Applied Natural Language Processing

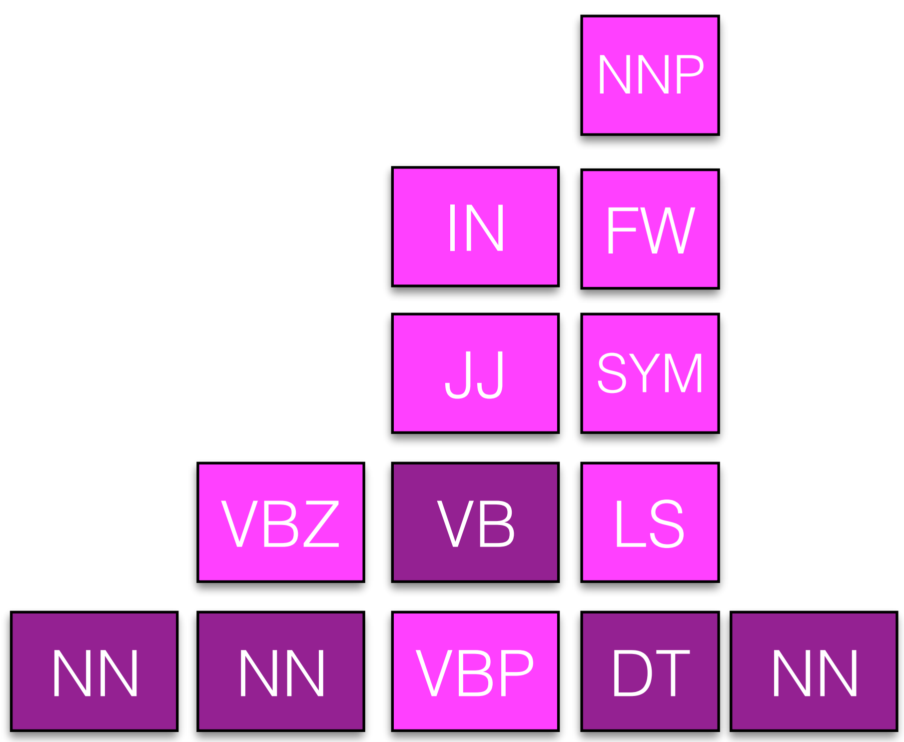
Info 256

Lecture 20: Sequence labeling (April 9, 2019)

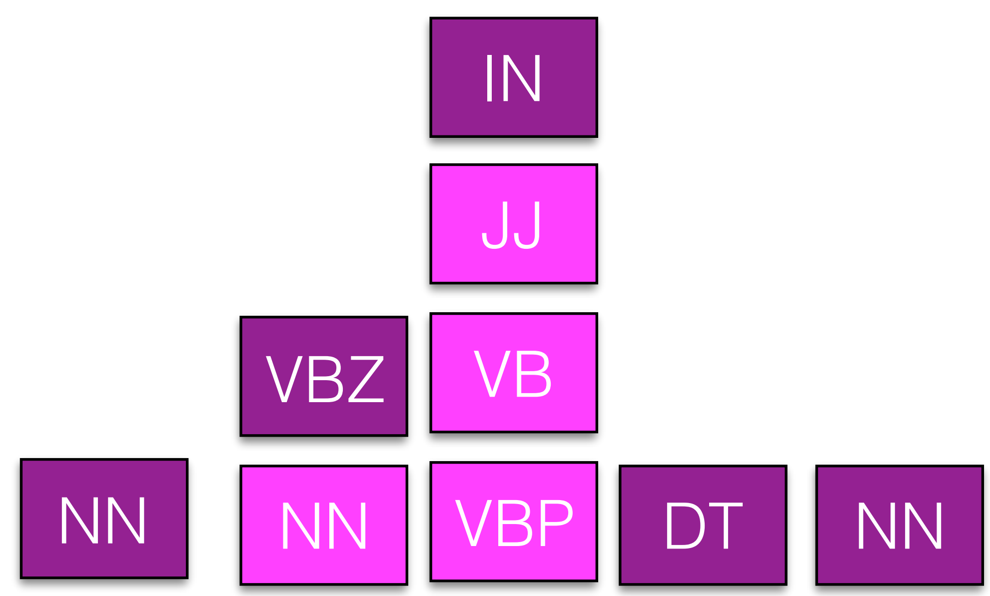
David Bamman, UC Berkeley

POS tagging

Labeling the tag that's correct for the context.



Fruit flies like a banana



Time flies like an arrow

(Just tags in evidence within the Penn Treebank — more are possible!)

Named entity recognition

B-PERS I-PERS O O O O ORG

tim cook is the ceo of apple

3 or 4-class:

- person
- location
- organization
- (misc)

7-class:

- person
- location
- organization
- time
- money
- percent
- date

Supersense tagging

O B-artifact I-artifact B-motion O B-time O O O B-group

The station wagons arrived at noon, a long shining line

O B-motion O O B-location I-location

that coursed through the west campus.

1	person	7	cognition	13	attribute	19	quantity	25	plant
2	communication	8	possession	14	object	20	motive	26	relation
3	artifact	9	location	15	process	21	animal		
4	act	10	substance	16	Tops	22	body		
5	group	11	state	17	phenomenon	23	feeling		
6	food	12	time	18	event	24	shape		

Noun supersenses (Ciarmita and Altun 2003)

Segmentation

- B = character is the start of new word
- I = character is inside existing word

#	b	l	a	c	k	l	i	v	e	s	m	a	t	t	e	r	
B	B	I	I	I	I	B	I	I	I	I	B	I	I	I	I	I	# black lives matter
B	B	I	I	I	I	B	I	I	I	B	I	I	I	I	I	I	# black live smatter

Sequence labeling

$$x = \{x_1, \ldots, x_n\}$$

$$y = \{y_1, \ldots, y_n\}$$

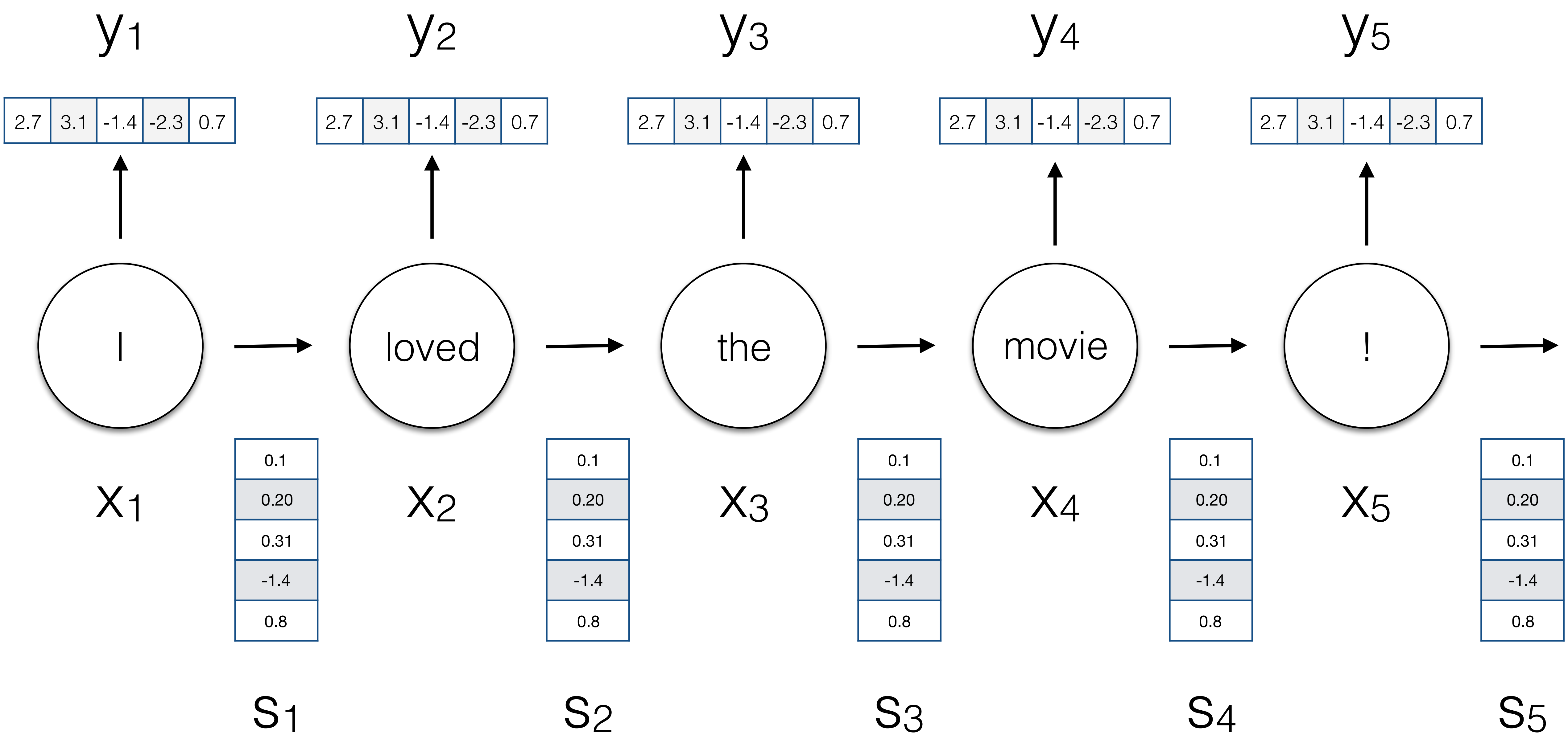
- For a set of inputs x with n sequential time steps, one corresponding label y_i for each x_i

Sequence labeling models

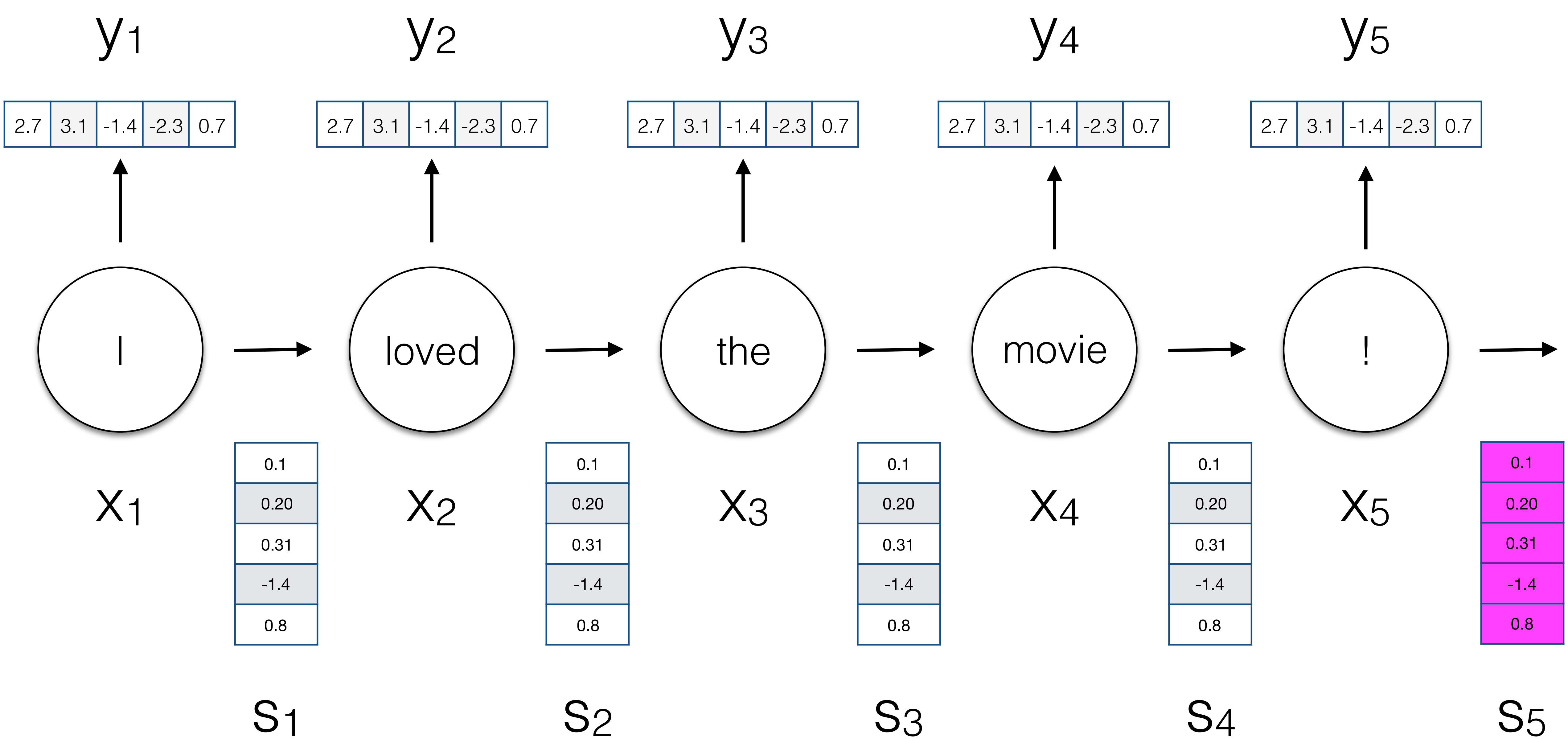
model	form	label dependency	rich features?
Hidden Markov Models	$\prod_{i=1}^{N} P(x_i \mid y_i)\, P(y_i \mid y_{i-1})$	Markov assumption	no
MEMM	$\prod_{i=1}^{N} P(y_i \mid y_{i-1}, x, \beta)$	Markov assumption	yes
CRF	$P(y \mid x, \beta)$	pairwise through entire sequence	yes
RNN	$\prod_{i=1}^{N} P(y_i \mid x_{1:i}, \beta)$	none	distributed

Back to RNNs

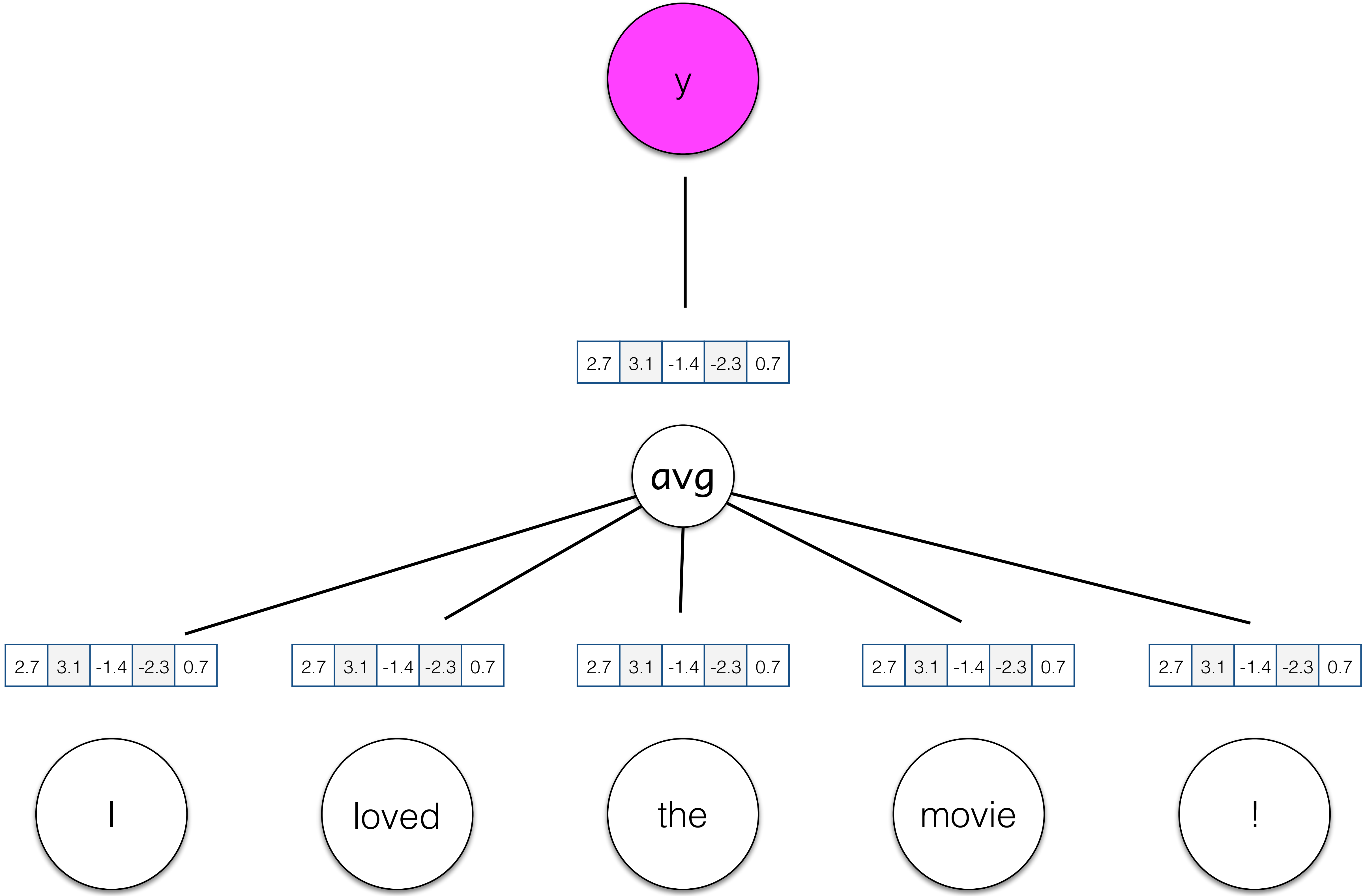
- RNN allow arbitarily-sized conditioning contexts; condition on the entire sequence history.
- We used RNNs for document classification to generate a representation of a sequence that we can then use for prediction.



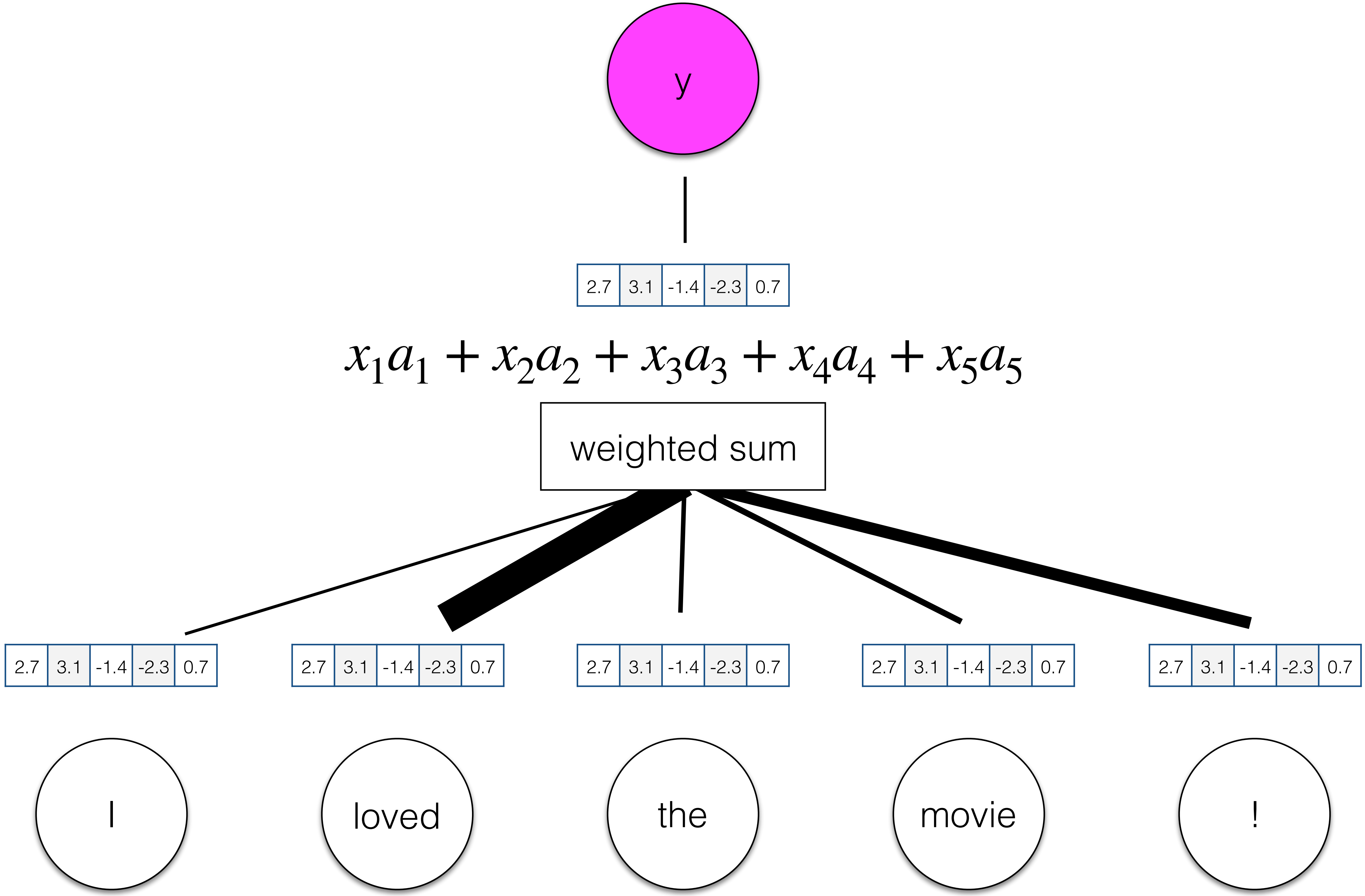
y_1
y_2
y_3
y_4
y_5
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!
x_1
x_2
x_3
x_4
x_5
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
s_1
s_2
s_3
s_4
s_5



y_1
y_2
y_3
y_4
y_5
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!
x_1
x_2
x_3
x_4
x_5
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
0.1 0.20 0.31 -1.4 0.8
s_1
s_2
s_3
s_4
s_5

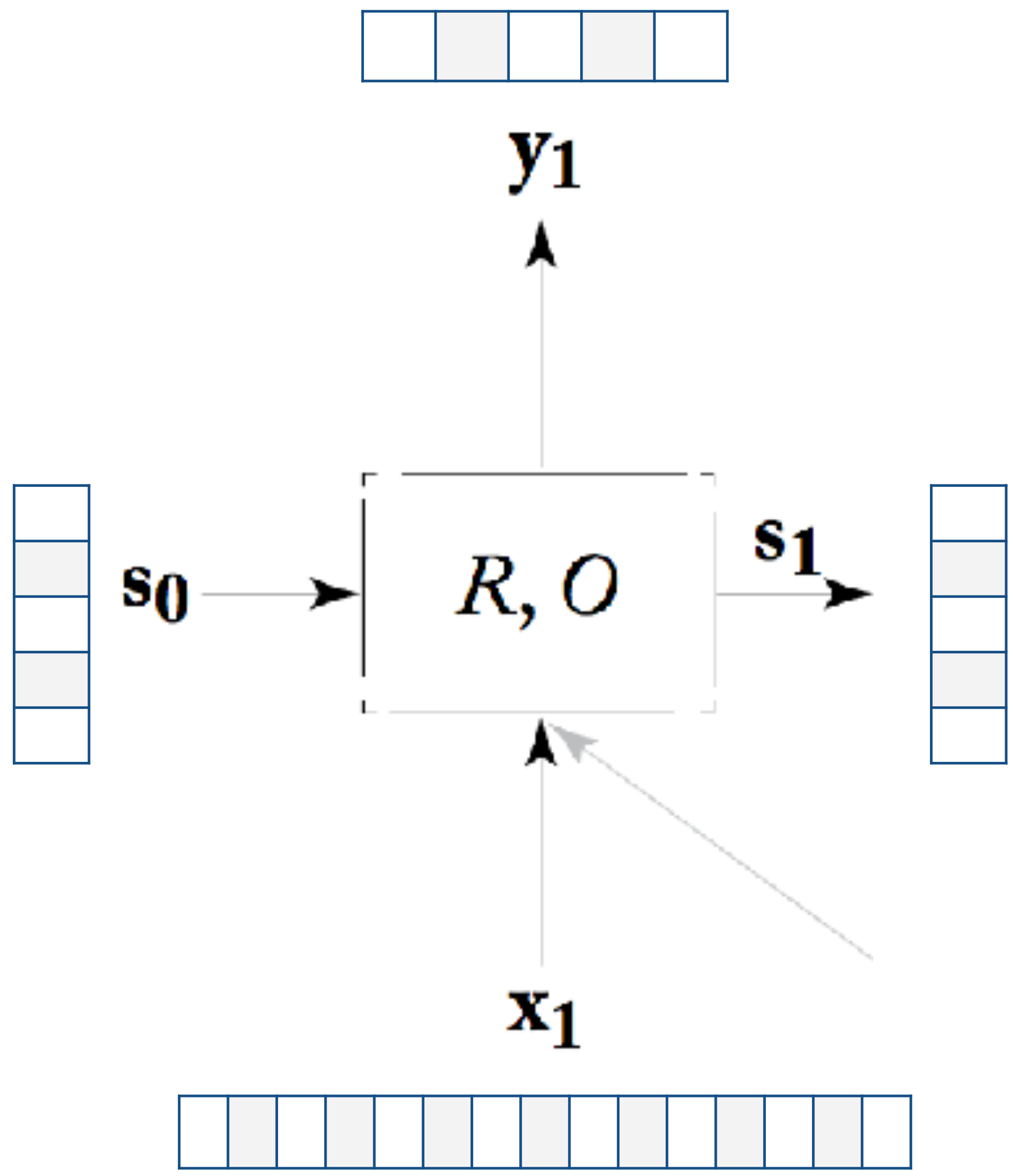


Iyyer et al. (2015), "Deep Unordered Composition Rivals Syntactic Methods for Text Classification" (ACL)


y
2.7 3.1 -1.4 -2.3 0.7
$x_1a_1 + x_2a_2 + x_3a_3 + x_4a_4 + x_5a_5$
weighted sum
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!

Recurrent neural network

- Each time step has two inputs:
  - x_i (the observation at time step i); one-hot vector, feature vector or word embedding.
  - s_{i-1} (the output of the previous state); base case: s_0 = 0 vector



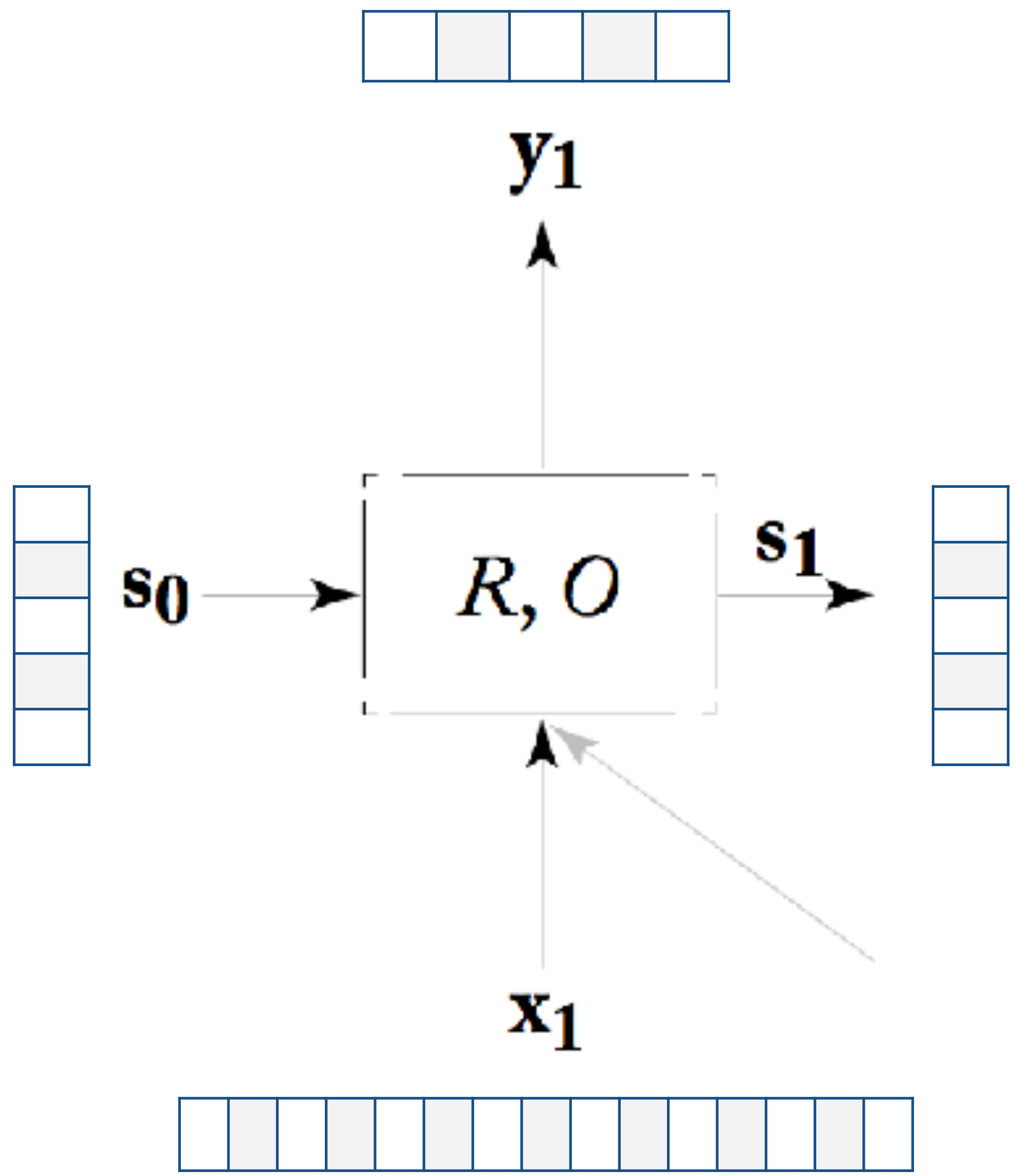
Recurrent neural network

$$s_i = R(x_i, s_{i-1})$$

R computes the output state as a function of the current input and previous state

$$y_i = O(s_i)$$

O computes the output as a function of the current output state



“Simple” RNN

g = tanh or relu

$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$$

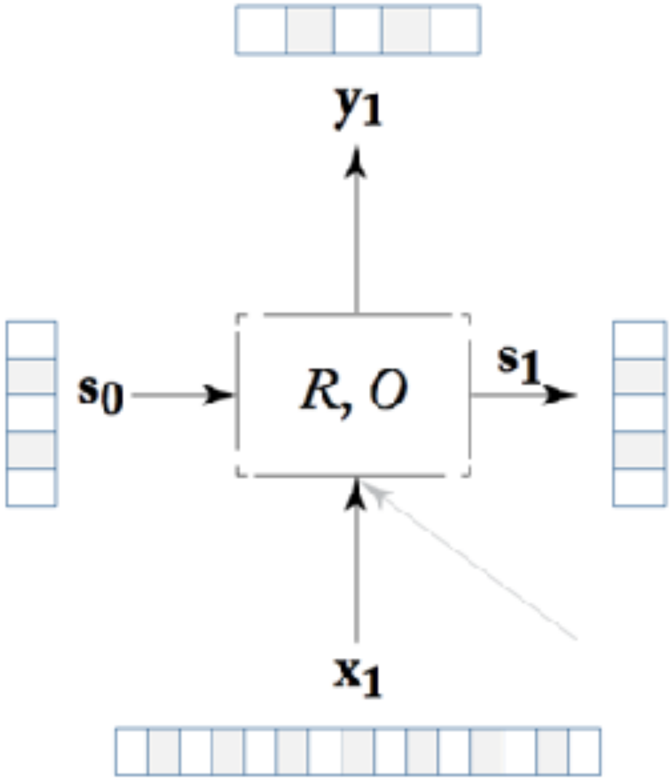
Different weight vectors W transform the previous state and current input before combining

$$W^s \in \mathbb{R}^{H \times H}$$

$$W^x \in \mathbb{R}^{D \times H}$$

$$b \in \mathbb{R}^H$$

$$y_i = O(s_i) = s_i$$



Elman 1990, Mikolov 2012

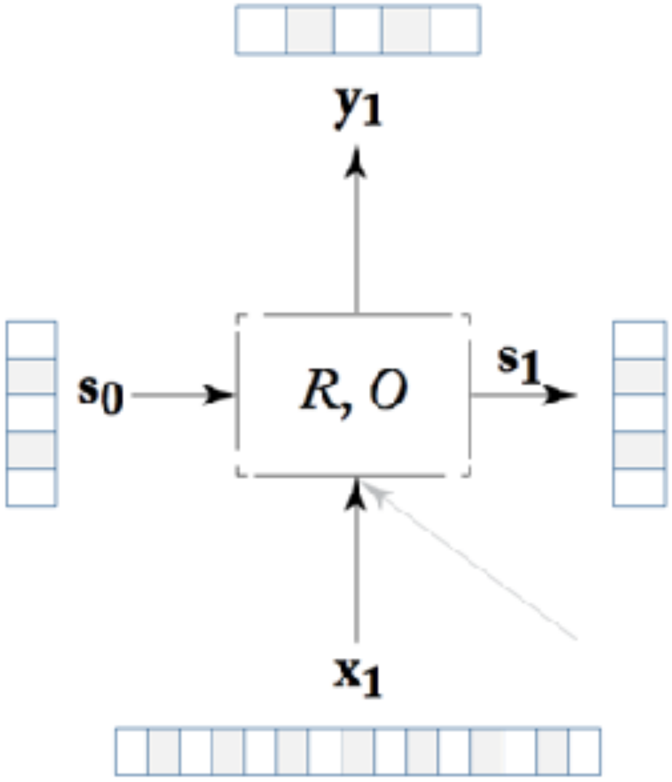
Recurrent neural network

- Often used for sequential prediction tasks:
  - Language models—predicting the next symbol (word, character) in a sequence
  - Machine translation—-predicting a sequence of words (sentence) in language *f* conditioned on sentence in language *e*
  - Sequence labeling (POS tagging, NER)

RNNs for sequence labeling

- The output state s_i is an H-dimensional real vector; we can transfer that into a probability by passing it through an additional linear transformation followed by a softmax

$$y_i = O(s_i) = \text{softmax}(s_i W^o + b^o)$$



Elman 1990, Mikolov 2012

Training RNNs

- Given this definition of an RNN:

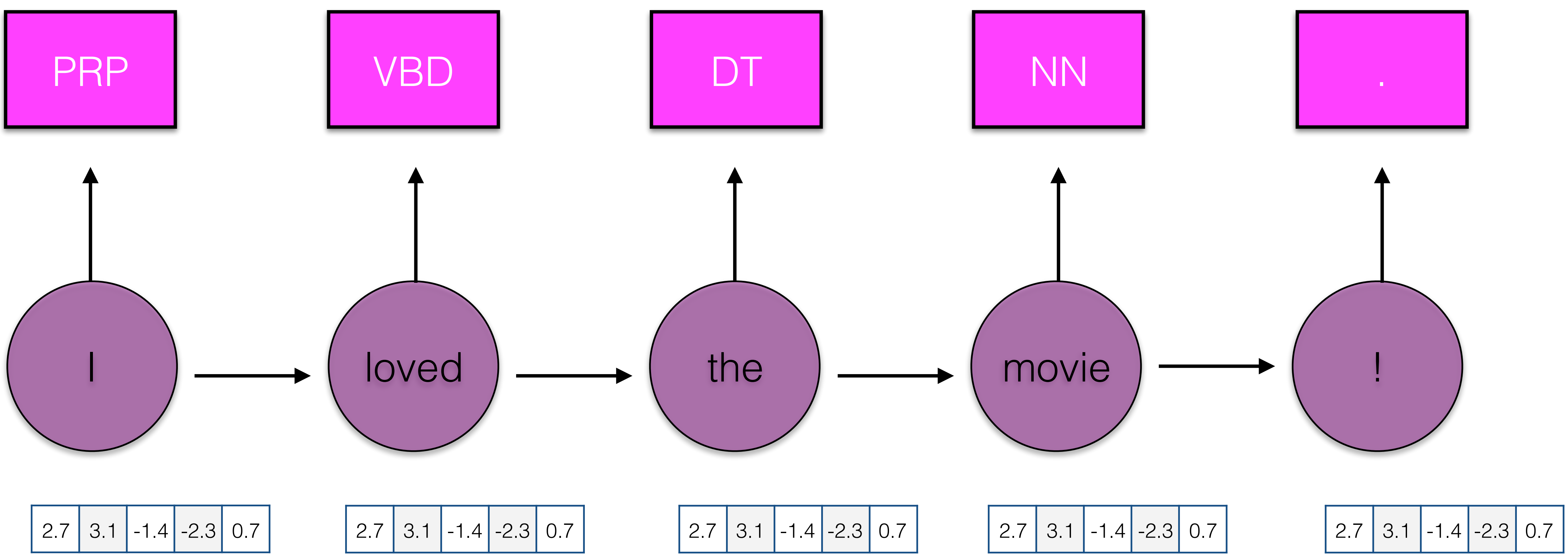
$$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_i W^x + b)$$

$$y_i = O(s_i) = \text{softmax}(s_i W^o + b^o)$$

- We have five sets of parameters to learn:

$$W^s, W^x, W^o, b, b^o$$

For POS tagging, predict the tag from $\boldsymbol{y}$ conditioned on the context



RNNs for POS

DT	NN	VBD	IN	DT	NN	???
The	horse	raced	past	the	barn	fell

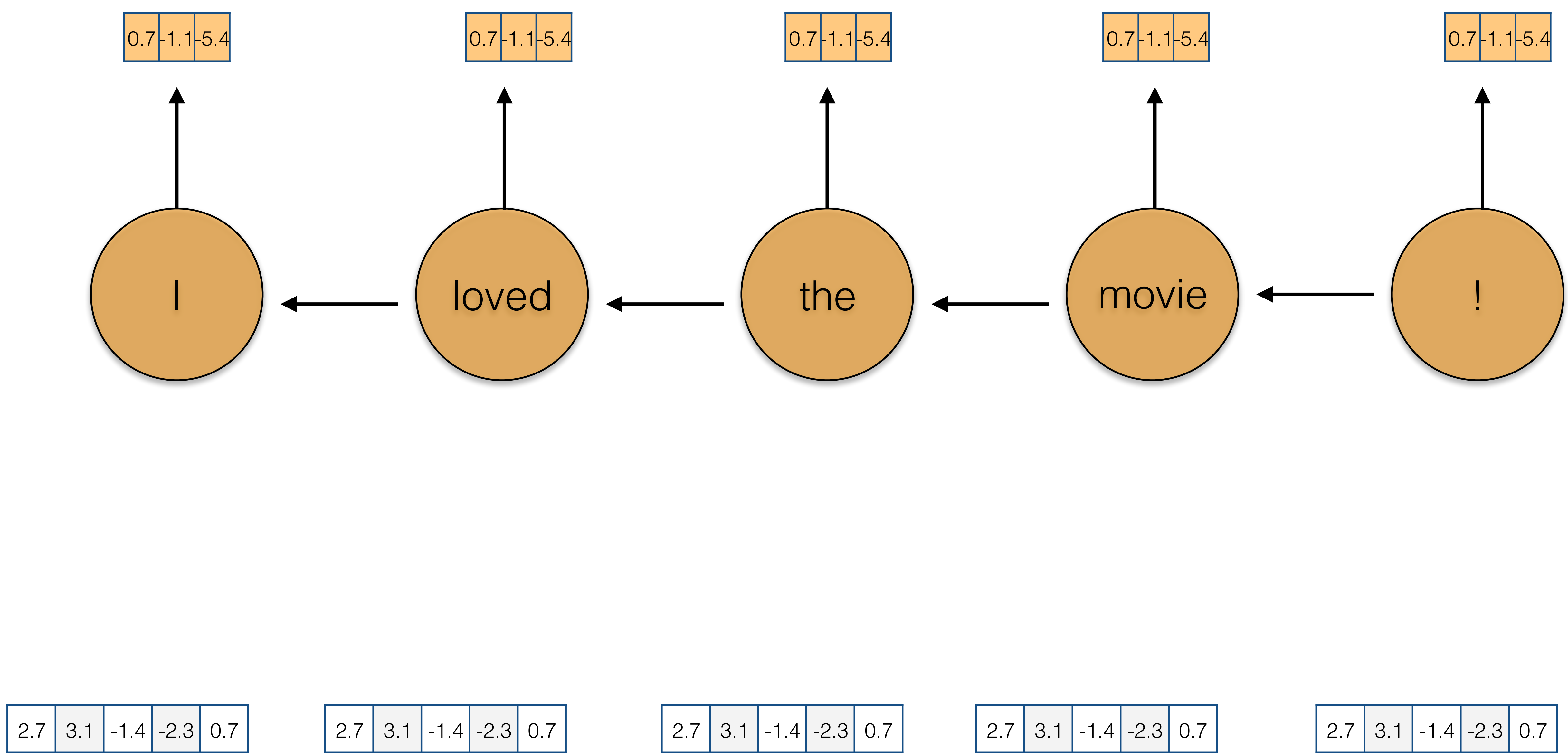
- To make a prediction for y_t, RNNs condition on all input seen through time t (x_1, …, x_t)
- But knowing something about the future can help (x_{t+1}, …, x_n)

Bidirectional RNN

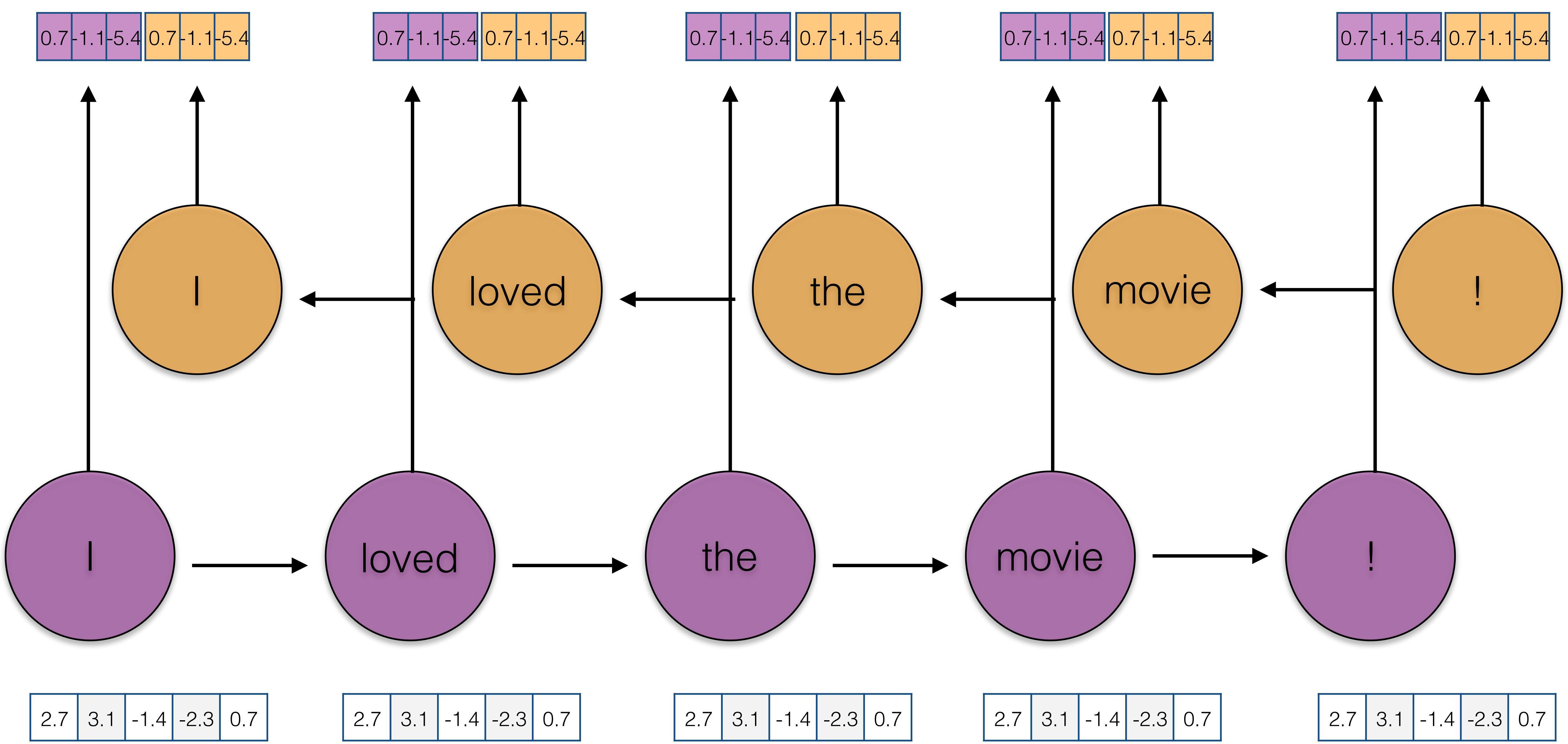
- A powerful alternative is make predictions conditioning both on the past and the future.
- Two RNNs
  - One running left-to-right
  - One right-to-left
- Each produces an output vector at each time step, which we concatenate

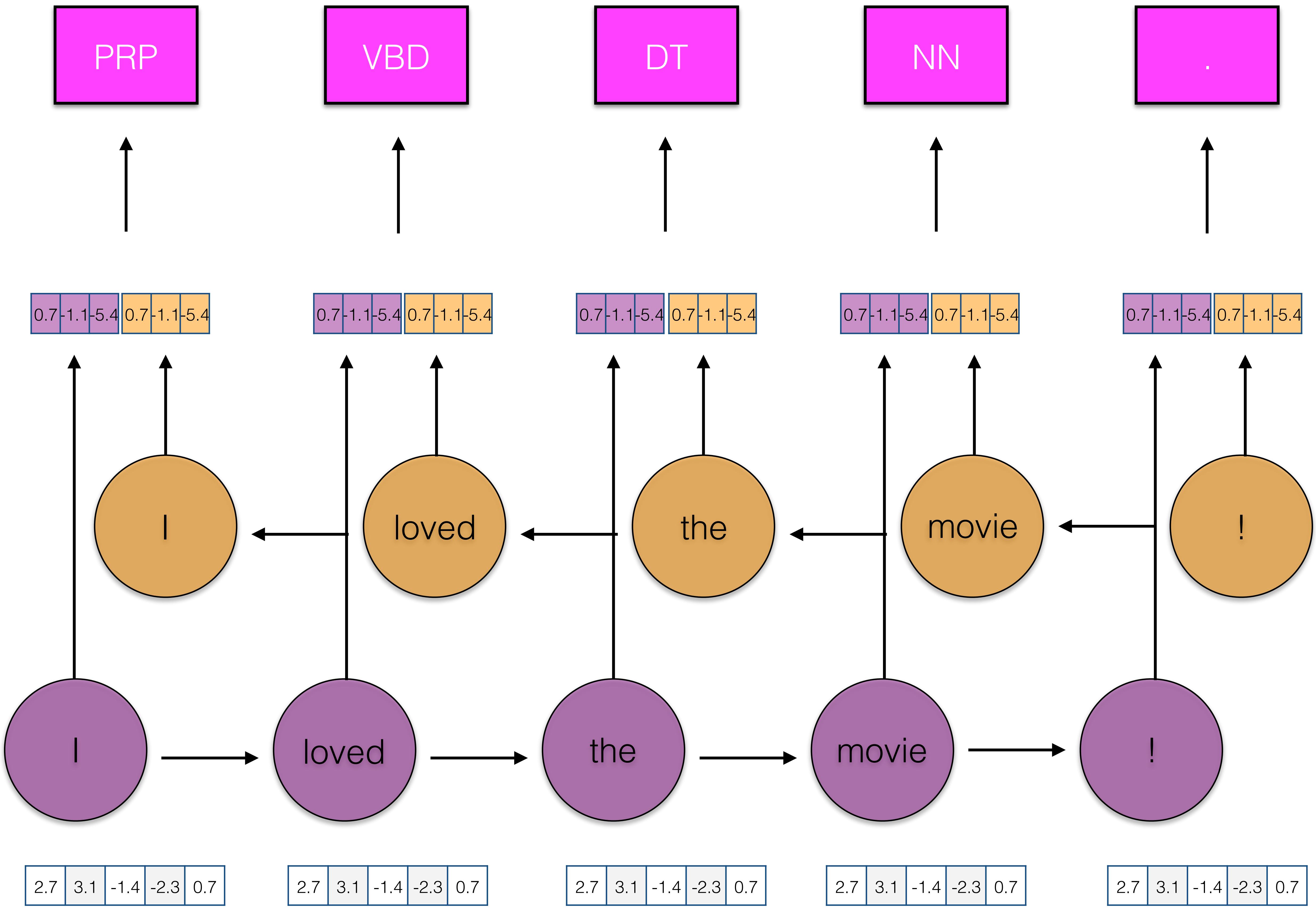
Bidirectional RNN

backward RNN



Bidirectional RNN





PRP
VBD
DT
NN
.
0.7 -1.1 -5.4 0.7 -1.1 -5.4
0.7 -1.1 -5.4 0.7 -1.1 -5.4
0.7 -1.1 -5.4 0.7 -1.1 -5.4
0.7 -1.1 -5.4 0.7 -1.1 -5.4
0.7 -1.1 -5.4 0.7 -1.1 -5.4
I
loved
the
movie
!
I
loved
the
movie
!
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7

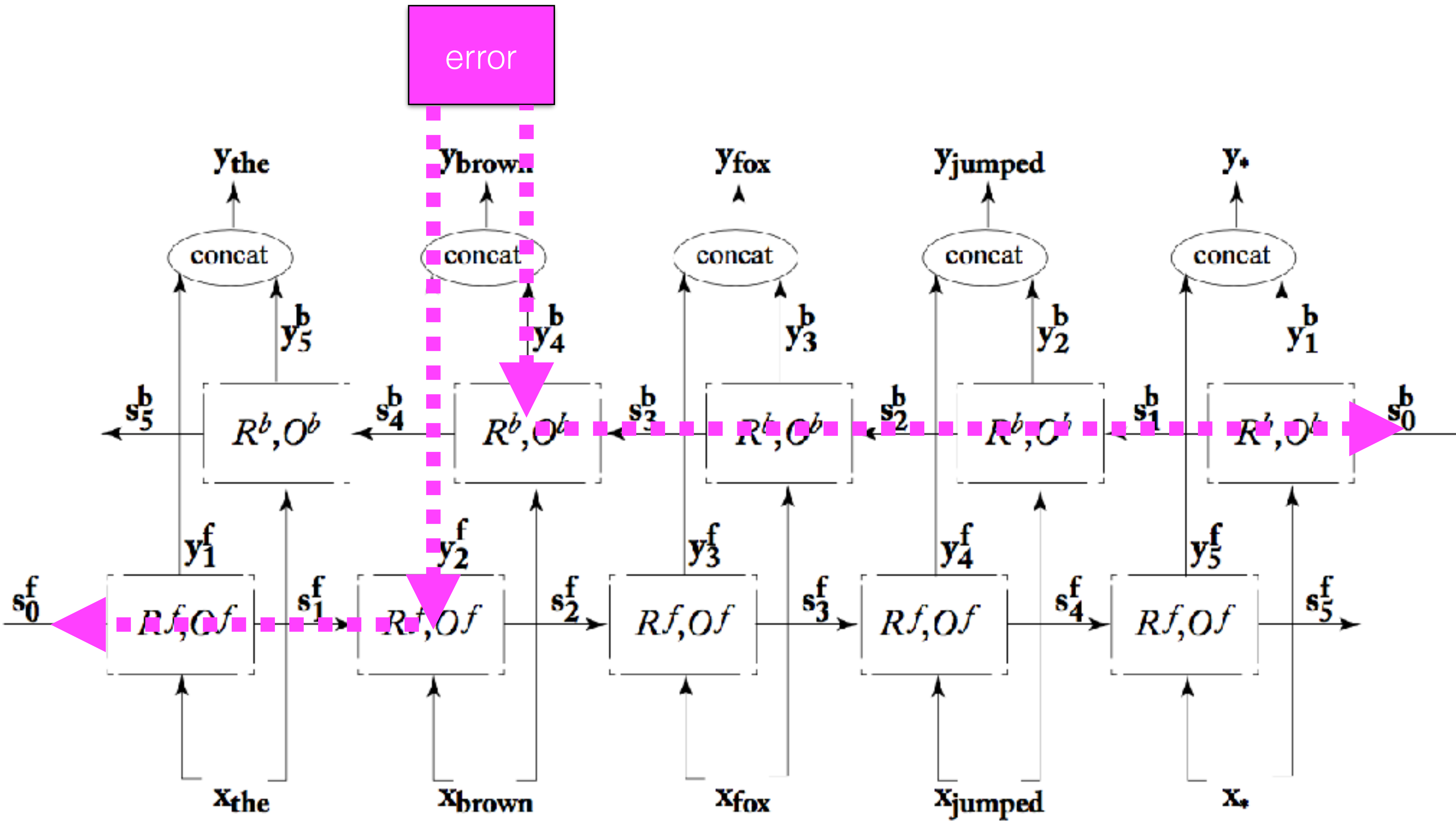
Training BiRNNs

- Given this definition of an BiRNN:

$$s_b^i = R_b(x^i, s_b^{i+1}) = g(s_b^{i+1} W_b^s + x^i W_b^x + b_b)$$
$$s_f^i = R_f(x^i, s_f^{i-1}) = g(s_f^{i-1} W_f^s + x^i W_f^x + b_f)$$

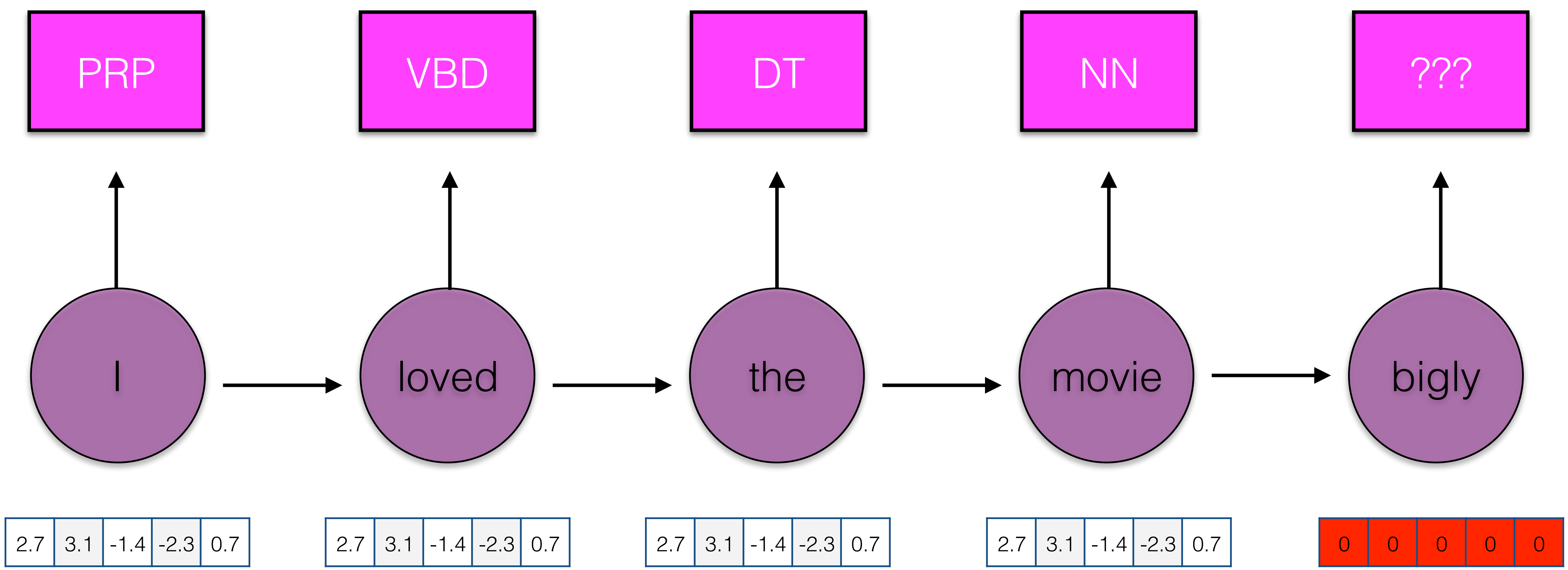
$$y_i = \mathrm{softmax}\left([s_f^i; s_b^i] W^o + b^o\right)$$

- We have 8 sets of parameters to learn (3 for each RNN + 2 for the final layer)



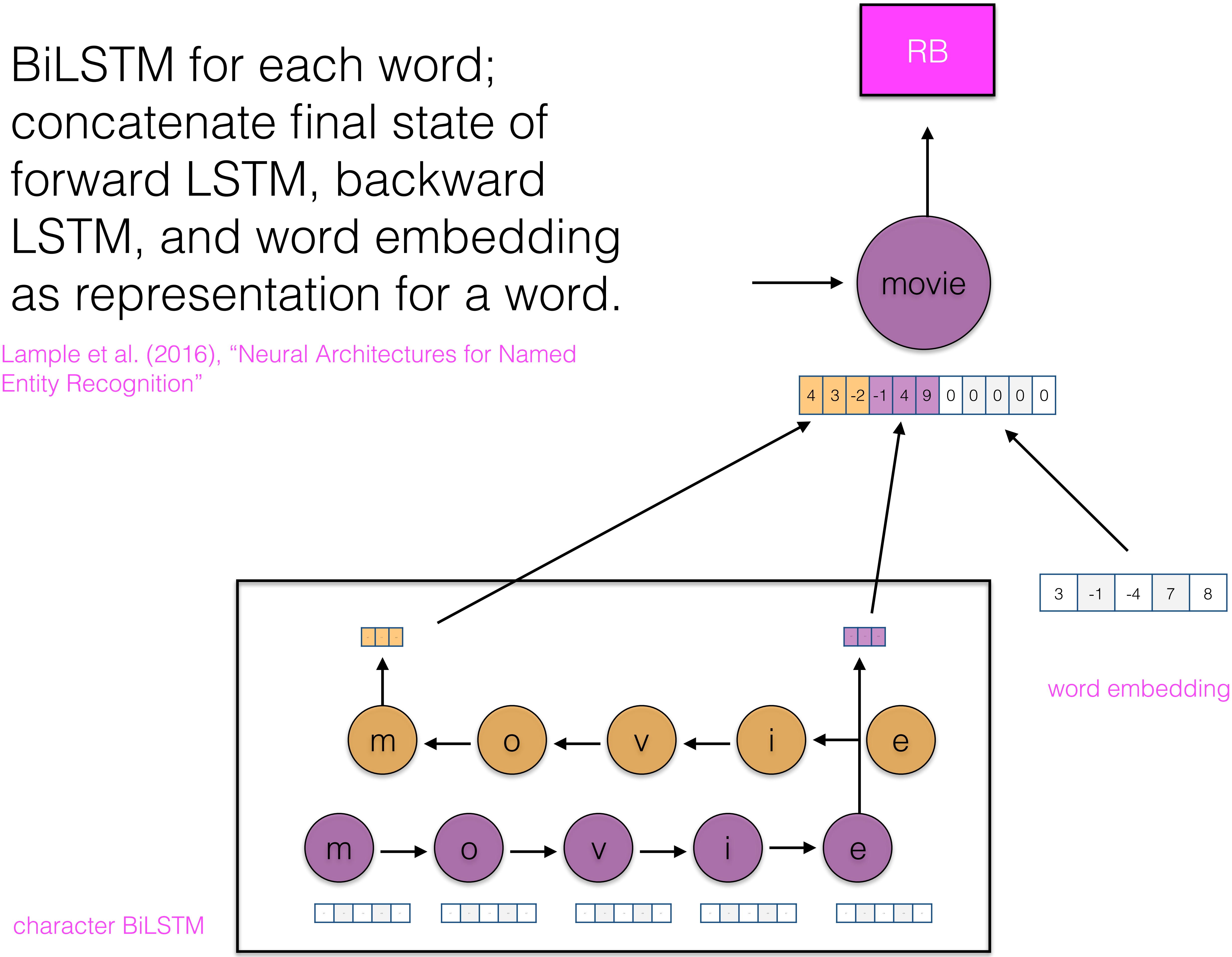
Goldberg 2017

How do we fix this?



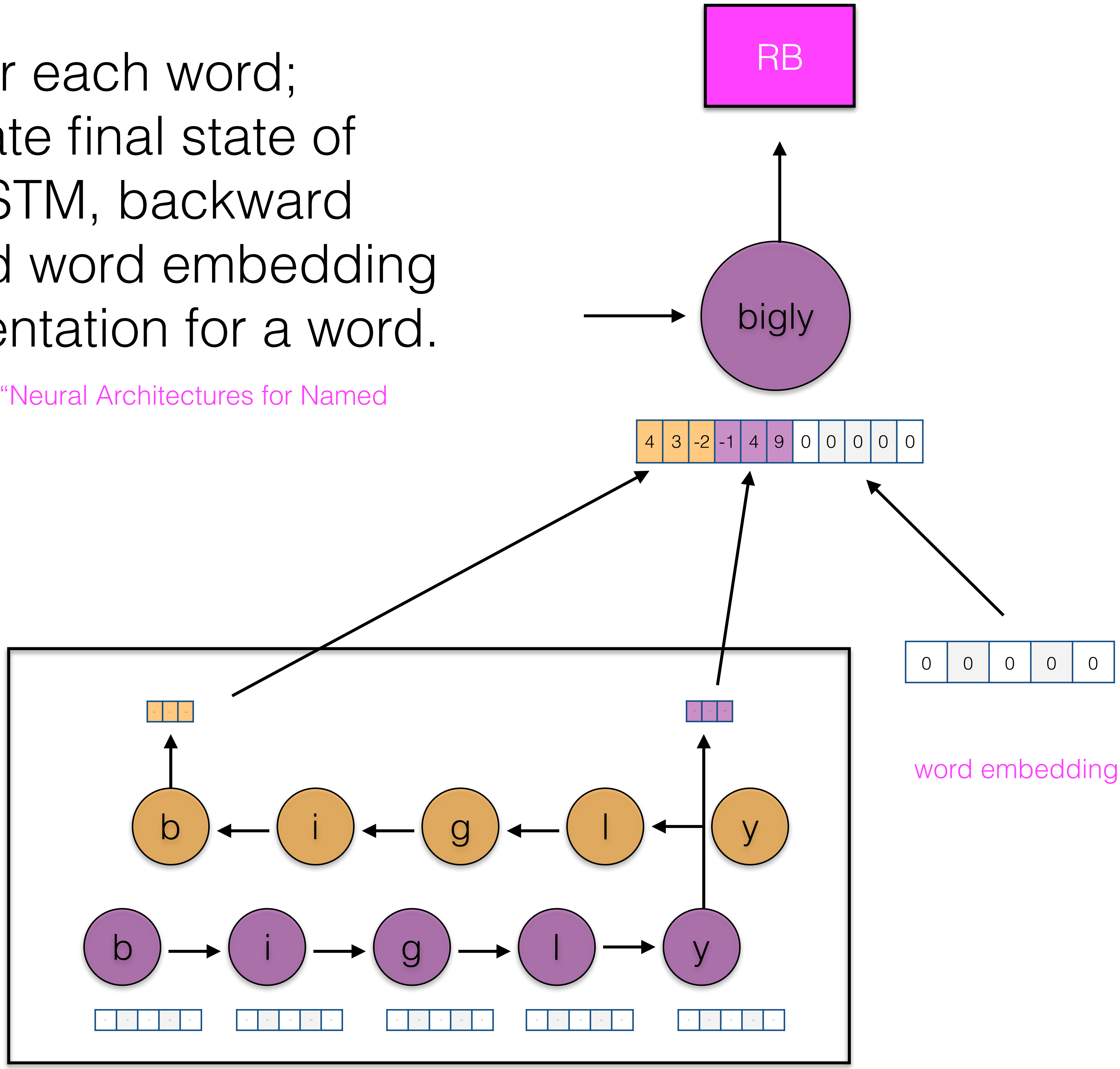
BiLSTM for each word; concatenate final state of forward LSTM, backward LSTM, and word embedding as representation for a word.

Lample et al. (2016), “Neural Architectures for Named Entity Recognition”



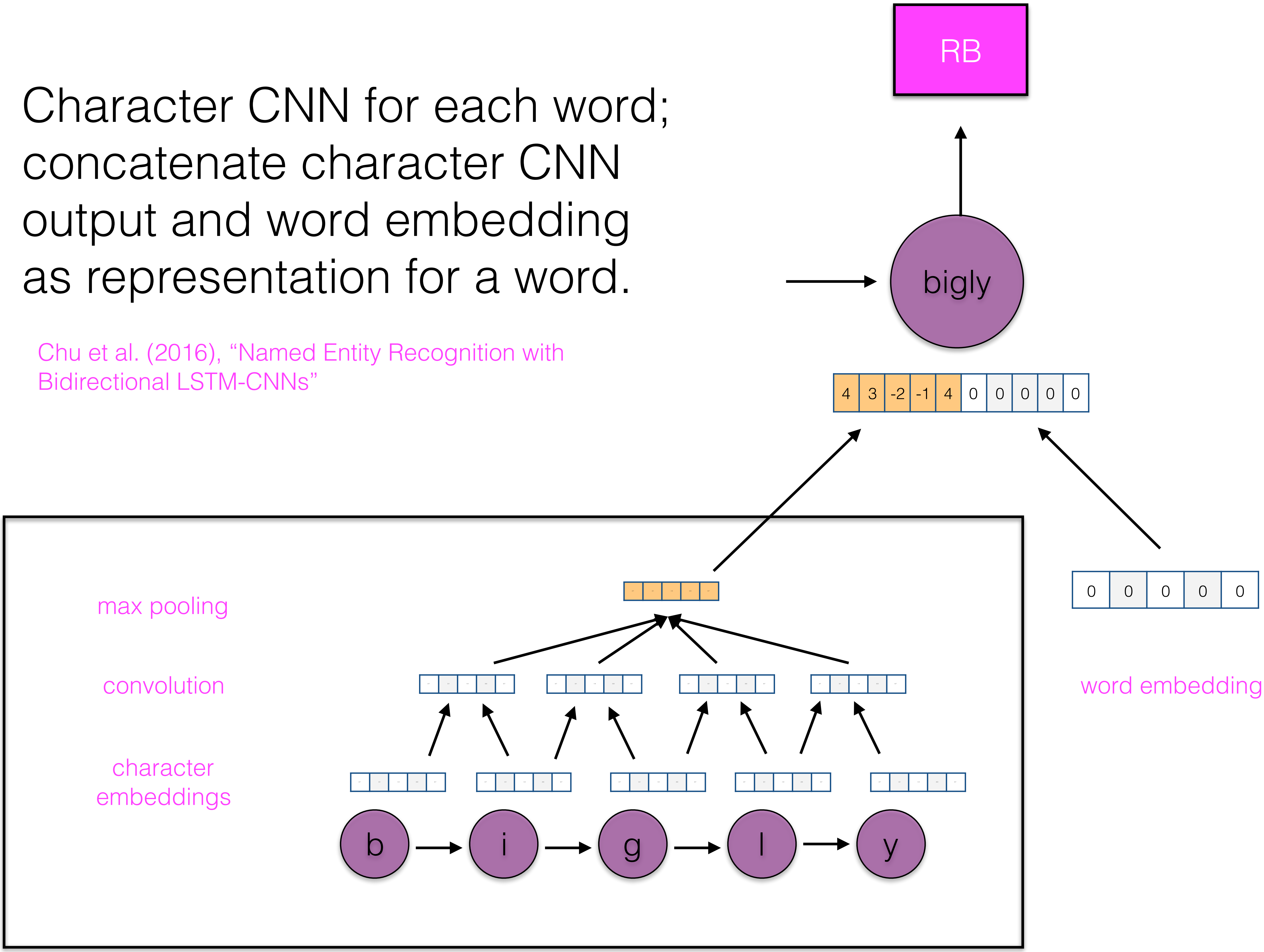
BiLSTM for each word; concatenate final state of forward LSTM, backward LSTM, and word embedding as representation for a word.

Lample et al. (2016), "Neural Architectures for Named Entity Recognition"



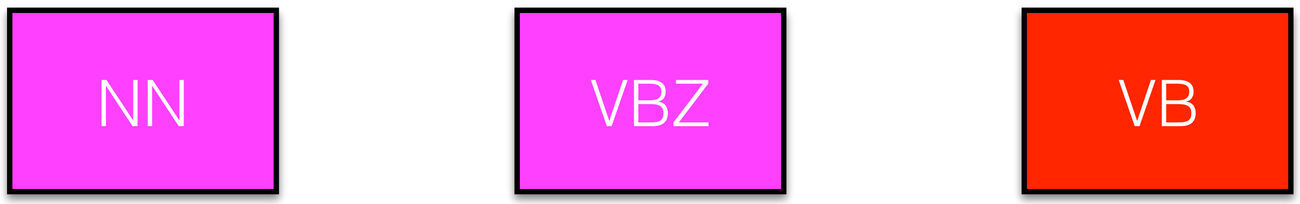
Character CNN for each word; concatenate character CNN output and word embedding as representation for a word.

Chu et al. (2016), “Named Entity Recognition with Bidirectional LSTM-CNNs”



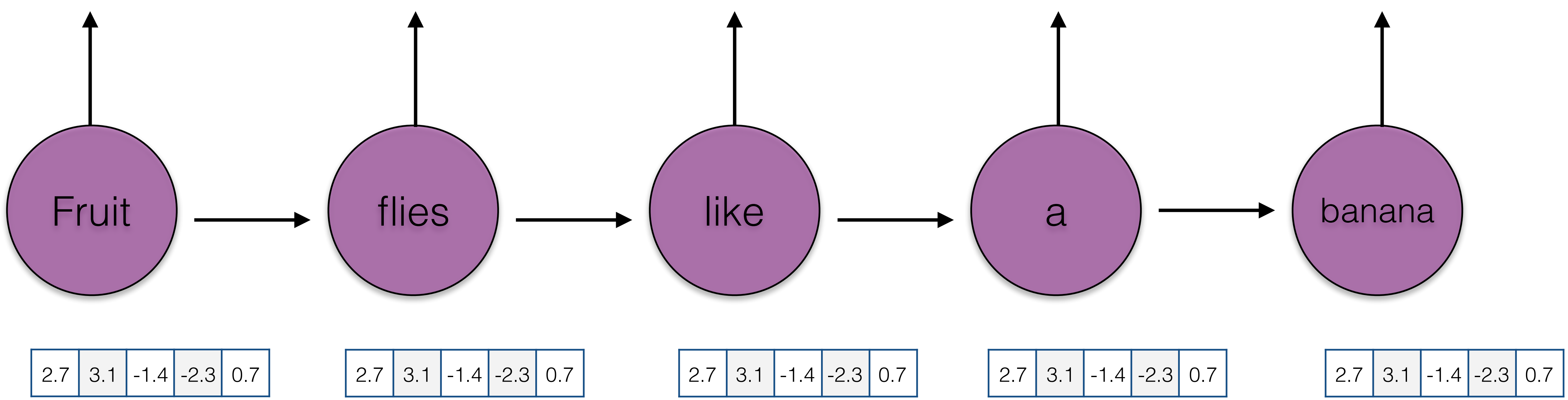
LSTM/RNN

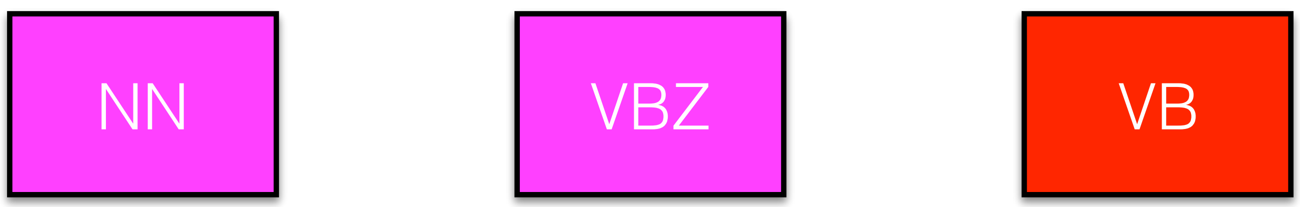
- An RNN doesn't use the dependencies between nearby labels in making predictions.



VBZ	0.51
NNS	0.48
JJ	0.01
NN	0
…	…

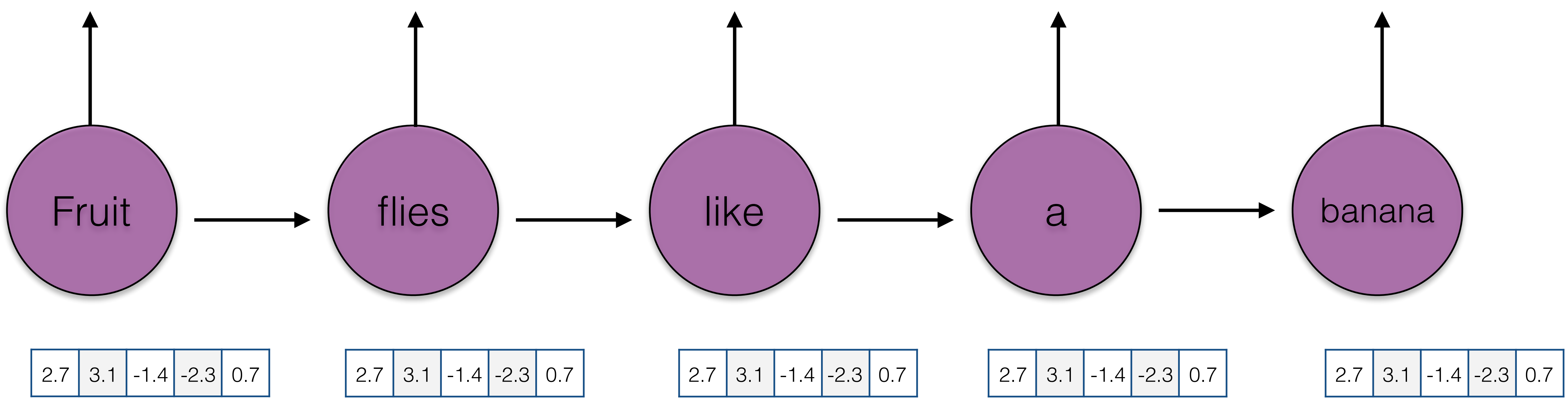
The information that's passed between states is not the categorical choice (VBZ) but a hidden state that generated the distribution.





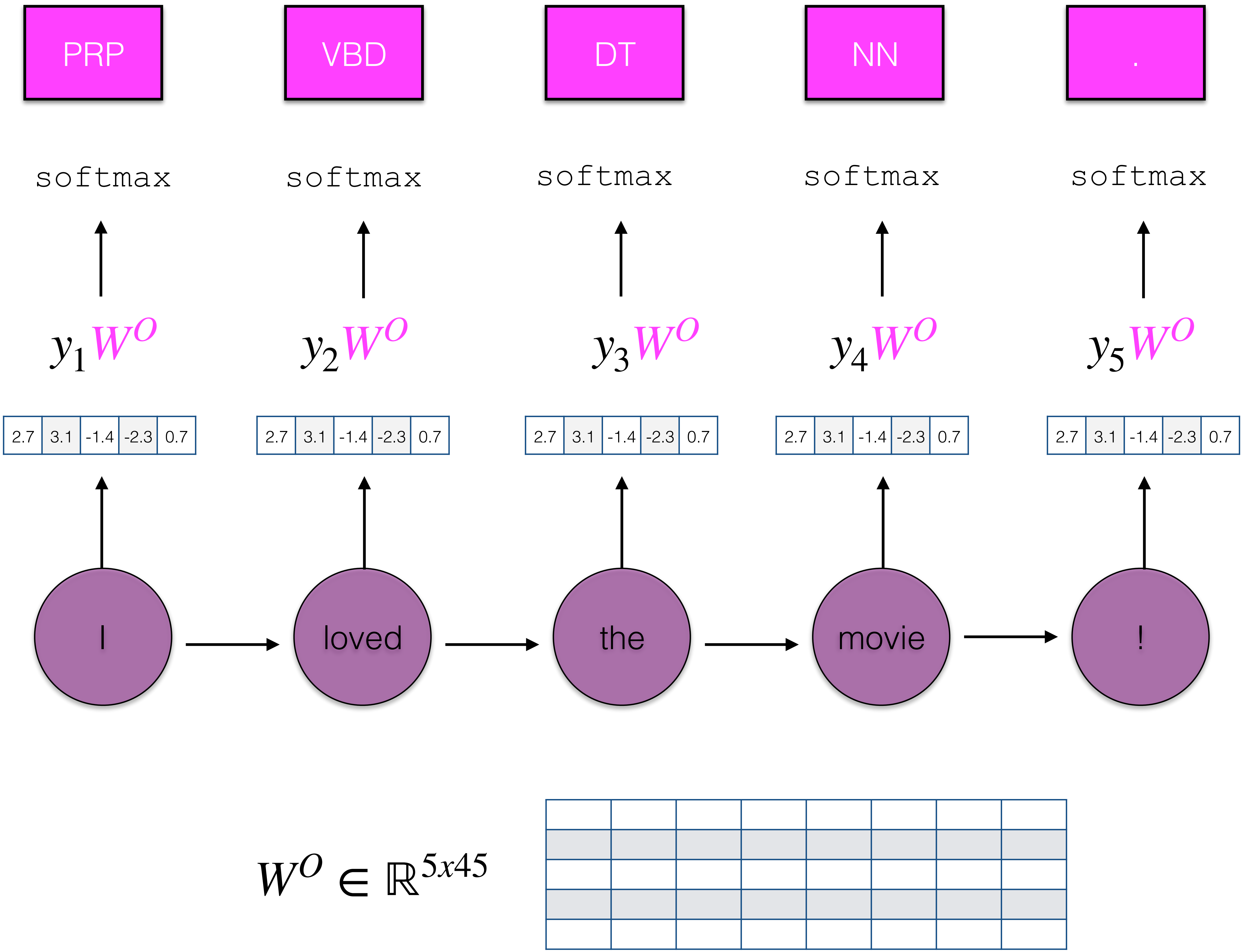
VBZ	0.51
NNS	0.48
JJ	0.01
NN	0
…	…

If we knew the categorical choice of VBZ at t_2, P(VB) at t_3 would be much lower.



TimeDistributed

- In keras, the `TimeDistributed` wrapper applies the same operation to every time step in a sequence (e.g., the same `Dense` layer with the same parameters)



PRP
VBD
DT
NN
.
softmax
softmax
softmax
softmax
softmax
$y_1 W^O$
$y_2 W^O$
$y_3 W^O$
$y_4 W^O$
$y_5 W^O$
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
2.7 3.1 -1.4 -2.3 0.7
I
loved
the
movie
!
$W^O \in \mathbb{R}^{5x45}$

TimeDistributed

```
lstm_output = LSTM(lstm_size,
return_sequences=True)
(embedded_sequences)

preds = TimeDistributed(Dense(output_dim,
activation="softmax"))(lstm_output)
```

Activity

- 12.ner/SequenceLabelingBiLSTM_TODO