Natural Language Processing

Info 159/259

Lecture 12: MEMMs, CRFs and neural sequence labeling (Feb 24, 2022)

David Bamman, UC Berkeley

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
- Model correlations in the labels y.

Named entity recognition

[tim cook]$_{PER}$ is the ceo of [apple]$_{ORG}$

- Identifying spans of text that correspond to typed entities

Named entity recognition

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	The **Mt. Sanitas** loop is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states, provinces	**Palo Alto** is raising the fees for parking.
Facility	FAC	bridges, buildings, airports	Consider the **Golden Gate Bridge**.
Vehicles	VEH	planes, trains, automobiles	It was a classic **Ford Falcon**.

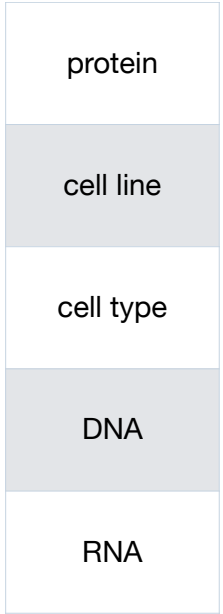
Figure 17.1 A list of generic named entity types with the kinds of entities they refer to.

ACE NER categories (+weapon)

Named entity recognition

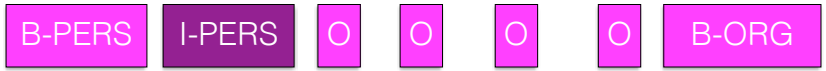
- GENIA corpus of MEDLINE abstracts (biomedical)

We have shown that [interleukin-1]$_{\text{PROTEIN}}$ ([IL-1]$_{\text{PROTEIN}}$) and [IL-2]$_{\text{PROTEIN}}$ control [IL-2 receptor alpha (IL-2R alpha) gene]$_{\text{DNA}}$ transcription in [CD4-CD8- murine T lymphocyte precursors]$_{\text{CELL LINE}}$



http://www.aclweb.org/anthology/W04-1213

BIO notation

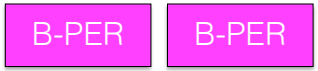


tim cook is the ceo of apple

- **B**eginning of entity
- **I**nside entity
- **O**utside entity

[tim cook]$_{PER}$ is the ceo of [apple]$_{ORG}$

Named entity recognition



After he saw Harry Tom went to the store

Evaluation

- We evaluate NER with precision/recall/F1 over typed chunks.

Evaluation

	1	2	3	4	5	6	7
	tim	cook	is	the	CEO	of	Apple
gold	B-PER	I-PER	O	O	O	O	B-ORG
system	B-PER	O	O	O	B-PER	O	B-ORG

<start, end, type>

Precision	1/3
Recall	1/2

gold

<1,2,PER>
<7,7,ORG>

system

<1,1,PER>
<5,5,PER>
<7,7,ORG>

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
- Model correlations in the labels y.

Generative vs. Discriminative models

- Generative models specify a joint distribution over the labels and the data. With this you could generate new data.

$$P(x, y) = P(y)\, P(x \mid y)$$

- Discriminative models specify the conditional distribution of the label y given the data x. These models focus on how to discriminate between the classes

$$P(y \mid x)$$

MEMM

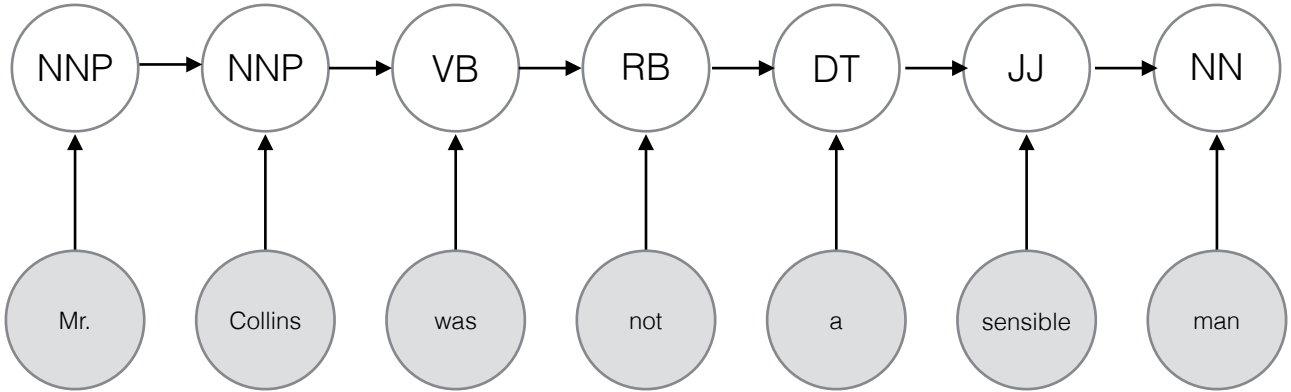
General maxent form

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

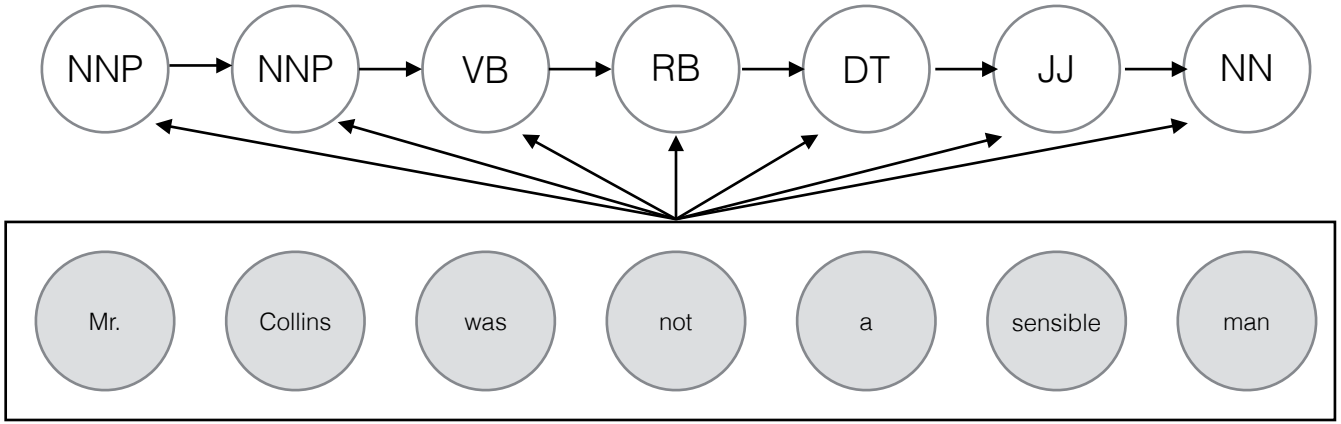
Maxent with first-order Markov assumption: Maximum Entropy Markov Model

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

MEMM

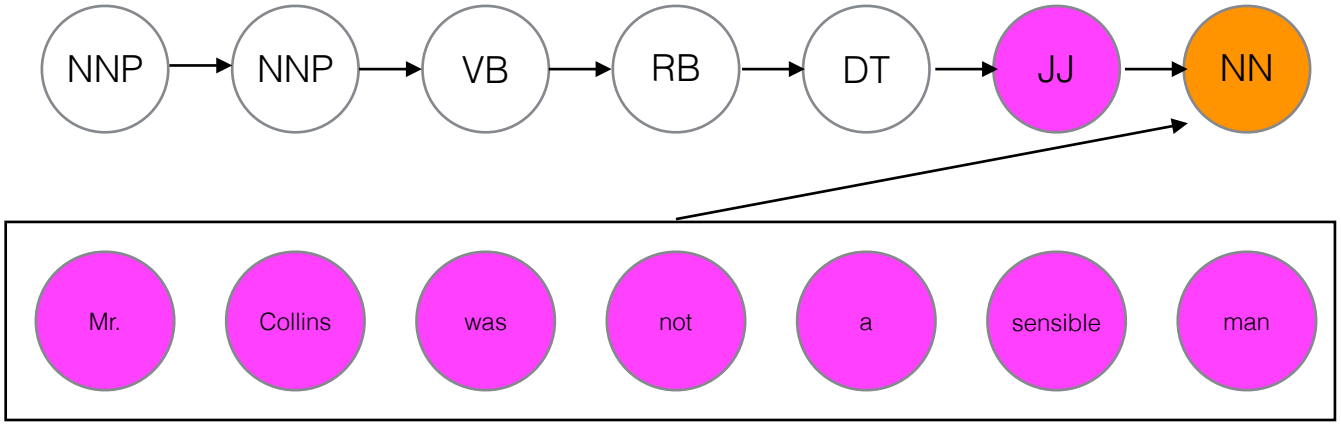


MEMM



MEMMs condition on the *entire* input

MEMM



Features

$$f(y_i, y_{i-1}; x_1, \ldots, x_n)$$

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

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

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

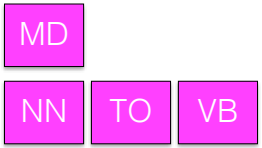
Label bias

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

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

Bottou 2001; Lafferty et al. 2001

Label bias



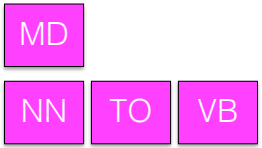
will to fight

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

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

Toutanova et al. 2003

Label bias



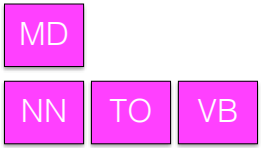
will to fight

But we know that MD + TO is very rare

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

Toutanova et al. 2003

Label bias



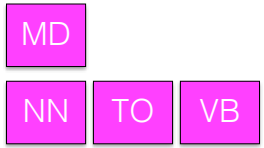
will to fight

	TO
x_i=to	10000000
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.

Toutanova et al. 2003

Label bias



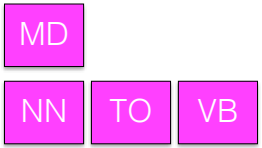
will to fight

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

Because of this local normalization, P(TO | context) will always be 1 if x="to"

Toutanova et al. 2003

Label bias



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

Toutanova et al. 2003

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

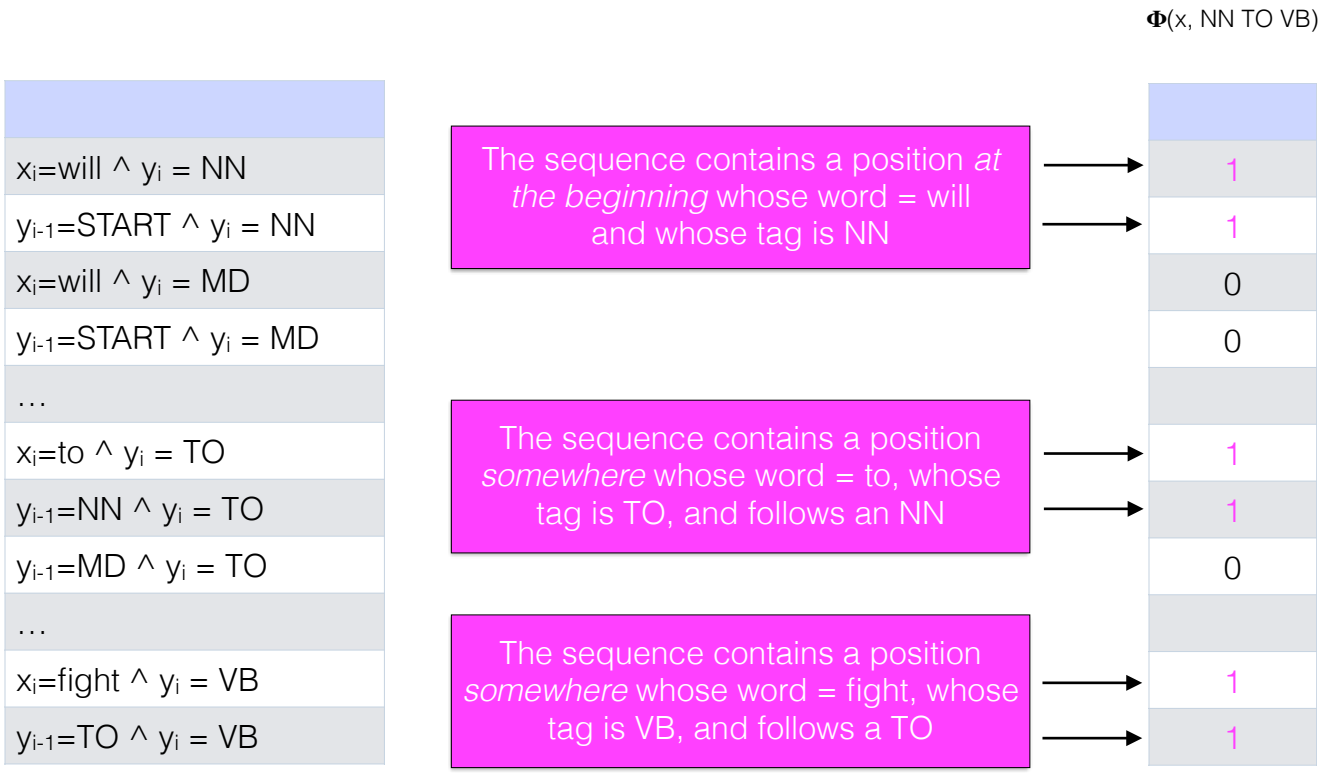
In an MEMM, we estimate $P(y_t \mid y_{t-1}, x, \beta)$ from each $\phi(x, t, y_t, y_{t-1})$ independently

	will $\phi(x, 1, y_1, y_0)$	to $\phi(x, 2, y_2, y_1)$	fight $\phi(x, 3, y_3, y_2)$
x_i=will ^ y_i = NN	1	0	0
y_{i-1}=START ^ y_i = NN	1	0	0
x_i=will ^ y_i = MD	0	0	0
y_{i-1}=START ^ y_i = MD	0	0	0
…			
x_i=to ^ y_i = TO	0	1	0
y_{i-1}=NN ^ y_i = TO	0	1	0
y_{i-1}=MD ^ y_i = TO	0	0	0
…			
x_i=fight ^ y_i = VB	0	0	1
y_{i-1}=TO ^ y_i = VB	0	0	1

In a CRF, we use features from the entire sequence (by summing the individual features at each time step)

	will $\phi(x, 1, y_1, y_0)$	to $\phi(x, 2, y_2, y_1)$	fight $\phi(x, 3, y_3, y_2)$	$\Phi(x, \text{NN TO VB})$
x_i=will ^ y_i = NN	1	0	0	1
y_{i-1}=START ^ y_i = NN	1	0	0	1
x_i=will ^ y_i = MD	0	0	0	0
y_{i-1}=START ^ y_i = MD	0	0	0	0
…				
x_i=to ^ y_i = TO	0	1	0	1
y_{i-1}=NN ^ y_i = TO	0	1	0	1
y_{i-1}=MD ^ y_i = TO	0	0	0	0
…				
x_i=fight ^ y_i = VB	0	0	1	1
y_{i-1}=TO ^ y_i = VB	0	0	1	1

In a CRF, we use features from the entire sequence (by summing the individual features at each time step)



This lets us isolate the global sequence features that separate good sequences (in our training data) from bad sequences (not in our training data)

	Φ(x, NN TO VB) GOOD	Φ(x, MD TO VB) BAD
x_i=will ^ y_i = NN	1	0
y_{i-1}=START ^ y_i = NN	1	0
x_i=will ^ y_i = MD	0	1
y_{i-1}=START ^ y_i = MD	0	1
…		
y_{i-1}=NN ^ y_i = TO	1	0
y_{i-1}=MD ^ y_i = TO	0	1
x_i=to ^ y_i = TO	1	1
x_i=fight ^ y_i = VB	1	1
y_{i-1}=TO ^ y_i = VB	1	1

these are the different (and so are potentially predictive of a good label sequence)

these are the same (and so are not)

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)

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

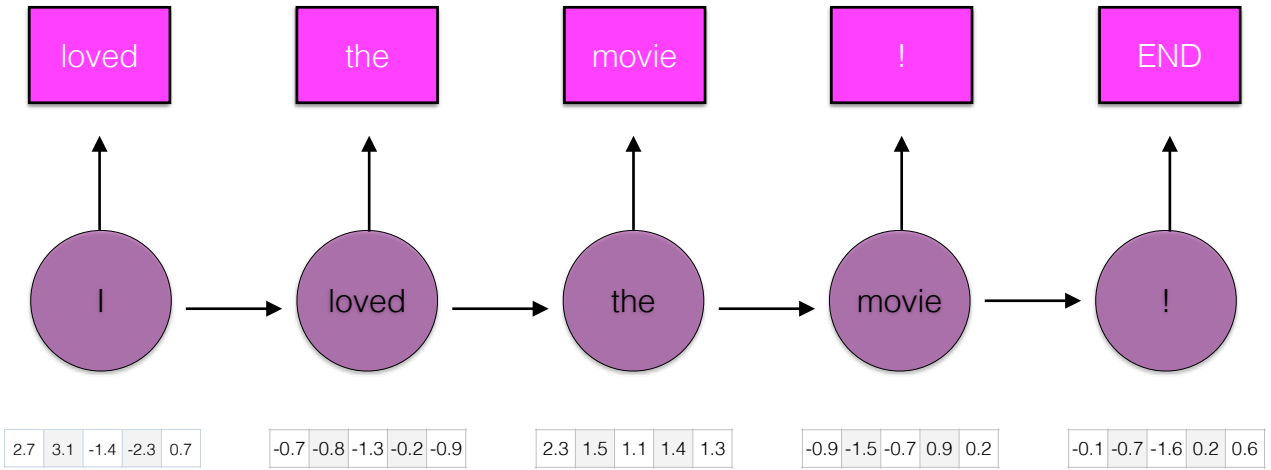
- Calculating the denominator naively would involve a summation over K^N terms
- But we can do this efficiently in NK^2 time using the forward algorithm

For details, see: Collins, “The Forward-Backward Algorithm”

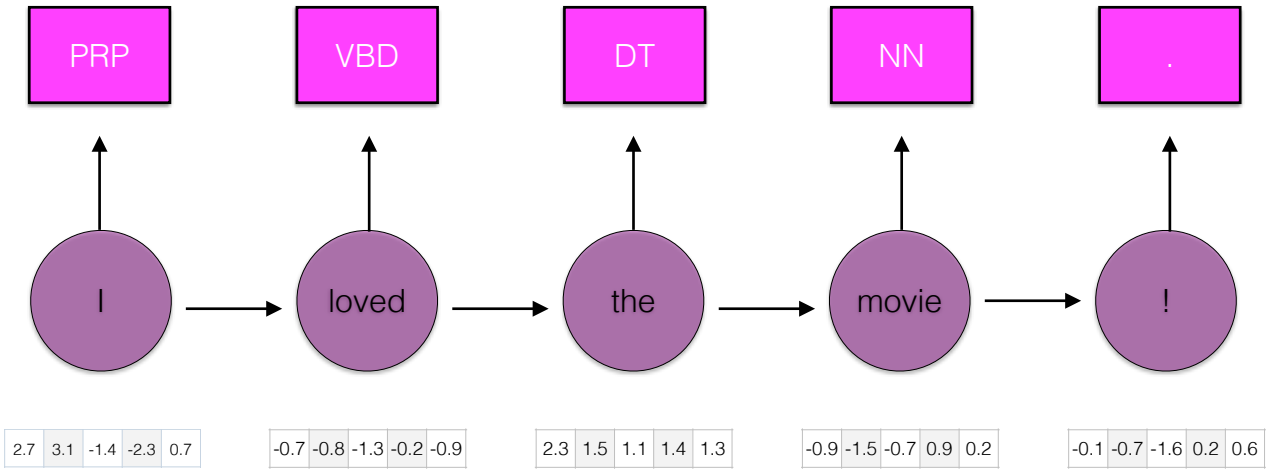
Recurrent neural network

- RNNs allow arbitarily-sized conditioning contexts and condition on the entire sequence history.

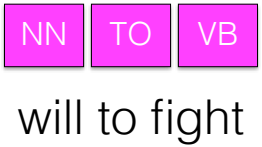
RNNs for language modeling are already performing a kind of sequence labeling: at each time step, predict the word from $\mathscr{V}$ conditioned on the context



For POS tagging, predict the tag from $\mathscr{Y}$ conditioned on the context



RNNs for POS



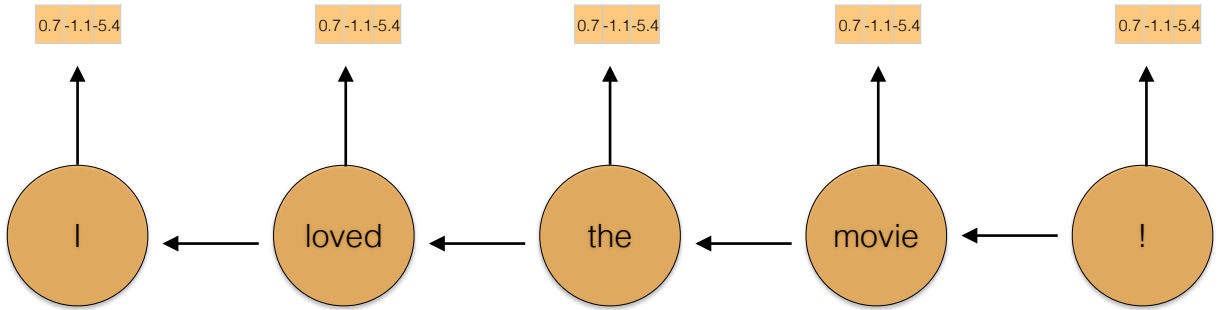
- To make a prediction for y_t, RNNs condition on all input seen through time t (x_1, …, x_t)
- But knowing something about the future can help (x_{t+1}, …, x_n)

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



2.7	3.1	-1.4	-2.3	0.7

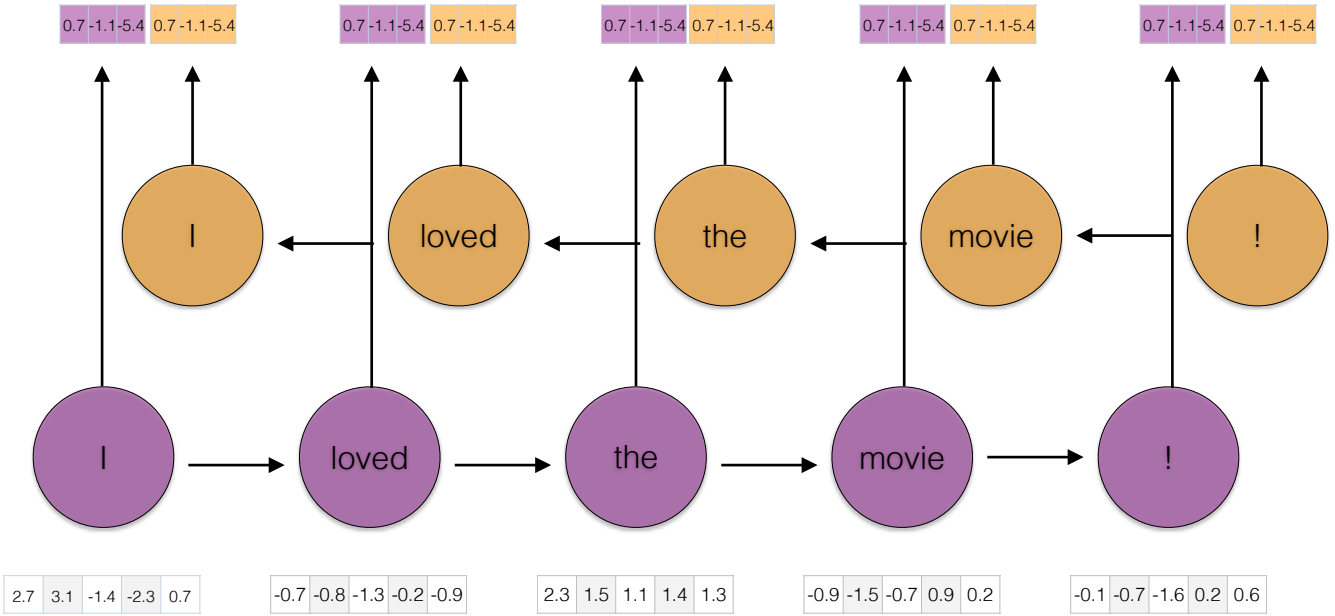
-0.7	-0.8	-1.3	-0.2	-0.9

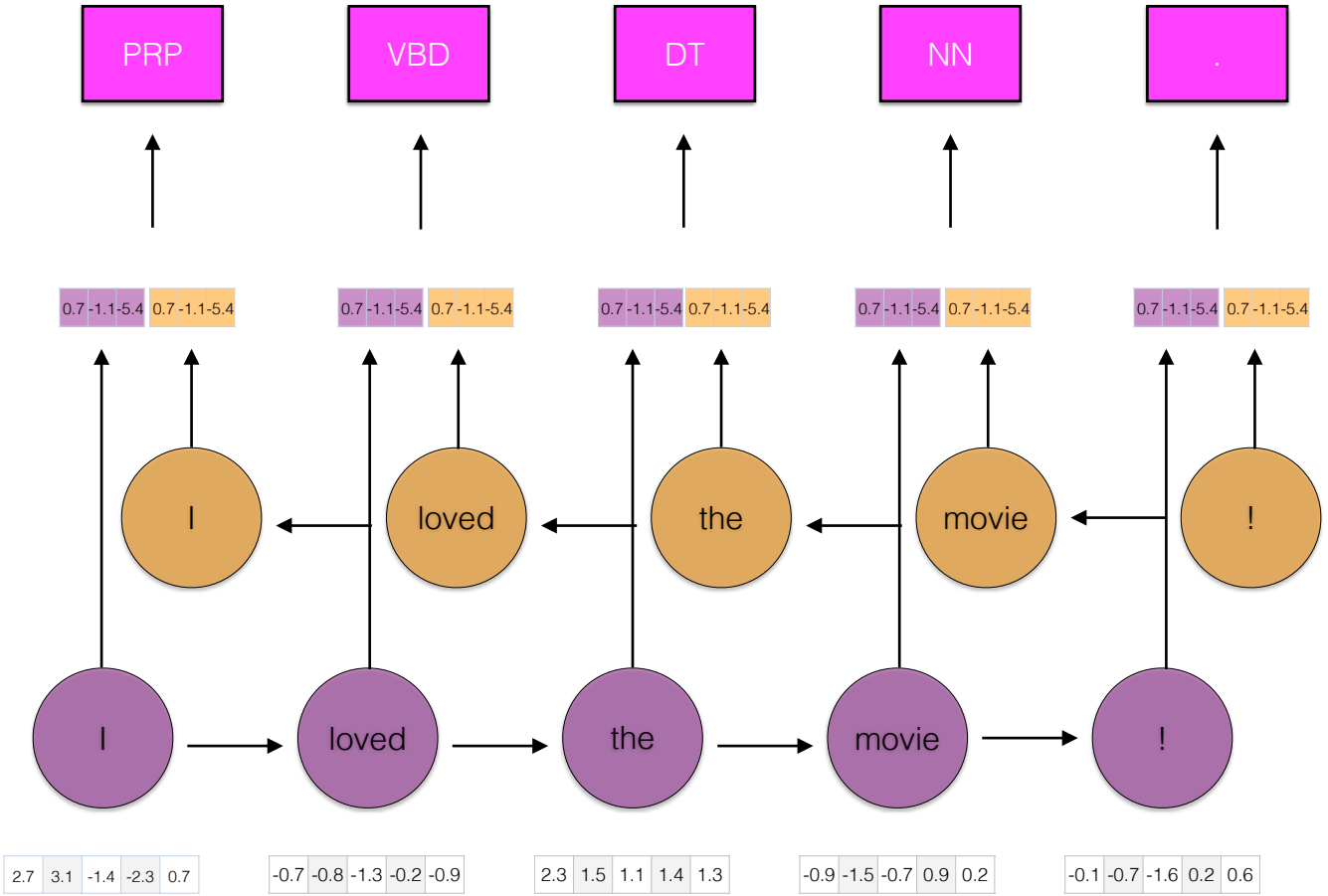
2.3	1.5	1.1	1.4	1.3

-0.9	-1.5	-0.7	0.9	0.2

-0.1	-0.7	-1.6	0.2	0.6

Bidirectional RNN

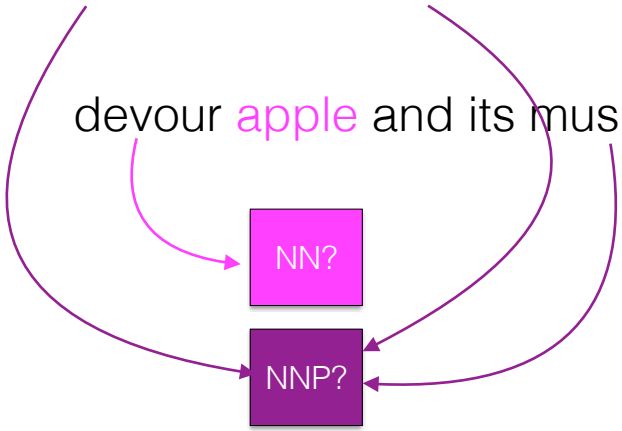



PRP
VBD
DT
NN
.
0.7 -1.1 -5.4 0.7 -1.1 -5.4
I
loved
the
movie
!
I
loved
the
movie
!
2.7 3.1 -1.4 -2.3 0.7
-0.7 -0.8 -1.3 -0.2 -0.9
2.3 1.5 1.1 1.4 1.3
-0.9 -1.5 -0.7 0.9 0.2
-0.1 -0.7 -1.6 0.2 0.6

RNNs for POS tagging

amazon and spotify's streaming services are going to

devour apple and its music purchasing model



RNNs for POS tagging

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?

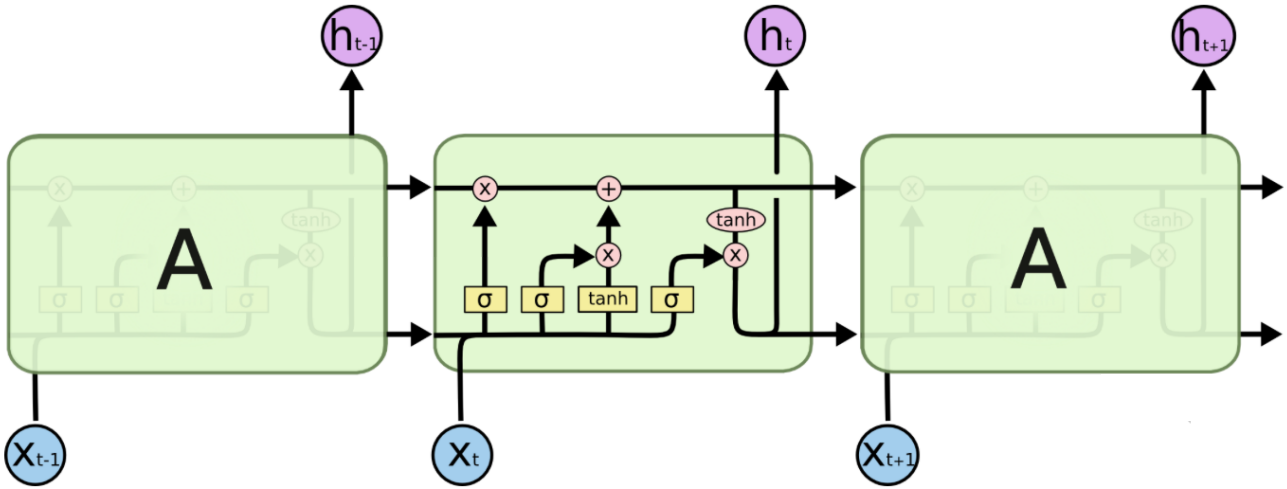
RNNs

- 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

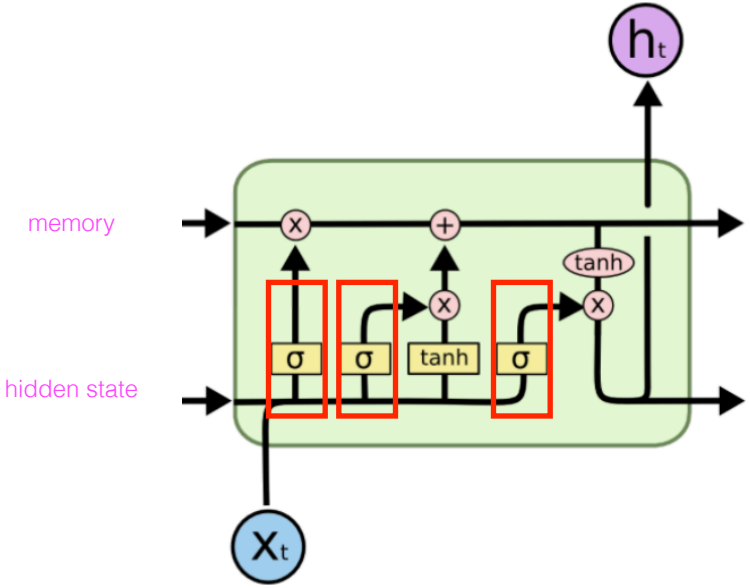
LSTMs



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

Gates

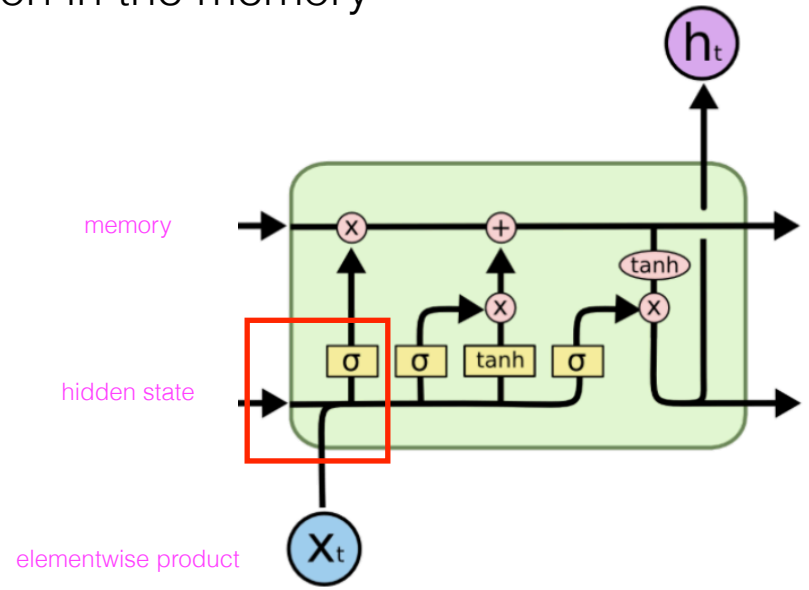
- LSTMs gates control the flow of information



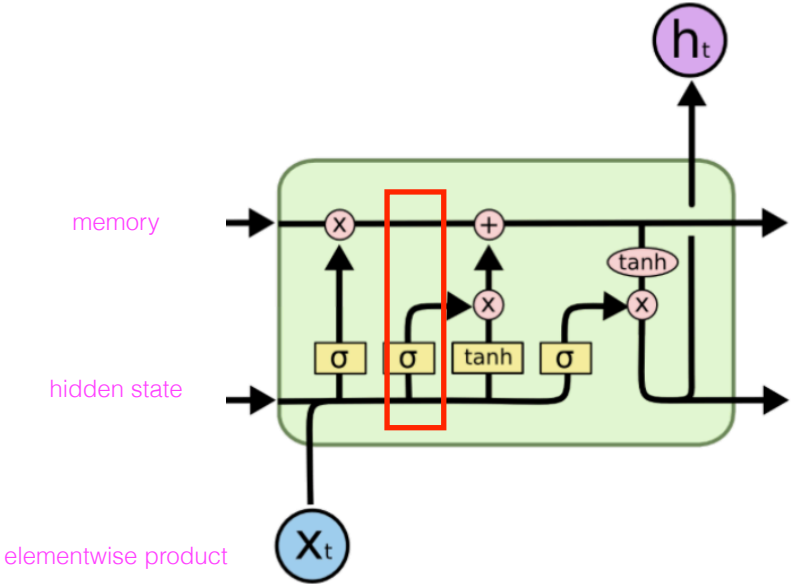
- A sigmoid squashes its input to between 0 and 1
- By multiplying the output of a sigmoid elementwise with another vector, we forget information in the vector (if multiplied by 0) or allow it to pass (if multiplied by 1)


input
3.7
1.4
-0.7
-1.4
7.8
gate
0.01
0.99
0.5
0.98
0.01
output
0.03
1.4
-0.35
-1.38
0.08

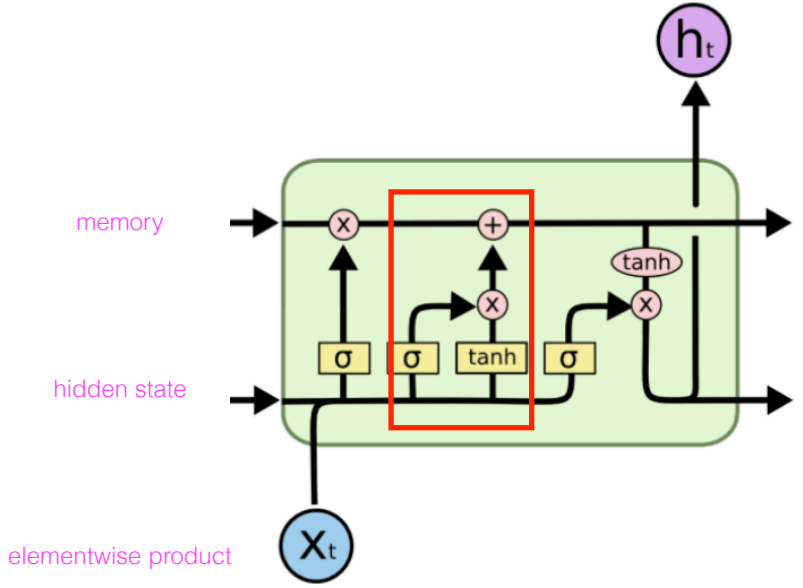
Forget gate: as a function of the previous hidden state and current input, forget information in the memory



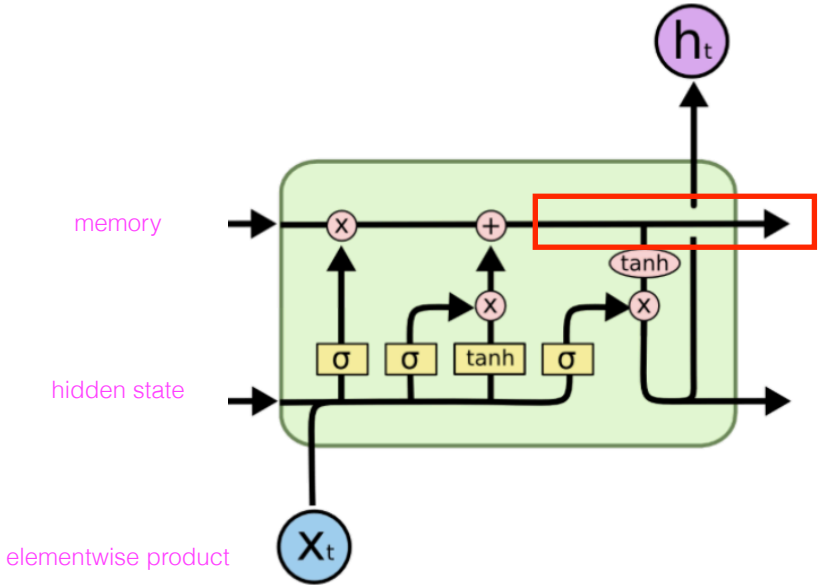
Input gate (but forget some information about the current observation)



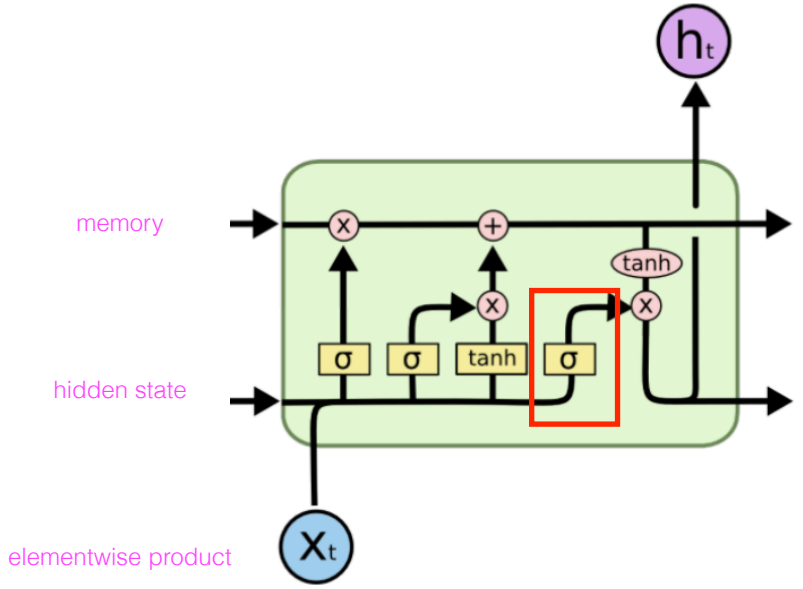
Update the memory (but forget some information about the current observation)



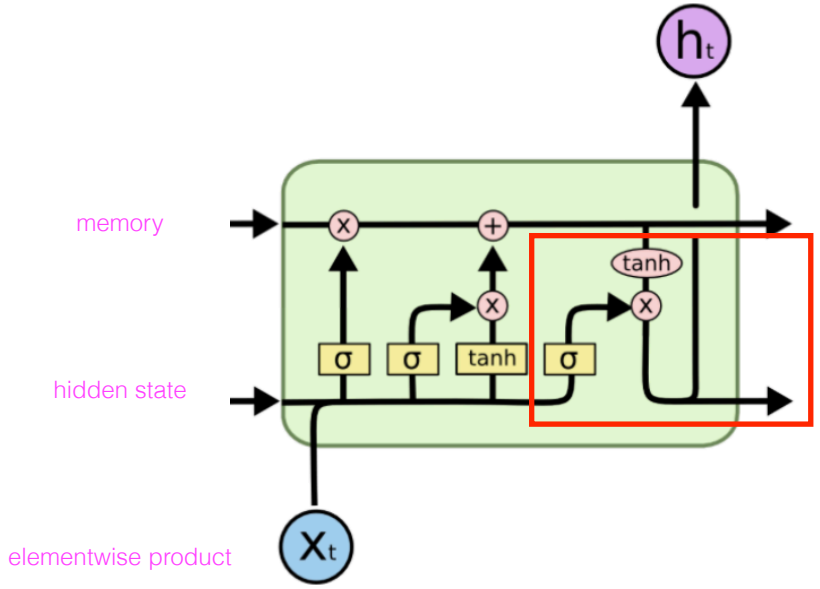
The memory passes directly to the next state



Output gate: forget some information to send to the hidden state

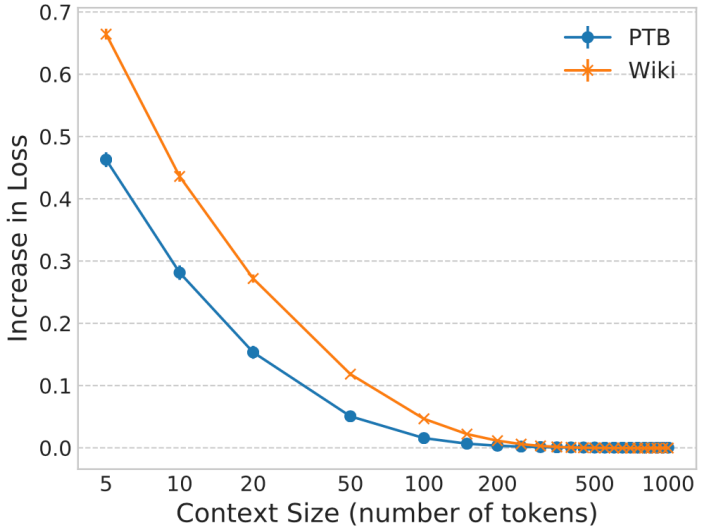


The hidden state is updated with the current observation and new context.



How much context?

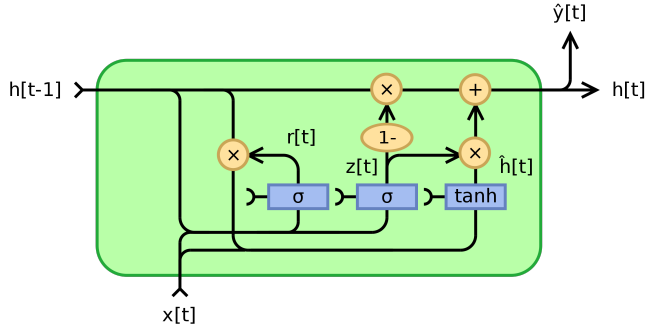
- For language modeling, LSTMs are aware of about 200 words of context
- Ignores word order beyond 50 words



Khandelwal et al. (2018), “Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context” (ACL)

GRU

- A gated recurrent unit adopts the same gating mechanism as an LSTM, but reduces the number of parameters to learn.



- Only one context vector (not a separate memory and hidden state vector) gets passed between timesteps.

- 2 gates (reset and update) instead of 3.

Neural sequence labeling

- Large design space for exploration in these models:
    - RNN/LSTM/GRU
    - Stacking
    - Hidden dimension size
    - Training with dropout and other forms of regularization.

LSTM/RNN

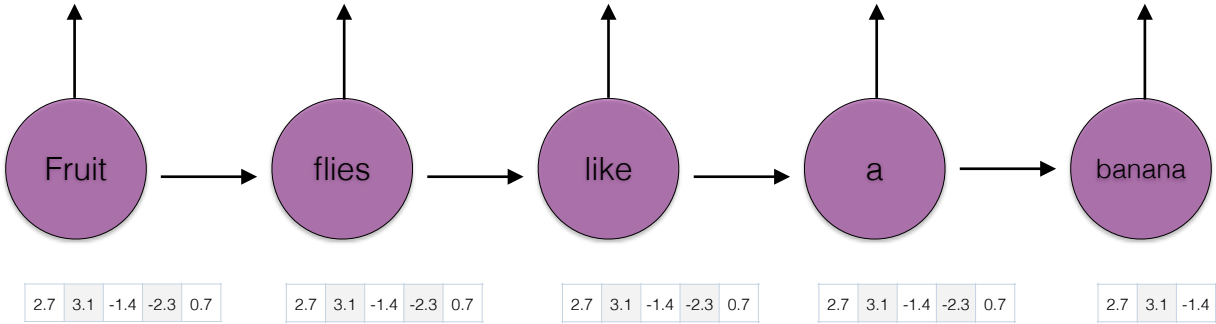
- Is an RNN the same kind of sequence labeling model as an MEMM or CRF?
- It doesn't use nearby labels in making predictions! (More like logistic regression in this respect)

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

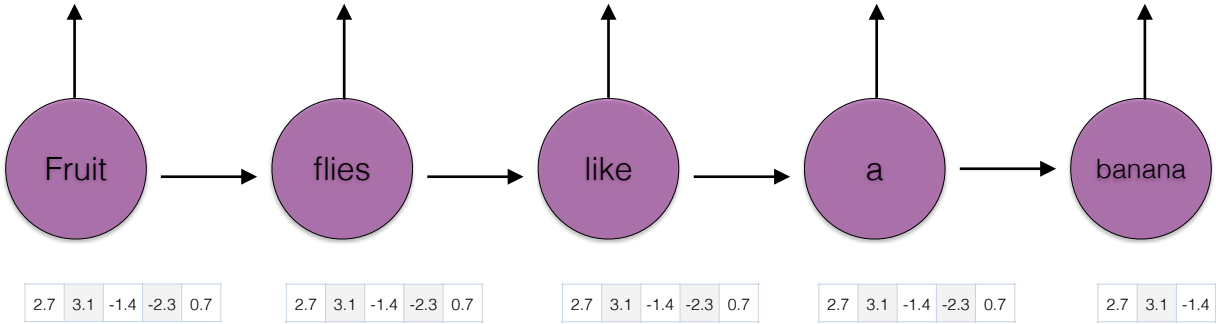
VBZ	0.51
NNS	0.48
JJ	0.01
NN	0
…	…

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



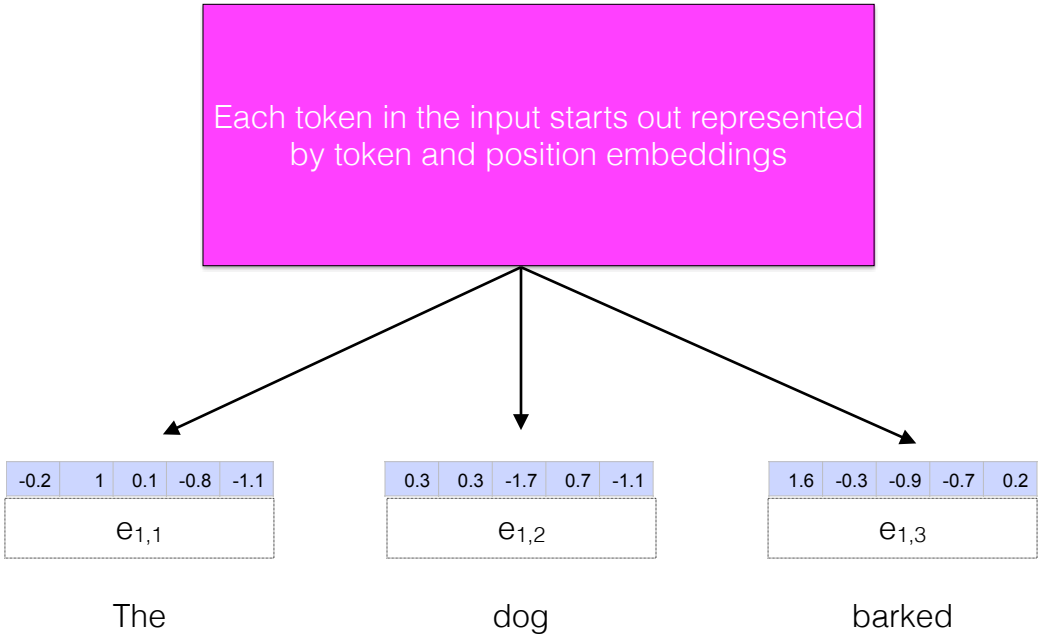
VBZ	0.51
NNS	0.48
JJ	0.01
NN	0
…	…

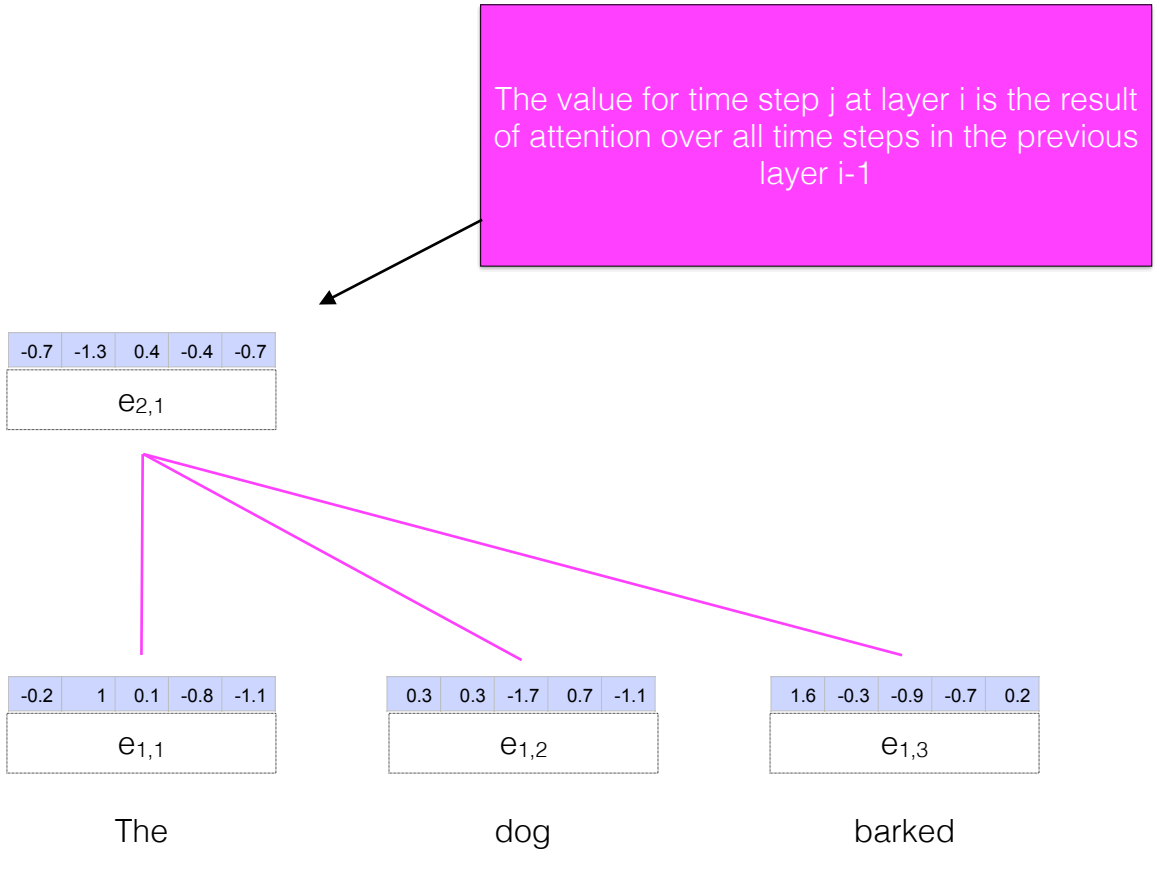
If we knew the categorical choice of VBZ at t_2, P(VB) at t_3 would be much lower.

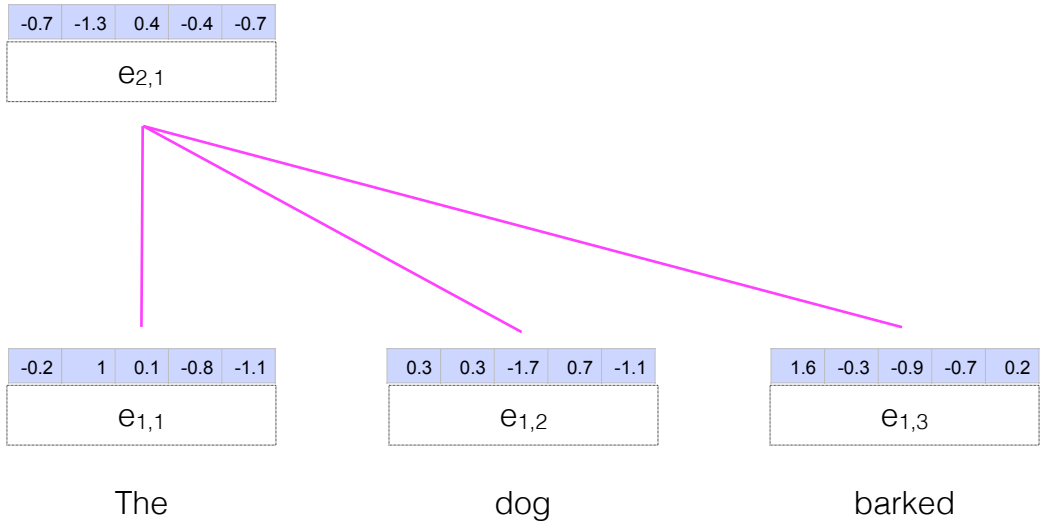


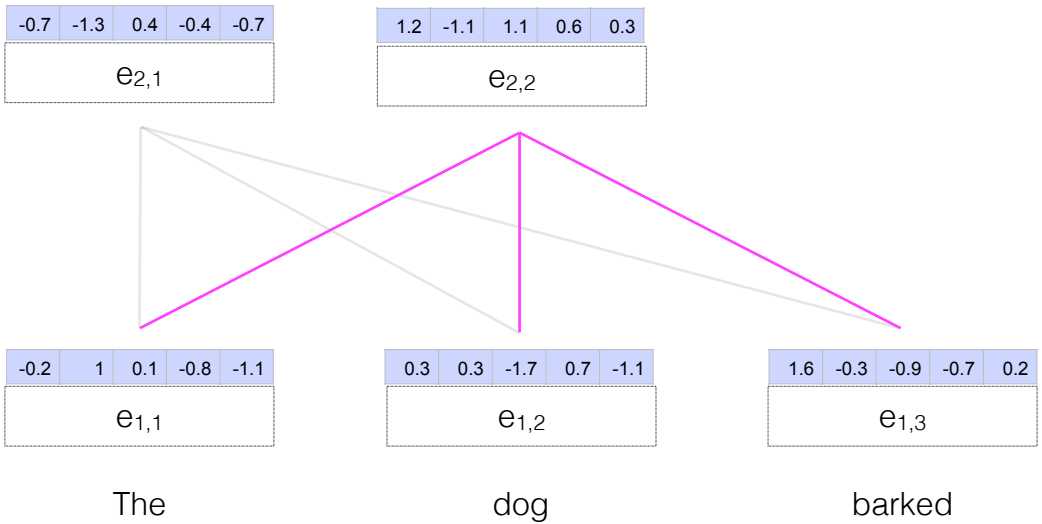
BERT

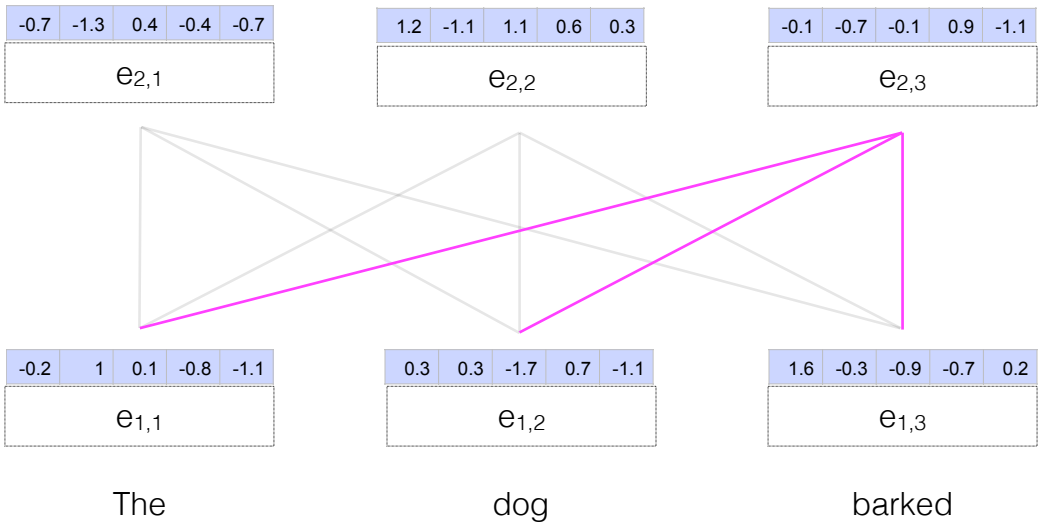
- Transformer-based model (Vaswani et al. 2017) to predict masked word using bidirectional context + next sentence prediction.
- Generates multiple layers of representations for each token sensitive to its context of use.

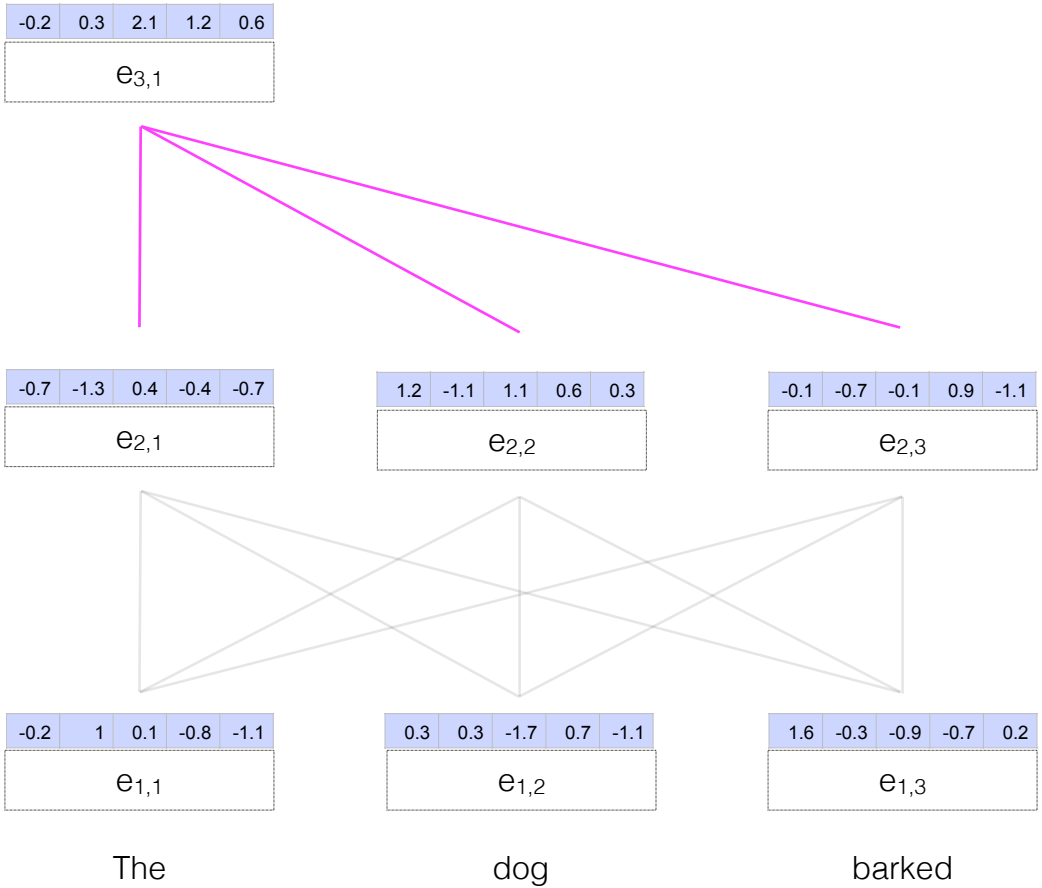

Each token in the input starts out represented by token and position embeddings
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

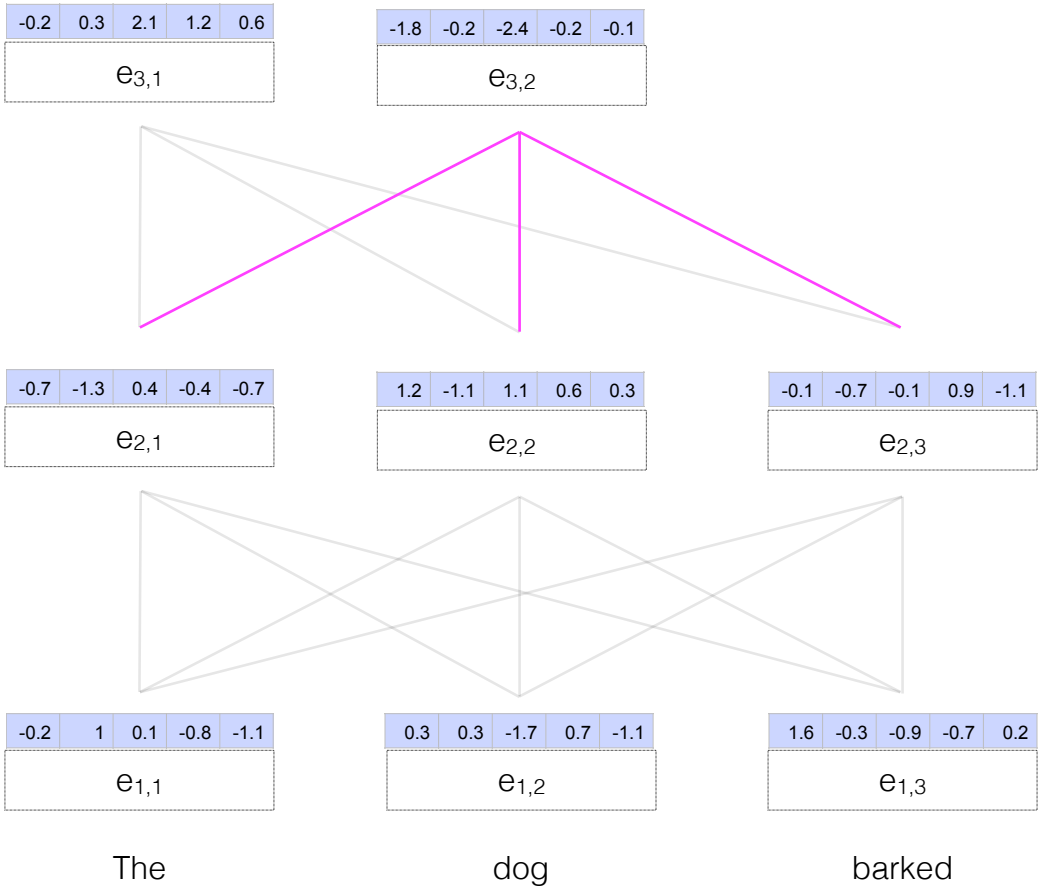

The value for time step j at layer i is the result of attention over all time steps in the previous layer i-1
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

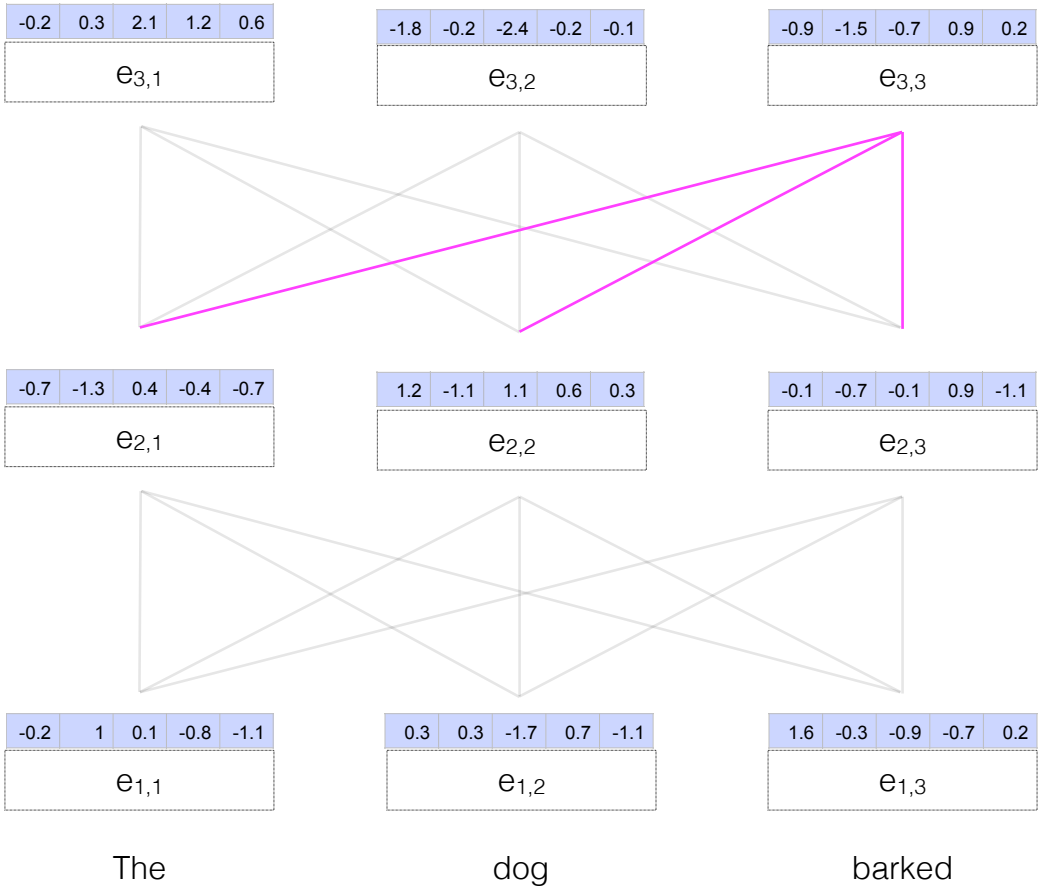

-0.7 -1.3 0.4 -0.4 -0.7
e2,1
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked


-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

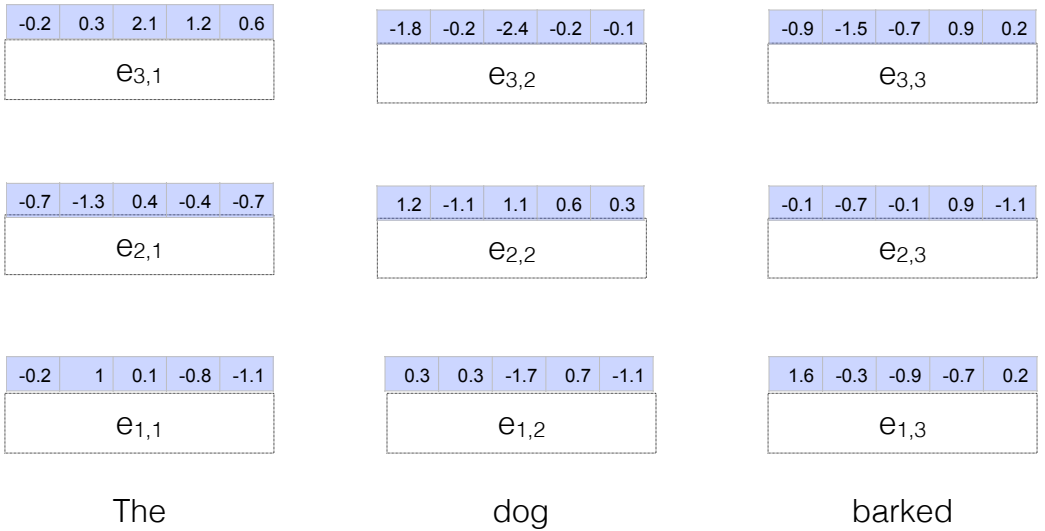

-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
e3,1
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
e3,1
-1.8 -0.2 -2.4 -0.2 -0.1
e3,2
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked


-0.2 0.3 2.1 1.2 0.6
e3,1
-1.8 -0.2 -2.4 -0.2 -0.1
e3,2
-0.9 -1.5 -0.7 0.9 0.2
e3,3
-0.7 -1.3 0.4 -0.4 -0.7
e2,1
1.2 -1.1 1.1 0.6 0.3
e2,2
-0.1 -0.7 -0.1 0.9 -1.1
e2,3
-0.2 1 0.1 -0.8 -1.1
e1,1
0.3 0.3 -1.7 0.7 -1.1
e1,2
1.6 -0.3 -0.9 -0.7 0.2
e1,3
The
dog
barked

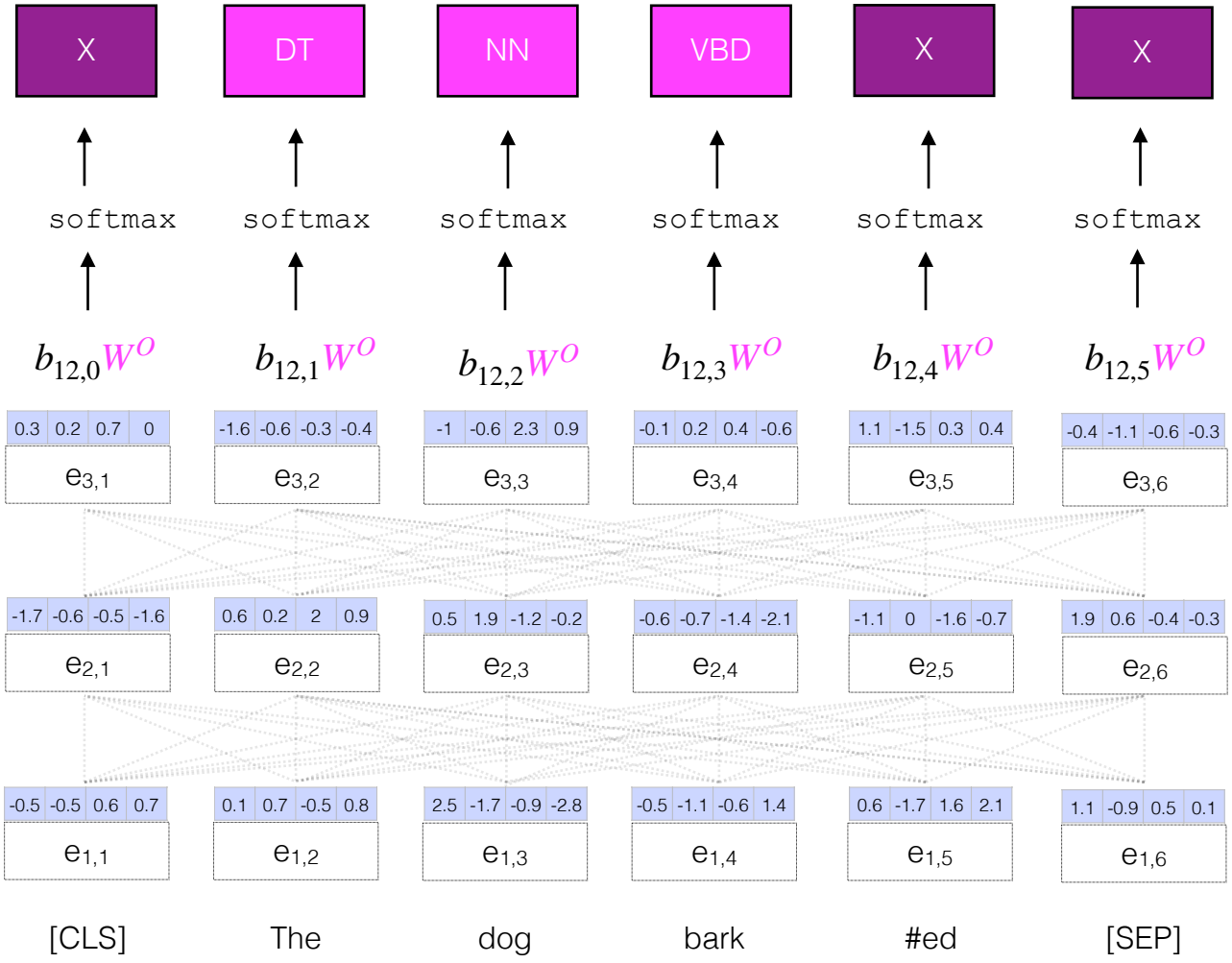
At the end of this process, we have one representation for each layer for each token



BERT

- BERT can be used not only as a language model to generate contextualized word representations, but also as a predictive model whose parameters are fine-tuned to a task.


neutral
softmax
$b_{12,0}W^o$
0.3 0.2 0.7 0
$e_{3,1}$
-1.6 -0.6 -0.3 -0.4
$e_{3,2}$
-1 -0.6 2.3 0.9
$e_{3,3}$
-0.1 0.2 0.4 -0.6
$e_{3,4}$
1.1 -1.5 0.3 0.4
$e_{3,5}$
-0.4 -1.1 -0.6 -0.3
$e_{3,6}$
-1.7 -0.6 -0.5 -1.6
$e_{2,1}$
0.6 0.2 2 0.9
$e_{2,2}$
0.5 1.9 -1.2 -0.2
$e_{2,3}$
-0.6 -0.7 -1.4 -2.1
$e_{2,4}$
-1.1 0 -1.6 -0.7
$e_{2,5}$
1.9 0.6 -0.4 -0.3
$e_{2,6}$
-0.5 -0.5 0.6 0.7
$e_{1,1}$
0.1 0.7 -0.5 0.8
$e_{1,2}$
2.5 -1.7 -0.9 -2.8
$e_{1,3}$
-0.5 -1.1 -0.6 1.4
$e_{1,4}$
0.6 -1.7 1.6 2.1
$e_{1,5}$
1.1 -0.9 0.5 0.1
$e_{1,6}$
[CLS]
The
dog
bark
#ed
[SEP]


X
DT
NN
VBD
X
X
softmax
softmax
softmax
softmax
softmax
softmax
$b_{12,0}W^O$
$b_{12,1}W^O$
$b_{12,2}W^O$
$b_{12,3}W^O$
$b_{12,4}W^O$
$b_{12,5}W^O$
0.3 0.2 0.7 0
$e_{3,1}$
-1.6 -0.6 -0.3 -0.4
$e_{3,2}$
-1 -0.6 2.3 0.9
$e_{3,3}$
-0.1 0.2 0.4 -0.6
$e_{3,4}$
1.1 -1.5 0.3 0.4
$e_{3,5}$
-0.4 -1.1 -0.6 -0.3
$e_{3,6}$
-1.7 -0.6 -0.5 -1.6
$e_{2,1}$
0.6 0.2 2 0.9
$e_{2,2}$
0.5 1.9 -1.2 -0.2
$e_{2,3}$
-0.6 -0.7 -1.4 -2.1
$e_{2,4}$
-1.1 0 -1.6 -0.7
$e_{2,5}$
1.9 0.6 -0.4 -0.3
$e_{2,6}$
-0.5 -0.5 0.6 0.7
$e_{1,1}$
0.1 0.7 -0.5 0.8
$e_{1,2}$
2.5 -1.7 -0.9 -2.8
$e_{1,3}$
-0.5 -1.1 -0.6 1.4
$e_{1,4}$
0.6 -1.7 1.6 2.1
$e_{1,5}$
1.1 -0.9 0.5 0.1
$e_{1,6}$
[CLS]
The
dog
bark
#ed
[SEP]

BERT

- Pre-training: train BERT through masked language modeling and next-sentence prediction to learn the parameters of BERT layers. Trained on Wikipedia + BookCorpus.

- Task fine-tuning: add additional linear transformation + softmax to get distribution over output space. Trained on annotated data.