



Natural Language Processing

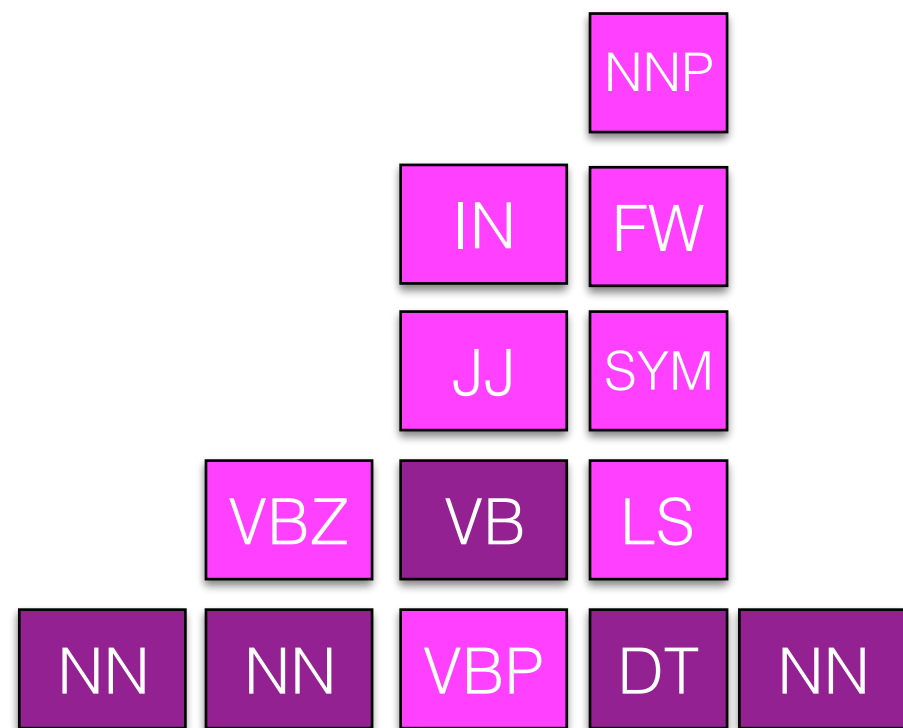
Info 159/259

Lecture 11: Sequence labeling 2 (Sept 28, 2017)

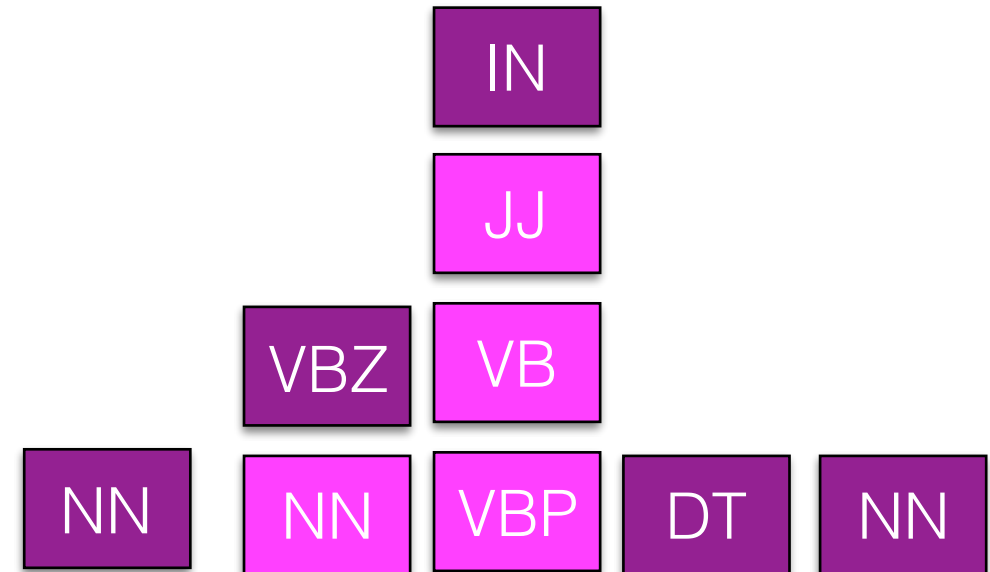
David Bamman, UC Berkeley

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

PERS

PERS

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

BIO notation

B-PERS

I-PERS

B-ORG

tim cook is the ceo of apple

BIO notation

B-PERS

B-PERS

After he helped Harry Ron went downstairs

Majority class

- Pick the label each word is seen most often with in the training data

fruit	flies	like	a	banana
NN 12	VBZ 7	VB 74	FW 8	NN 3
	NNS 1	VBP 31	SYM 13	
		JJ 28	LS 2	
		IN 533	JJ 2	
			IN 1	
			DT 25820	
			NNP 2	

Naive Bayes

- Treat each prediction as independent of the others

$$P(y \mid x) = \frac{P(y)P(x \mid y)}{\sum_{y' \in \mathcal{Y}} P(y')P(x \mid y')}$$

$$P(\text{VBZ} \mid \text{flies}) = \frac{P(\text{VBZ})P(\text{flies} \mid \text{VBZ})}{\sum_{y' \in \mathcal{Y}} P(y')P(\text{flies} \mid y')}$$

Reminder: how do we learn $P(y)$ and $P(x|y)$ from training data?

Logistic regression

- Treat each prediction as independent of the others but condition on much more expressive set of features

$$P(y \mid x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

$$P(\text{VBZ} \mid \text{flies}) = \frac{\exp(x^\top \beta_{\text{VBZ}})}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

Hidden Markov Model

Prior probability of label sequence

$$P(y) = P(y_1, \dots, y_n)$$

$$P(y_1, \dots, y_n) \approx \prod_{i=1}^{n+1} P(y_i \mid y_{i-1})$$

- We'll make a first-order Markov assumption and calculate the joint probability as the product the individual factors conditioned **only on the previous tag**.

MEMM

General maxent form

$$\arg \max_y P(y \mid x, \beta)$$

Maxent with first-order Markov
assumption: Maximum Entropy
Markov Model

$$\arg \max_y \prod_{i=1}^n P(y_i \mid y_{i-1}, x)$$

Features

$$f(t_i, t_{i-1}; x_1, \dots, x_n)$$

Features are scoped over
the previous predicted
tag and the entire
observed input

feature	example
$x_i = \text{man}$	1
$t_{i-1} = \text{JJ}$	1
$i=n$ (last word of sentence)	1
x_i ends in -ly	0

MEMM Training

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

For all training data, we want probability of the true label y_i conditioned on the previous true label y_{i-1} to be high.

This is simply multiclass logistic regression

MEMM Training

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

Locally normalized — at each time step,
each conditional distribution sums to 1

Label bias

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

- For a given conditioning context, the probability of the a tag (e.g., VBZ) only competes against other tags with that same context (e.g., NN)

Label bias

NN TO VB

will to fight

	NN	MD
$x_i = \text{will}$	10	40
$y_{i-1} = \text{START}$	-1	7
BIAS	7	-2

Modals show up much more frequently at the start of the sentence than nouns do (e.g., questions)

Label bias

NN TO VB

will to fight

But we know that MD + TO is very rare

- *can to eat
- *would to eat
- *could to eat
- *may to eat

Label bias

NN TO VB

will to fight

	TO
$x_i = to$	100000000
$y_{i-1} = NN$	0
$y_{i-1} = MD$	0

to is relatively deterministic
(almost always TO) so it doesn't
matter what tag precedes it.

Label bias

NN TO VB

will to fight

$$\prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

Because of this **local normalization**, $P(\text{TO} \mid \text{context})$ will always be 1 if $x = \text{"to"}$

Label bias

NN TO VB

will to fight

That means our prediction for *to* can't help us disambiguate will. We lose the information that MB + TO sequences rarely happen.

Conditional random fields

- We can solve this problem using global normalization (over the entire sequences) rather than locally normalized factors.

MEMM

$$P(y \mid x, \beta) = \prod_{i=1}^n P(y_i \mid y_{i-1}, x, \beta)$$

CRF

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

Conditional random fields

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

Feature vector scoped over the entire input and label sequence

$$\Phi(x, y) = \sum_{i=1}^n \phi(x, i, y_i, y_{i-1})$$

ϕ is the same feature vector we used for local predictions using MEMMs

Features

$$\phi(x, i, y_i, y_{i-1})$$

Features are scoped over
the previous predicted
tag and the entire
observed input

feature	example
$x_i = \text{man}$	1
$y_{i-1} = \text{JJ}$	1
$i=n$ (last word of sentence)	1
x_i ends in -ly	0

Conditional random fields

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

- In MEMMs, we normalize over the set of 45 POS tags
- CRFs are globally normalized, but the normalization complexity is huge — every possible sequence of labels of length n .

Forward algorithm (CRF)

$$P(y \mid x, \beta) = \frac{\exp(\Phi(x, y)^\top \beta)}{\sum_{y' \in \mathcal{Y}} \exp(\Phi(x, y')^\top \beta)}$$

- Calculating the denominator naively would involve a summation over K^N terms
- But we can do this efficiently in NK^2 time using the forward algorithm

For details, see: Collins, "The Forward-Backward Algorithm"

Forward algorithm (CRF)

END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	\$

$$\alpha(1, y) = \exp(\phi(x, i, y, \text{START})^\top \beta)$$

$$\alpha(i, y) = \sum_{y' \in \mathcal{S}} \alpha(i-1, y') \times \exp(\phi(x, i, y, y')^\top \beta)$$

Forward algorithm (CRF)

END							
DT							
NNP							
VB							
NN							
MD							
START							
	^	Janet	will	back	the	bill	\$

$$Z = \sum_{y' \in \mathcal{Y}} \exp \left(\Phi(x, y')^\top \beta \right) = \sum_{s \in \mathcal{S}} \alpha(n, s)$$

Conditional random fields

With a CRF, we have exactly the same parameters as we do with an equivalent MEMM; but we learn the best values of those parameters that leads to the best probability of the **sequence** overall (in our training data)

MEMM

	TO
$x_i = \text{to}$	100000000
$y_{i-1} = \text{NN}$	0
$y_{i-1} = \text{MD}$	0

CRF

	TO
$x_i = \text{to}$	7.8
$y_{i-1} = \text{NN}$	1.4
$y_{i-1} = \text{MD}$	-5.8

Parameter estimation

- Just like logistic regression/MEMM, we can find the optimal values for the parameters using stochastic gradient descent for a given sequence x and true labels y .

$$\frac{\partial L}{\partial \beta_k} = \Phi_k(x, y) - \sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta) \Phi_k(x, y')$$

Features for the true
label y

Expected feature counts under the
probability for a sequence assigned by
current β

Parameter estimation

- Just like logistic regression/MEMM, we can find the optimal values for the parameters using stochastic gradient descent for a given sequence x and true labels y .

$$\frac{\partial L}{\partial \beta_k} = \Phi_k(x, y) - \sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta) \Phi_k(x, y')$$

If current model assigns probability of 1 to true sequence and 0 to all other K^N sequences, then gradient = 0

Parameter estimation

$$\sum_{y' \in \mathcal{Y}} P(y' \mid x, \beta) \Phi_k(x, y')$$

$$= \sum_{i=1}^n \sum_{a \in \mathcal{S}, b \in \mathcal{S}} \phi_k(x, i, a, b) \sum_{y' \in \mathcal{Y}: y'_{i-1}=a, y'_i=b} P(y' \mid x, \beta)$$

This entire sum can be found in NK^2 time using the forward-backward algorithm

For details, see: Collins, "The Forward-Backward Algorithm"

Viterbi Decoding

$$v_1(y) = \exp(\phi(x, 1, START, y)^\top \beta)$$

$$v_t(y) = \max_{u \in \mathcal{S}} [v_{t-1}(u) \times \exp(\phi(x, t, y, u)^\top \beta)]$$

(equivalently)

$$v_1(y) = \phi(x, 1, START, y)^\top \beta$$

$$v_t(y) = \max_{u \in \mathcal{S}} [v_{t-1}(u) + \phi(x, t, y, u)^\top \beta]$$

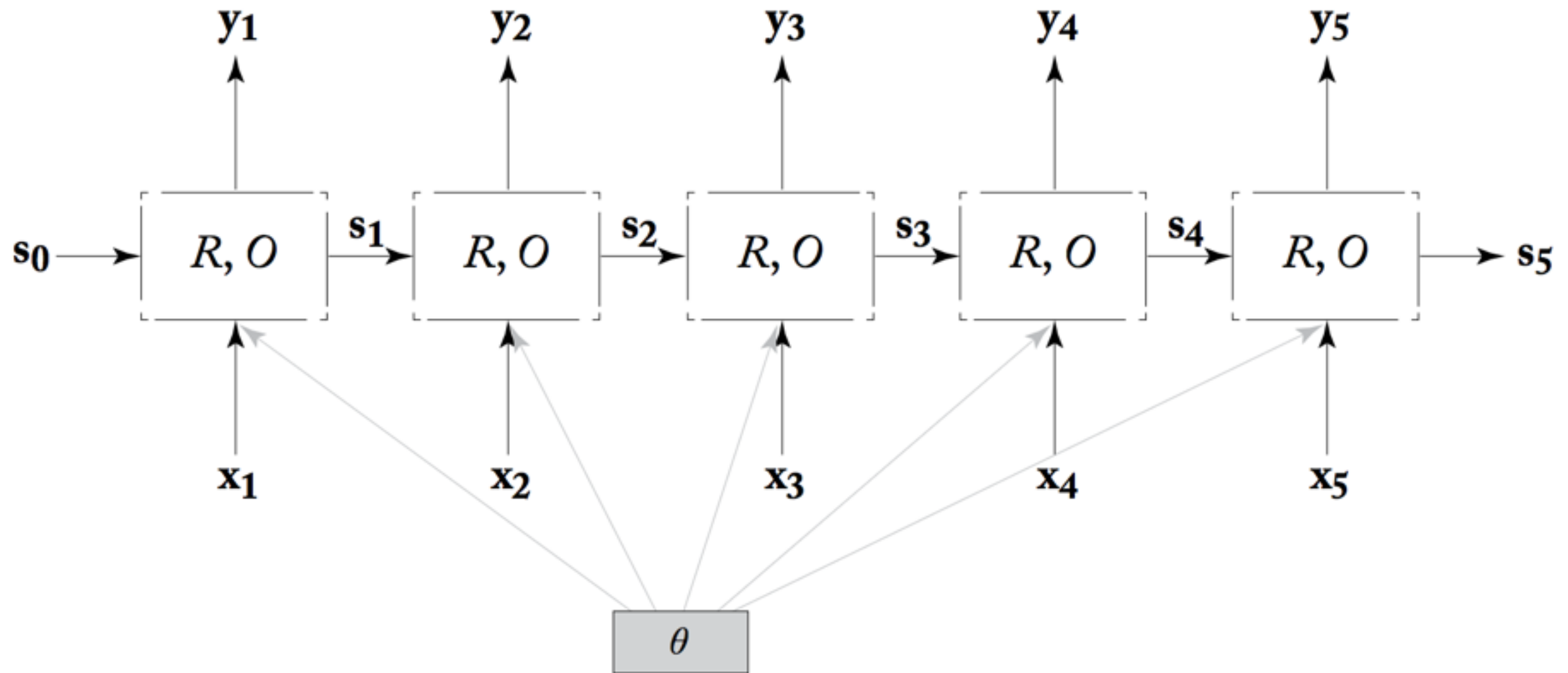
In practice

- CRF training is slow! NK^2 complexity for each sequence at each gradient step.
- Accuracy is typically better than MEMMs

Recurrent neural network

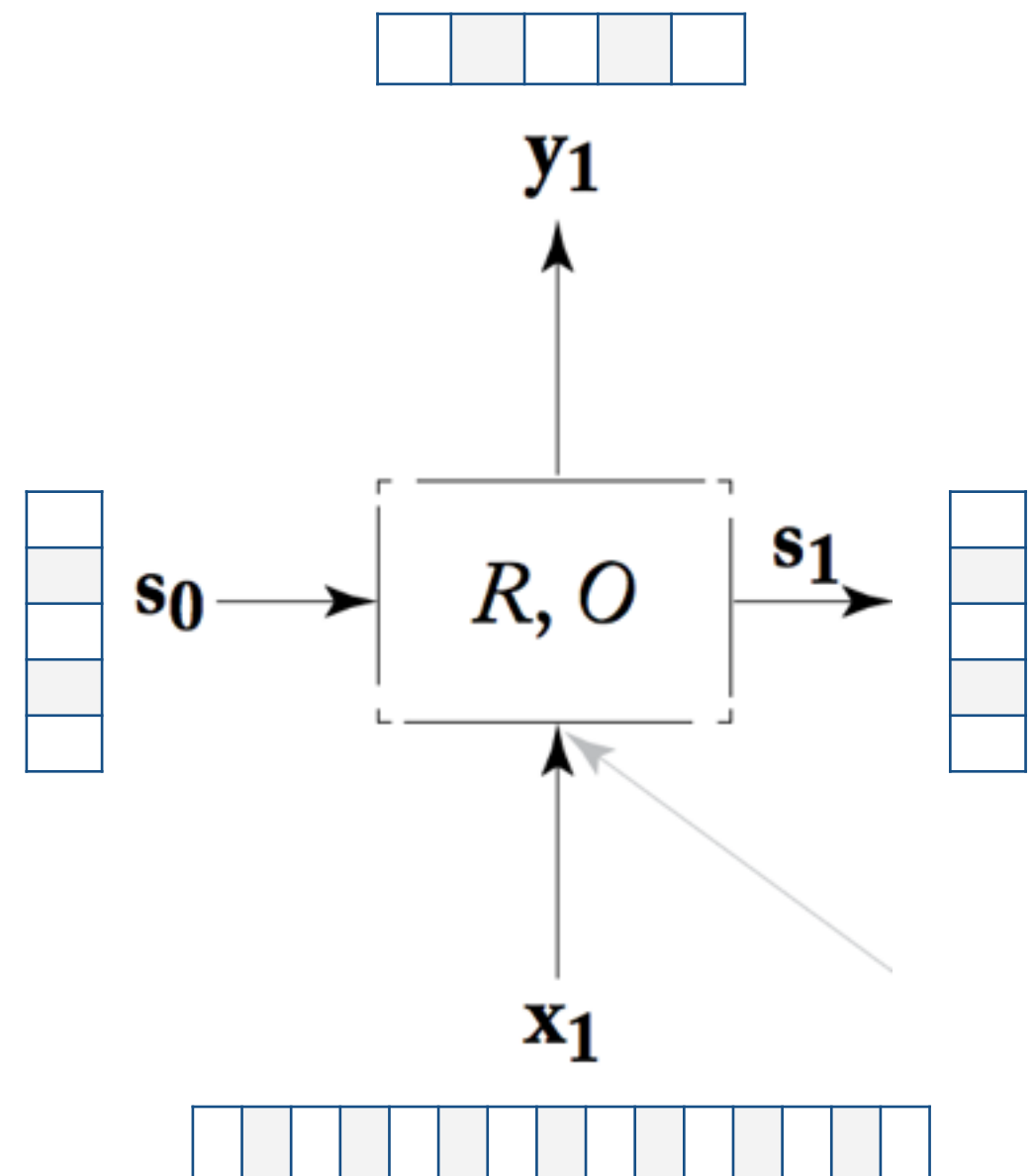
- RNN allow arbitrarily-sized conditioning contexts; condition on the **entire sequence history**.

Recurrent neural network



Recurrent neural network

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



Training RNNs

- Given this definition of an RNN:

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

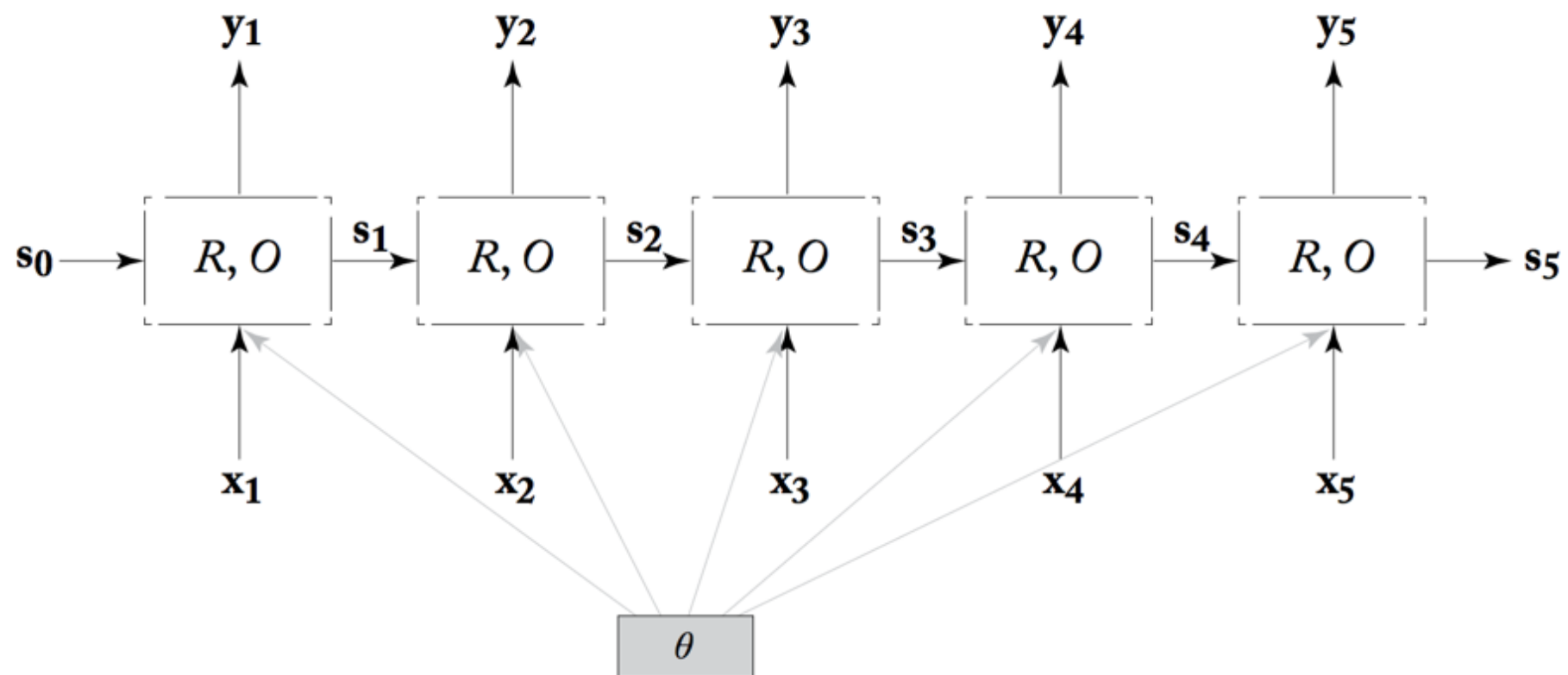
$$y_i = O(s_i) = \text{softmax}(s_iW^o + b^o)$$

- We have five sets of parameters to learn:

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

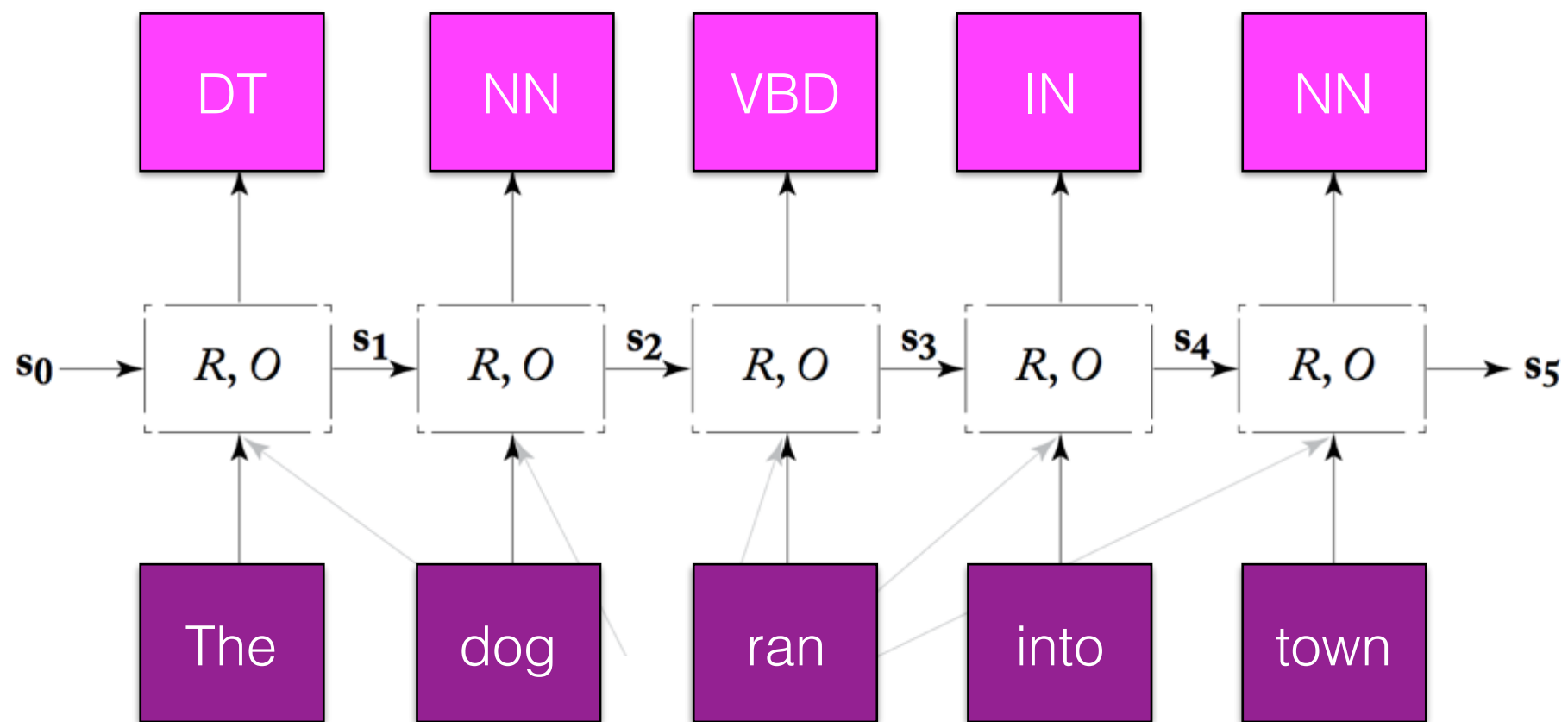
Recurrent neural network

- RNNs for language modeling are already performing a kind of sequence labeling: at each time step, predict the **word** from $\mathcal{V}$ conditioned on the context



Recurrent neural network

- For POS tagging, predict the **tag** from **y** conditioned on the context



RNNs for POS

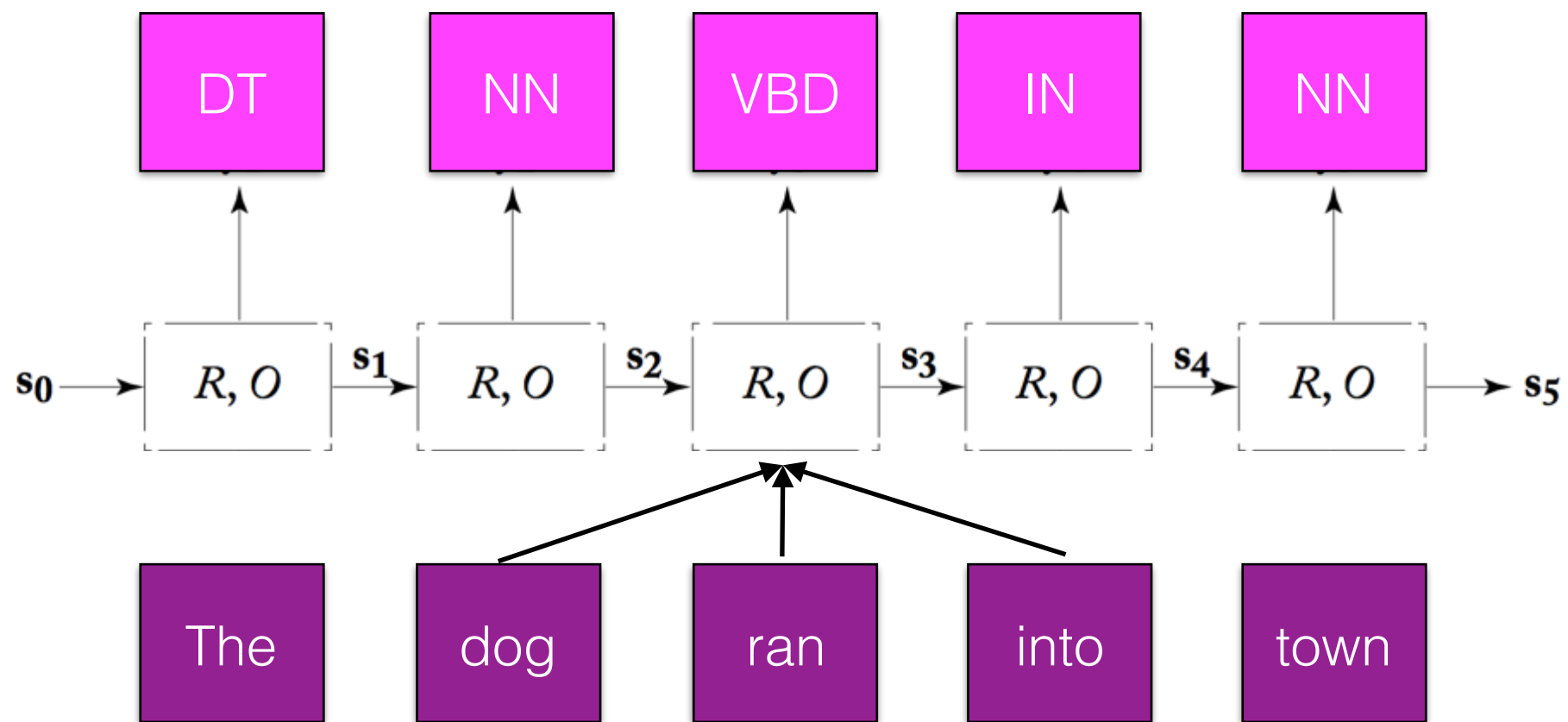
NN TO VB

will to fight

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

Recurrent neural network

- The simplest thing we could do is condition on a window of inputs



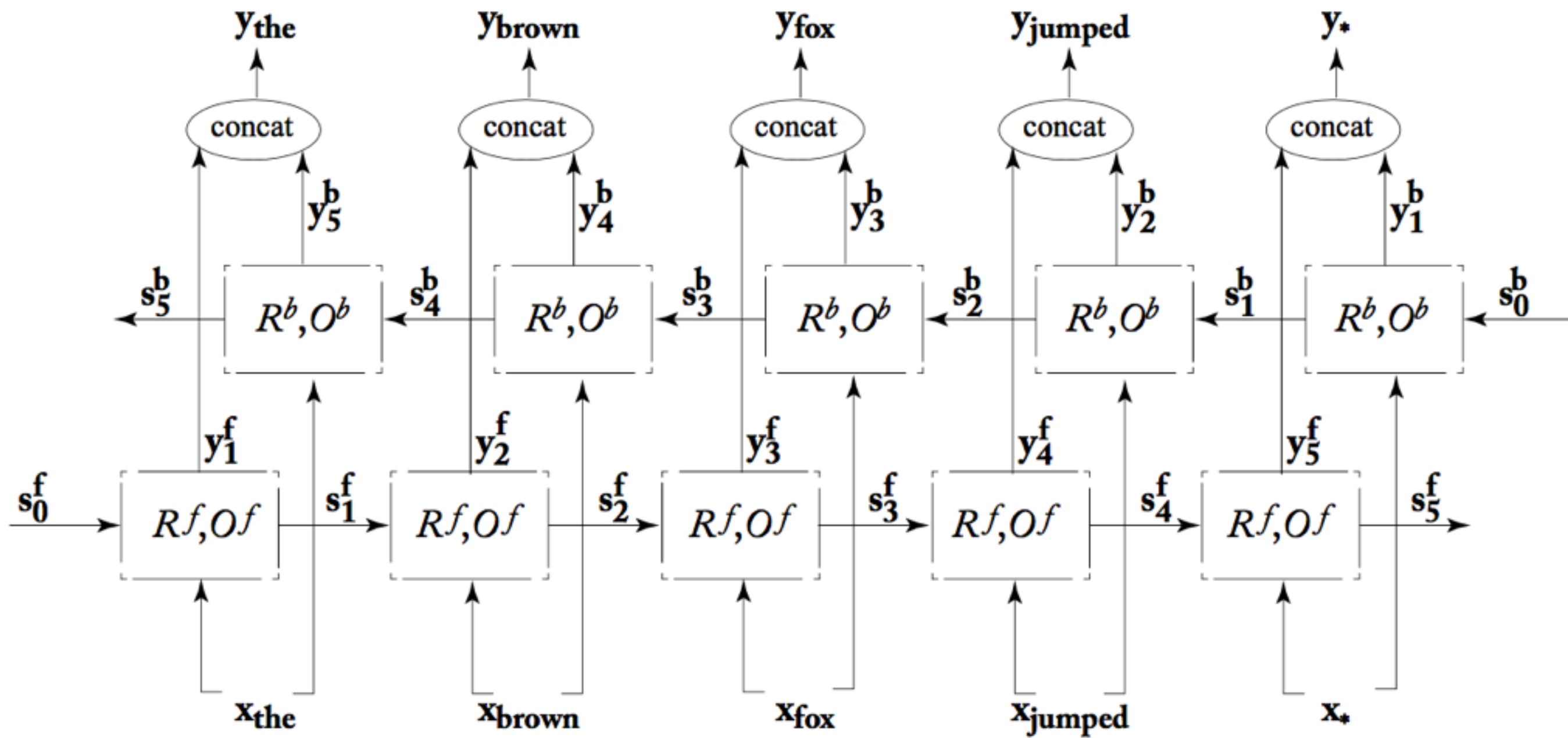
Recurrent neural network

- e.g. 5 time steps, input = $[X_{i-1}, X_i, X_{i+1}]$
- Quite similar to what we do with logistic regression

0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
-4.5	-4.5	-4.5	-4.5	-4.5
0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
-4.5	-4.5	-4.5	-4.5	-4.5
0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
0.7	0.7	0.7	0.7	0.7
1.3	1.3	1.3	1.3	1.3
-4.5	-4.5	-4.5	-4.5	-4.5

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



Training BiRNNs

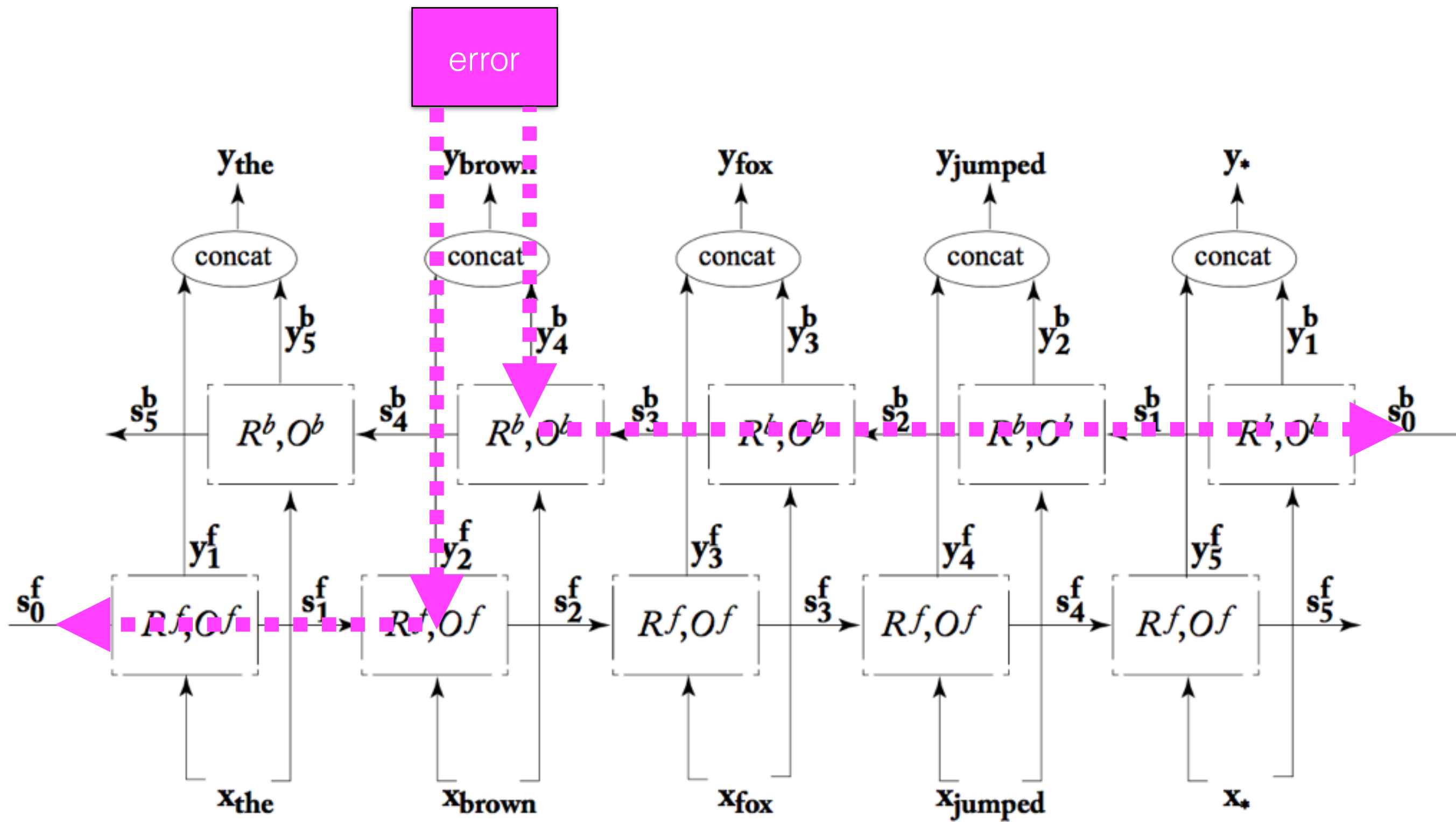
- Given this definition of an BiRNN:

$$s_b^i = R_b(x^i, s_b^{i+1}) = g(s_b^{i+1} \mathbf{W}_b^s + x^i \mathbf{W}_b^x + \mathbf{b}_b)$$

$$s_f^i = R_f(x^i, s_f^{i-1}) = g(s_f^{i-1} \mathbf{W}_f^s + x^i \mathbf{W}_f^x + \mathbf{b}_f)$$

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

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



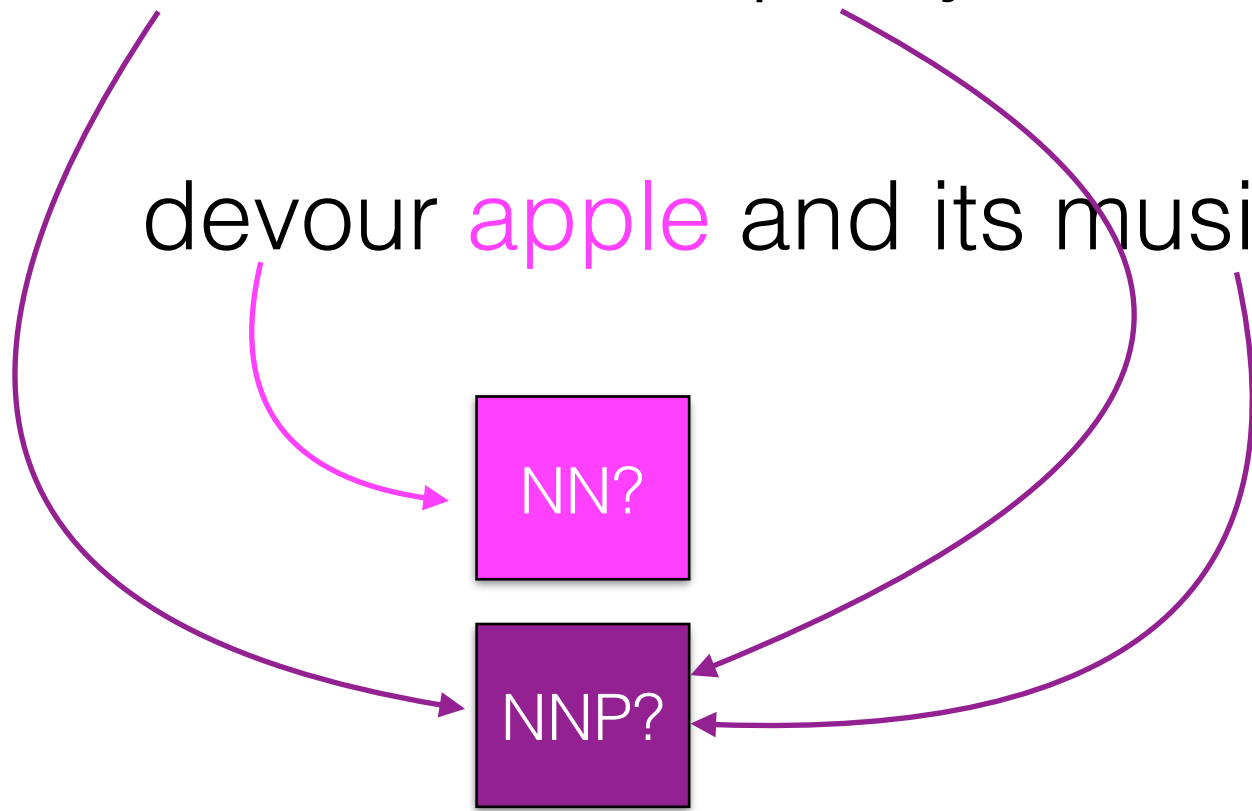
RNNs for POS

amazon and spotify's streaming services are going to

devour **apple** and its music purchasing model

NN?

NNP?



RNNs for POS

amazon and spotify's streaming services are going to devour **apple** and its music purchasing model

Prediction:

Can the information from far away get to the time step that needs it?

Training:

Can error reach that far back during backpropagation?

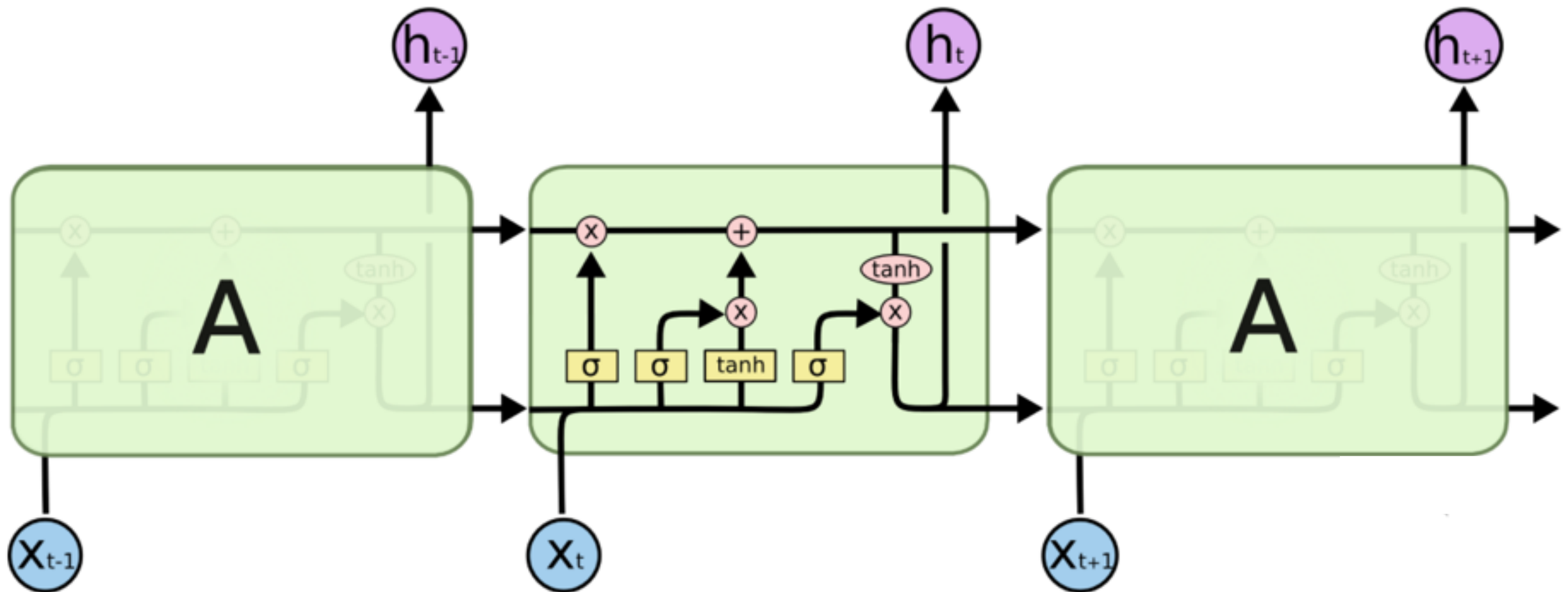
RNNs

- Recurrent networks are deep in that they involve one “layer” for each time step (e.g., words in a sentence)
- **Vanishing gradient problem**: as error is back propagated through the layers of a deep network, they tend toward 0.

Long short-term memory network (LSTM)

- Designed to account for the vanishing gradient problem
- Basic idea: split the s vector propagated between time steps into a **memory** component and a **hidden state** component

LSTMs



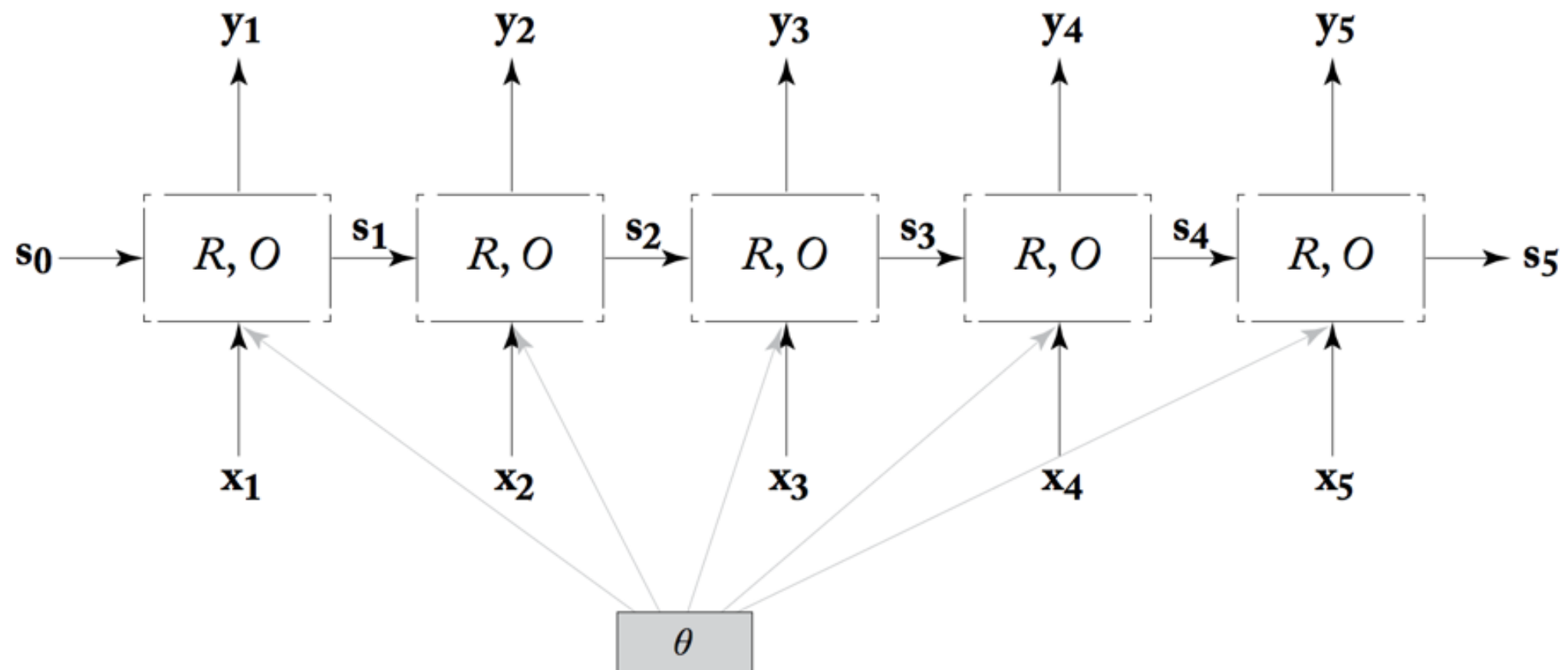
<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

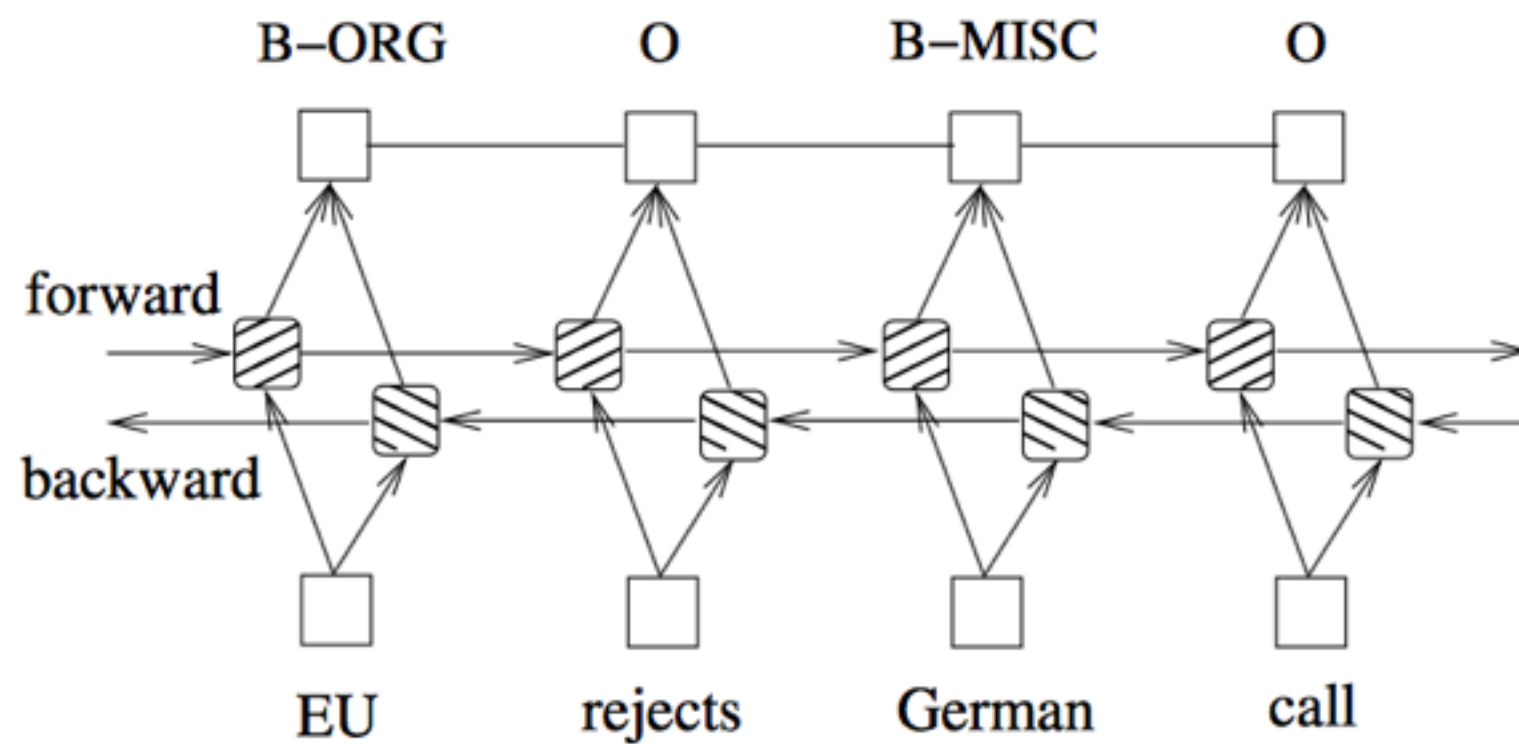
LSTM/RNN

- Is an RNN the same kind of sequence labeling model as an MEMM or CRF?
- It doesn't use nearby **labels** in making predictions!
(More like logistic regression in this respect)

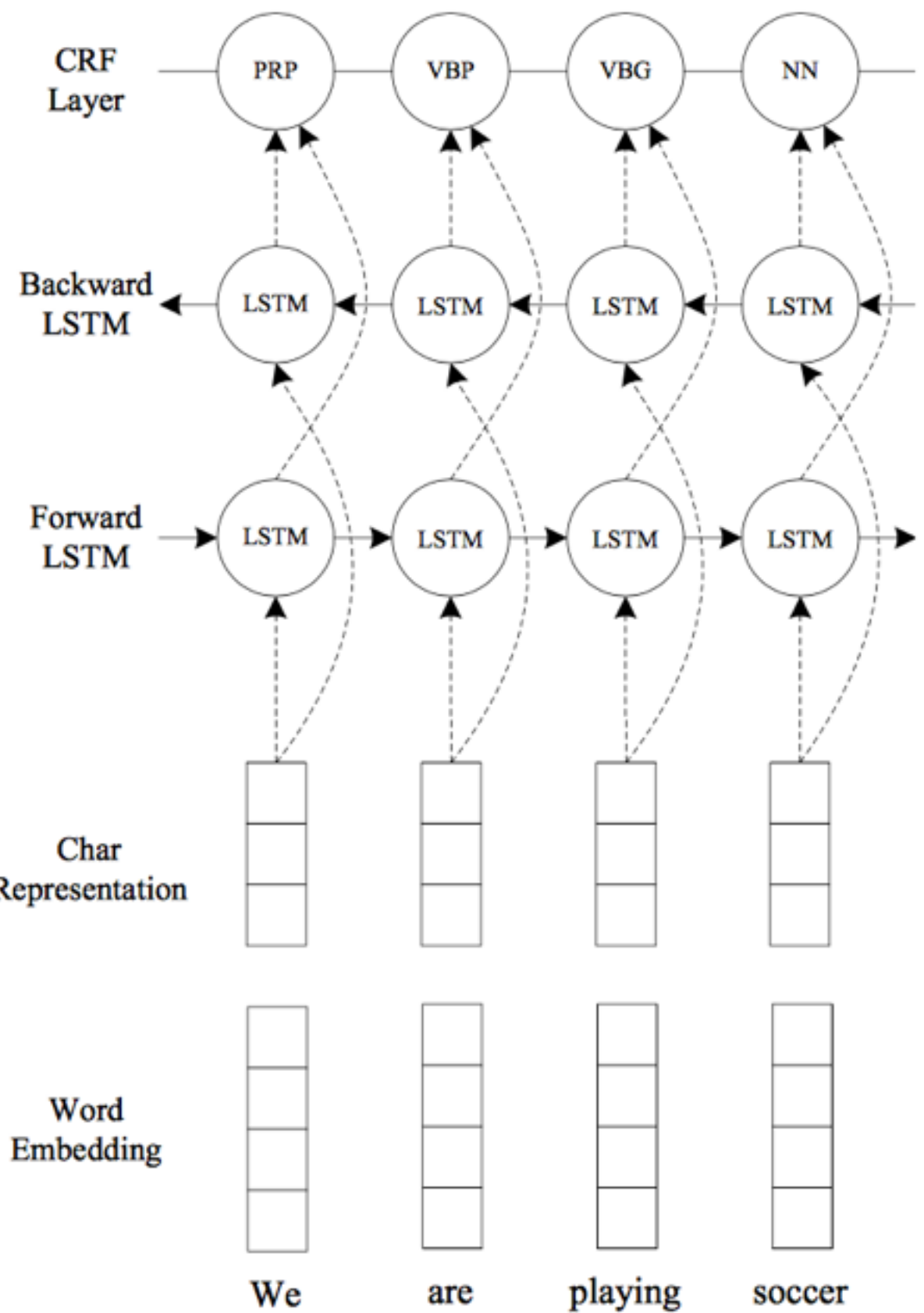
Recurrent neural network

- How could we incorporate nearby label information into a single-direction RNN?





Huang et al. 2015, "Bidirectional LSTM-CRF Models for Sequence Tagging"



Ma and Hovy (2016), "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF"

Model	POS		NER					
	Dev	Test	Dev			Test		
	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

Ma and Hovy (2016), “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF”