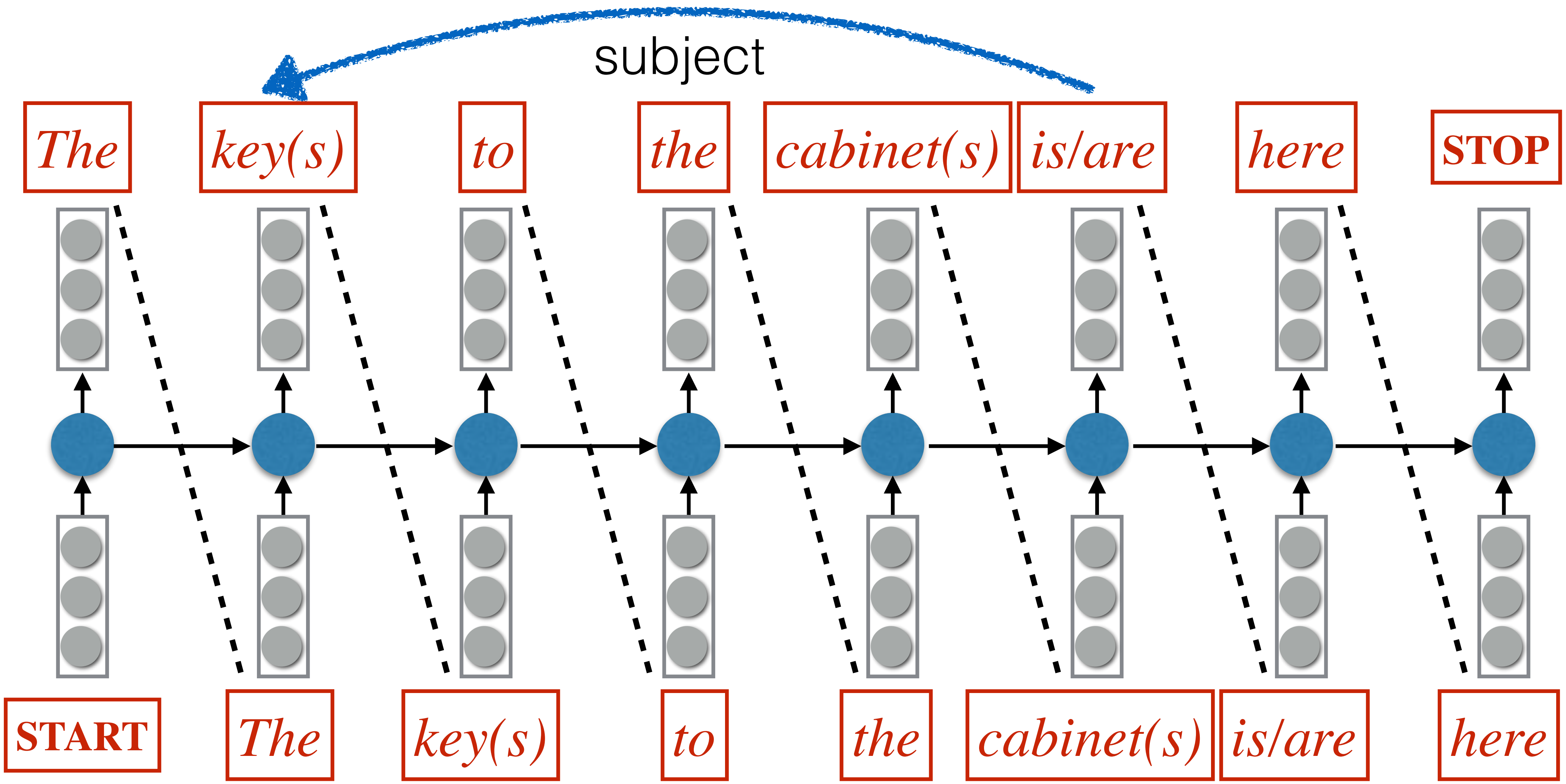
Linzen, Dupoux, and Goldberg (2017, *TACL*)

What do Neural Nets Learn about Linguistics?



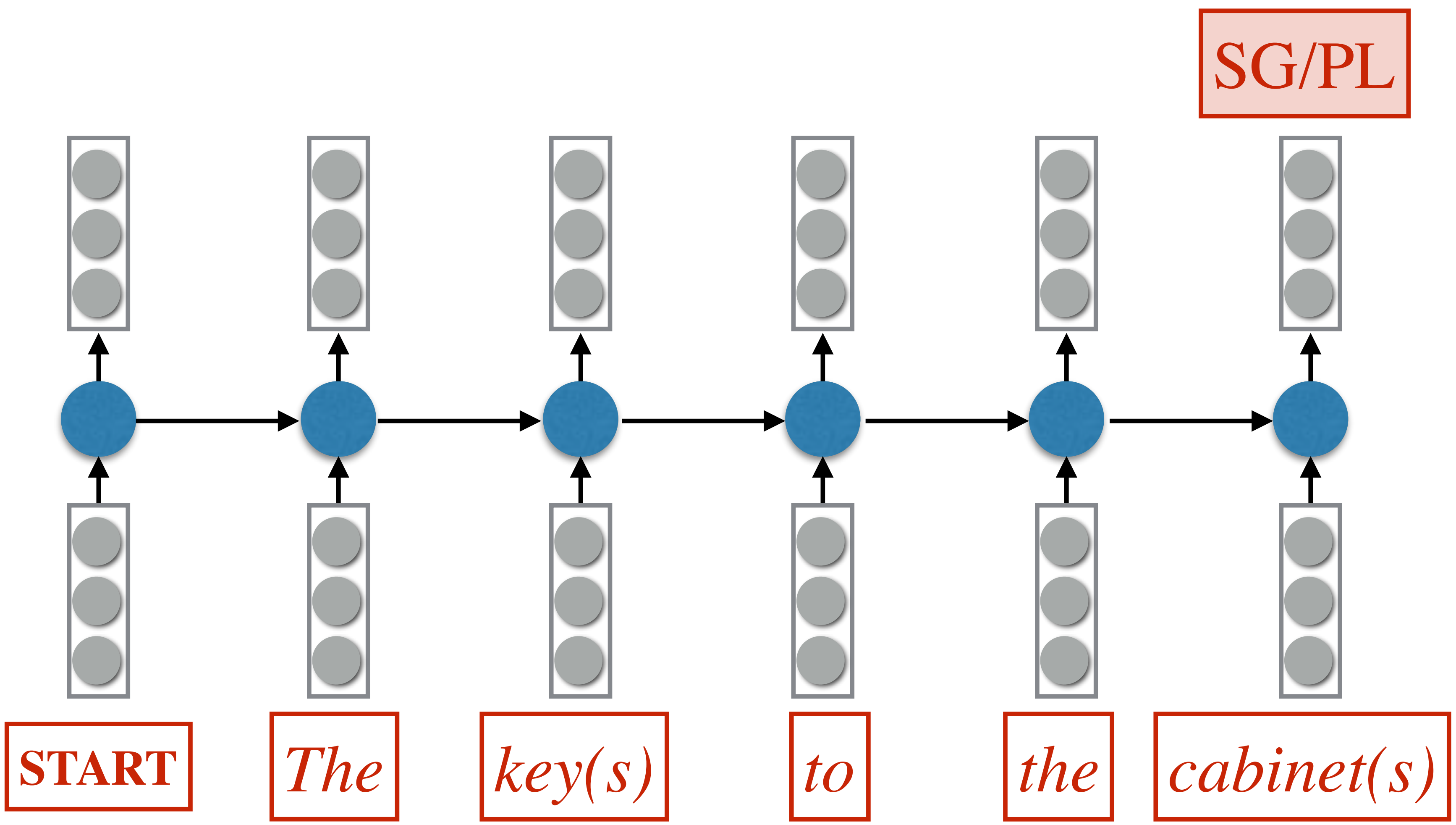
Linzen, Dupoux, and Goldberg (2017, *TACL*)

Experiment 1: Can a RNN learn syntax?



Linzen, Dupoux, and Goldberg (2017, *TACL*)

Experiment 1: Can a RNN learn syntax?

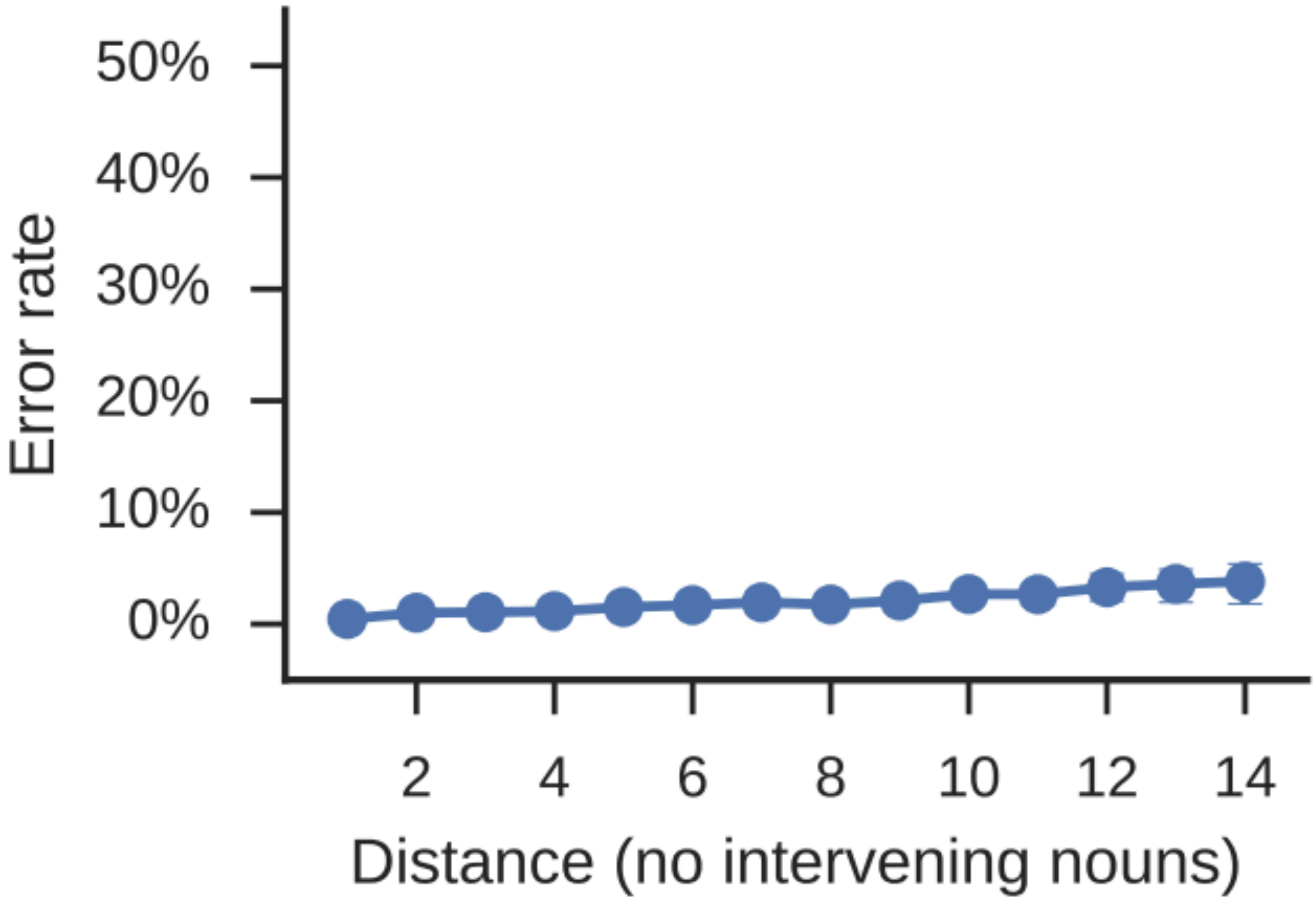


Linzen, Dupoux, and Goldberg (2017, *TACL*)

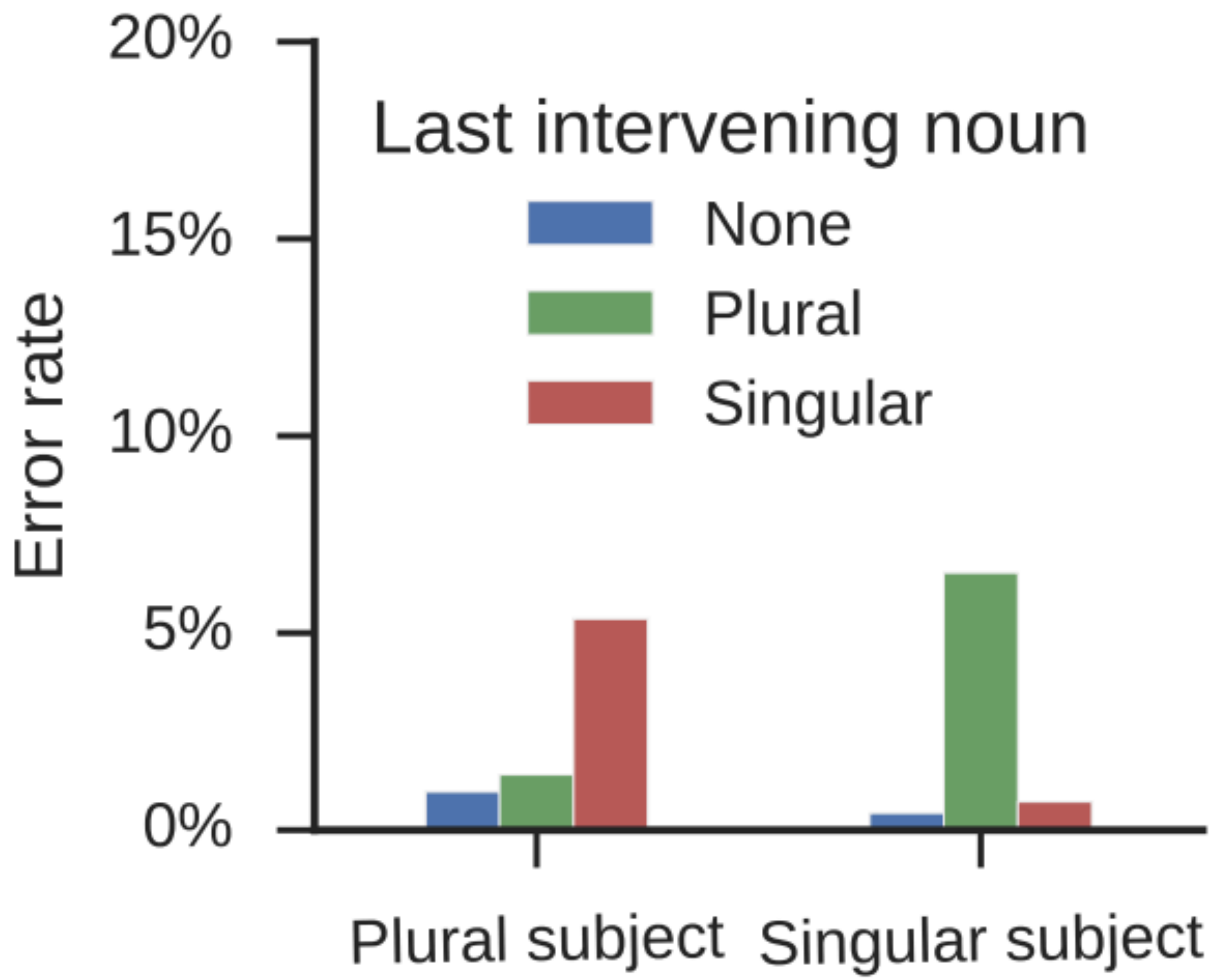
Experiment 1: Can a RNN learn syntax?

- This is a great set up!
  - To generate training data, we just need to be able to tag present tense verbs in a corpus
  - Authors used ~1.4M sentences from Wikipedia
  - To analyze, we might want a bit more of information about the sentences to know when the model gets it right and when it gets it wrong

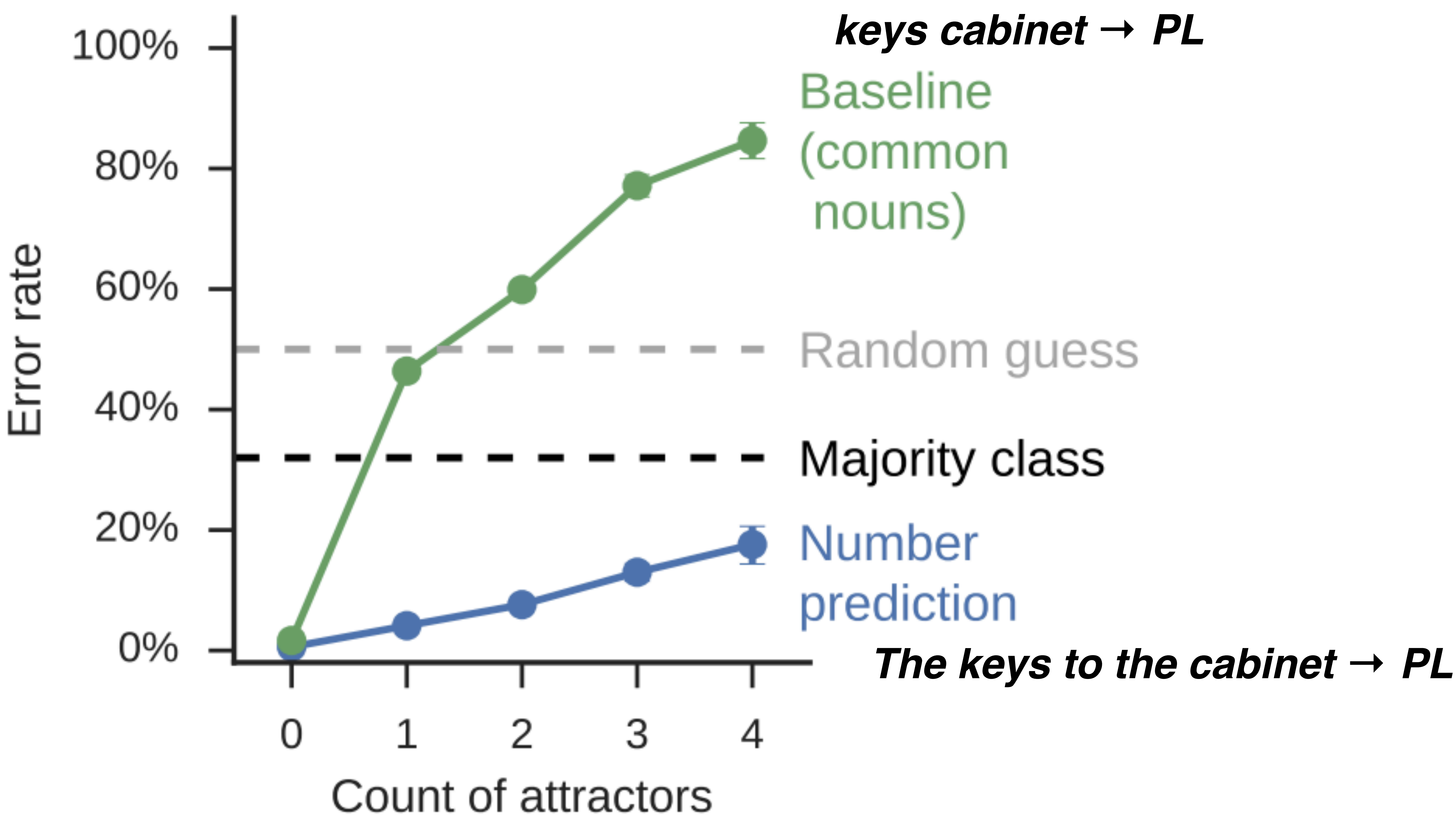
Experiment 1 Results



Experiment 1 Results



Experiment 1 Results



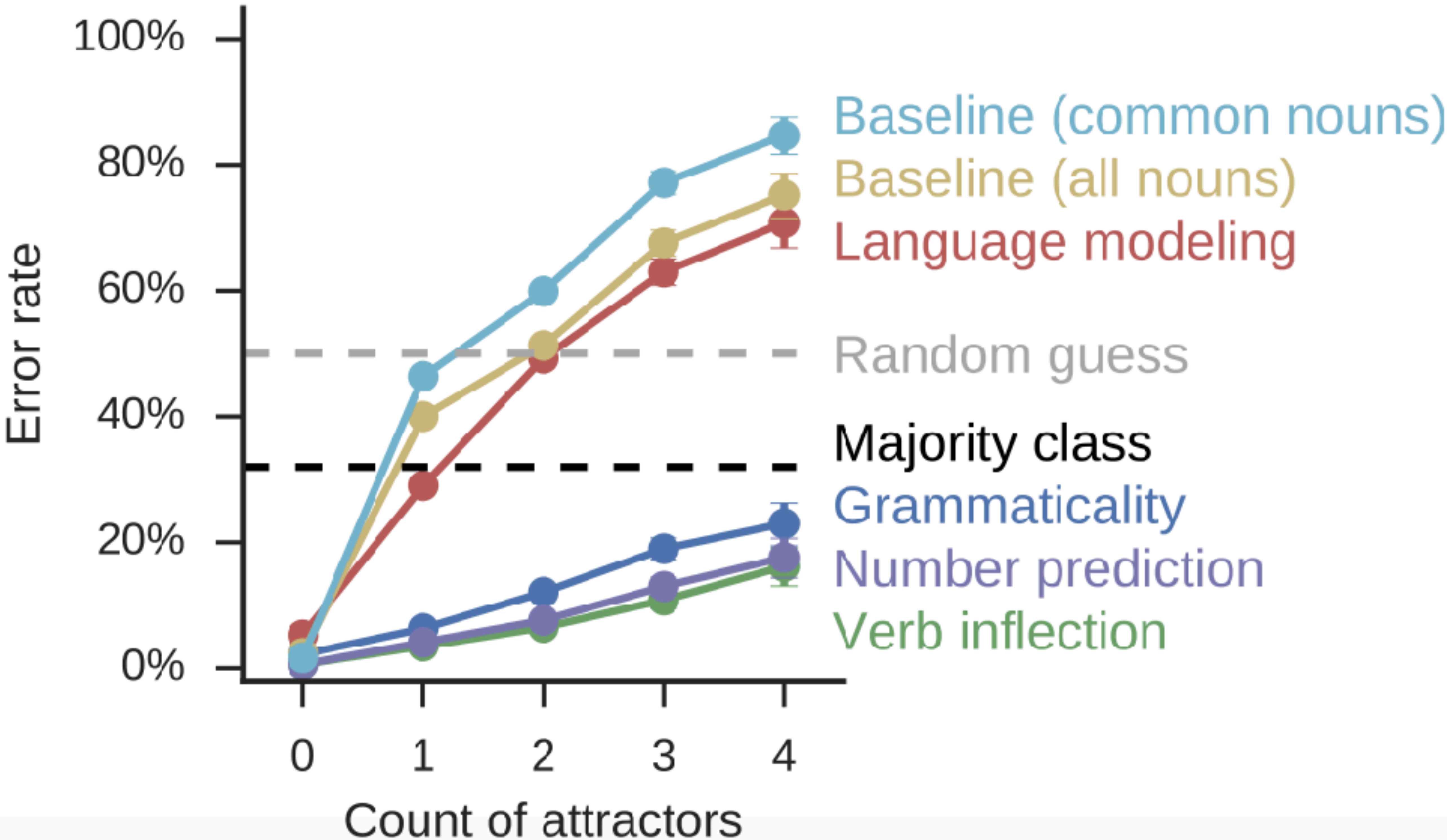
Experimental Variants

	Training signal	Evaluation Task
The keys to the cabinet	{SINGULAR, **PLURAL**}	P(PL) > P(SG)?
The keys to the cabinet is/are	{SINGULAR, **PLURAL**}	P(PL) > P(SG)?
The keys to the cabinet are here	{**GRAMMATICAL**, UNGRAMMATICAL}	P(GRAMMATICAL) > P(UNGRAMMATICAL)?
The keys to the cabinet	{**are**, is, cat, dog, the, …}	P(are) > P(is)?

Experimental Variants

	Training signal	Evaluation Task
The keys to the cabinet	{SINGULAR, **PLURAL**}	P(PL) > P(SG)?
The keys to the cabinet is/are	{SINGULAR, **PLURAL**}	P(PL) > P(SG)?
The keys to the cabinet are here	{**GRAMMATICAL**, UNGRAMMATICAL}	P(GRAMMATICAL) > P(UNGRAMMATICAL)?
The keys to the cabinet	{**are**, is, cat, dog, the, …}	P(are) > P(is)?

More Results



Summary

- RNN Language Models are not learning the correct generalizations about syntax
- Open questions
  - If RNNs are trained jointly to predict “singular/plural” and the next word, would they do better? [**Auxiliary objective**]
  - Would RNNGs do a better job on this task?
- Other experimental variants
  - Is there a “simple” function $f(\mathbf{h}_t) \rightarrow \{\mathrm{SG}, \mathrm{PL}\}$?
  - Is there a single dimension corresponding to “number”?

Linguistics in DL

- Two benefits:
  - Help us design better models based on knowledge about nature
  - Help us interrogate our models to see if they behave like they should
- Any questions?