



Natural Language Processing

Info 159/259

Lecture 11: Neural sequence labeling (Feb 21, 2023)

David Bamman, UC Berkeley

Sequence labeling

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

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

- 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 .

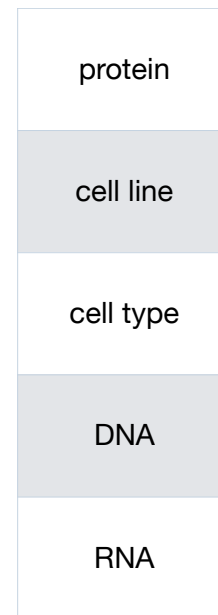
Figure 17.1 A list of generic named entity types with the kinds of entities they refer to.

ACE NER categories (+weapon)

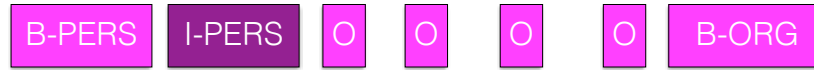
Named entity recognition

- GENIA corpus of MEDLINE abstracts (biomedical)

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



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

B-PER

B-PER

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
<i>gold</i>	B-PER	I-PER	O	O	O	O	B-ORG
<i>system</i>	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

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

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

- 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.

$$P(x, y) = P(y) P(x | y)$$

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

$$P(y | x)$$

MEMM

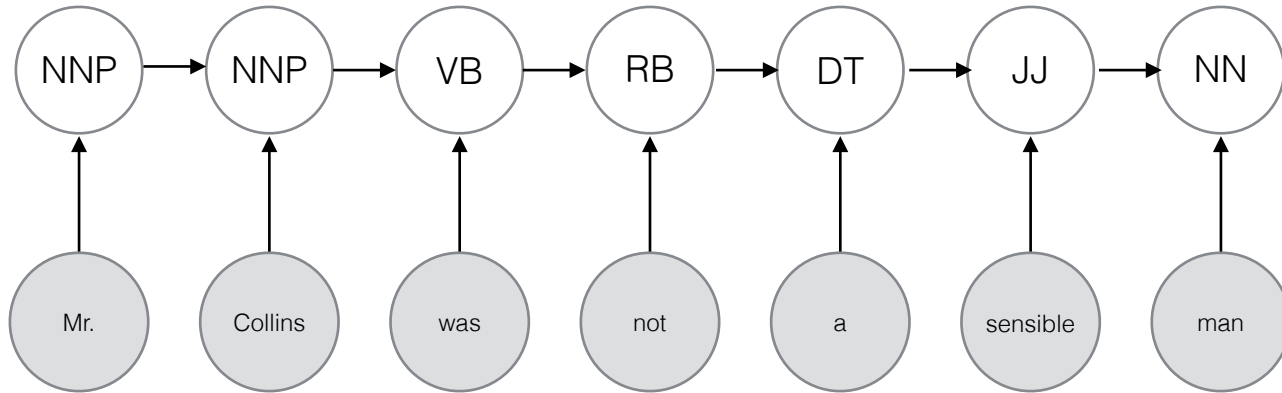
General maxent form

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

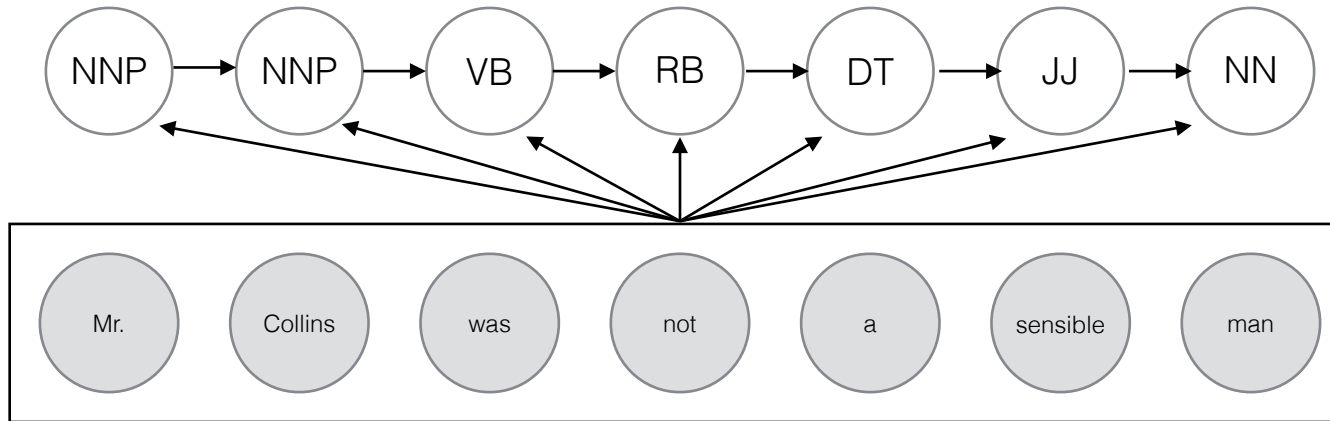
Maxent with first-order Markov
assumption: Maximum Entropy
Markov Model

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

MEMM

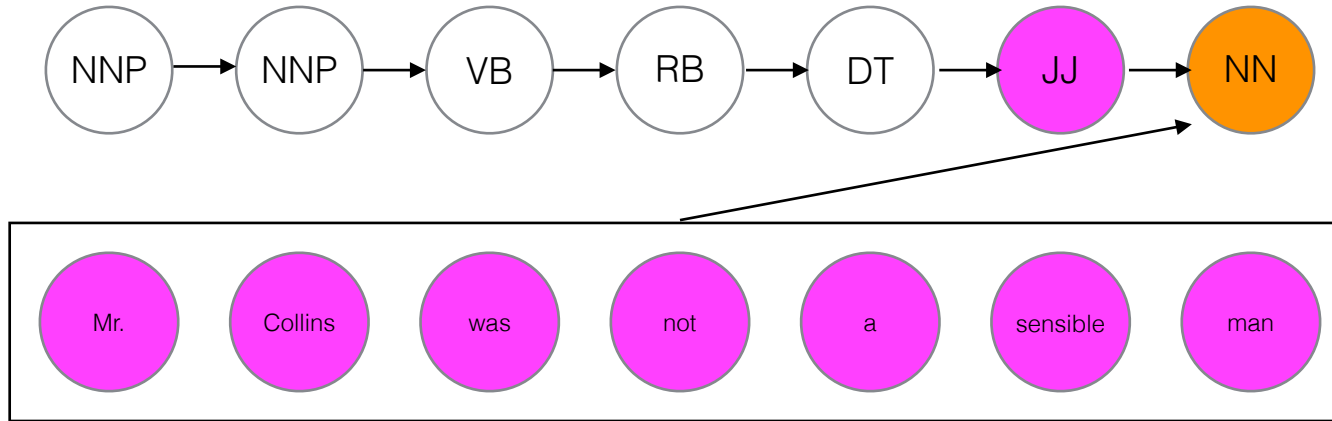


MEMM



MEMMs condition on the *entire* input

MEMM



Features

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

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

Training

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

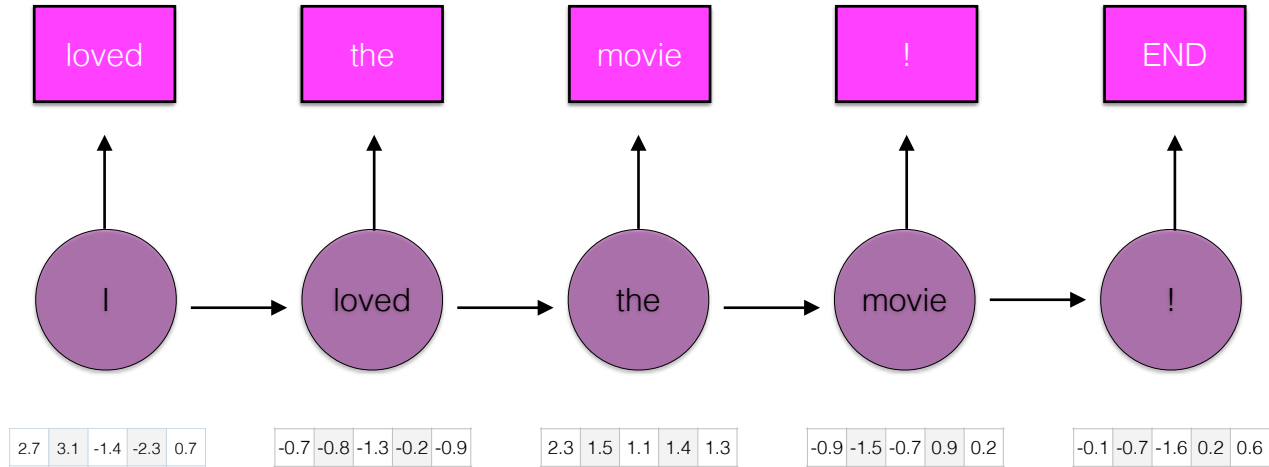
For all training data, we want probability of the true label y_i conditioned on the previous true label y_{i-1} to be high.

This is simply multiclass logistic regression

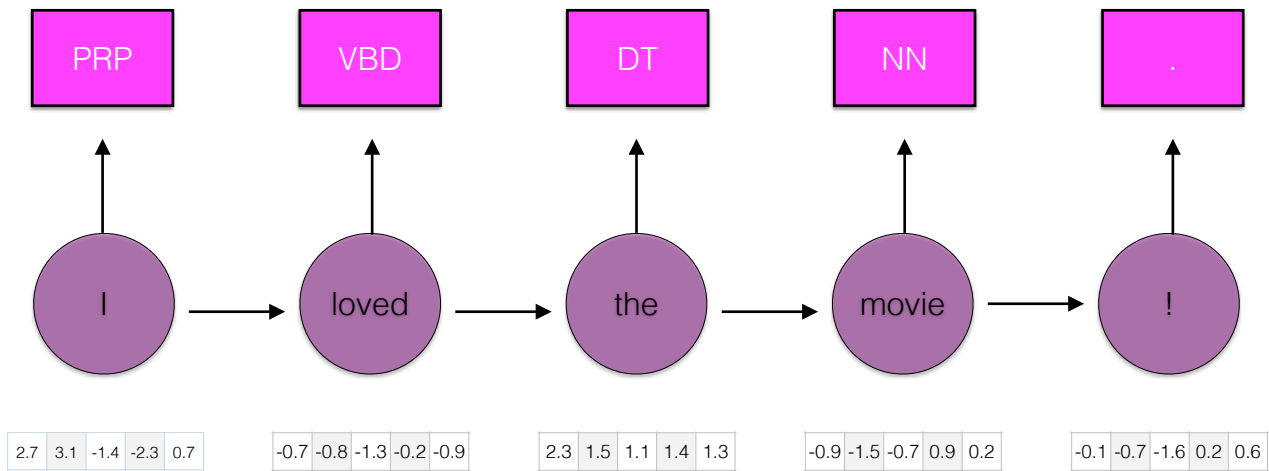
Recurrent neural network

- RNNs allow arbitrarily-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 \mathcal{V} conditioned on the context



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



RNNs for POS

NN TO VB

will to fight

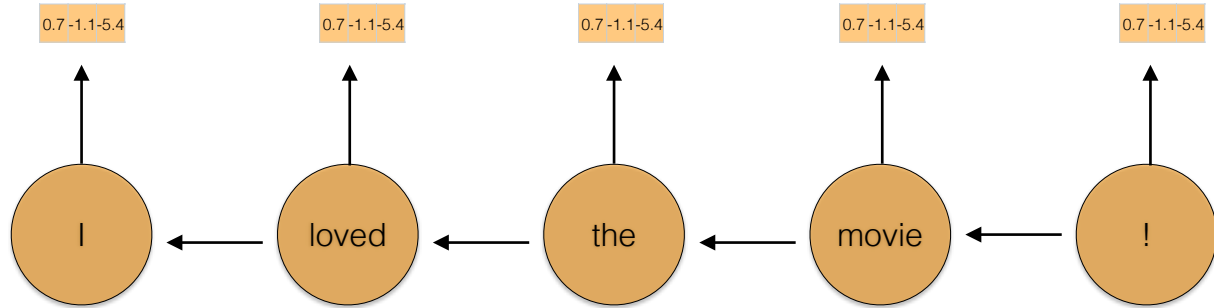
- To make a prediction for y_t , RNNs condition on all input seen through time t (x_1, \dots, x_t)
- But knowing something about the future can help (x_{t+1}, \dots, x_n)

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

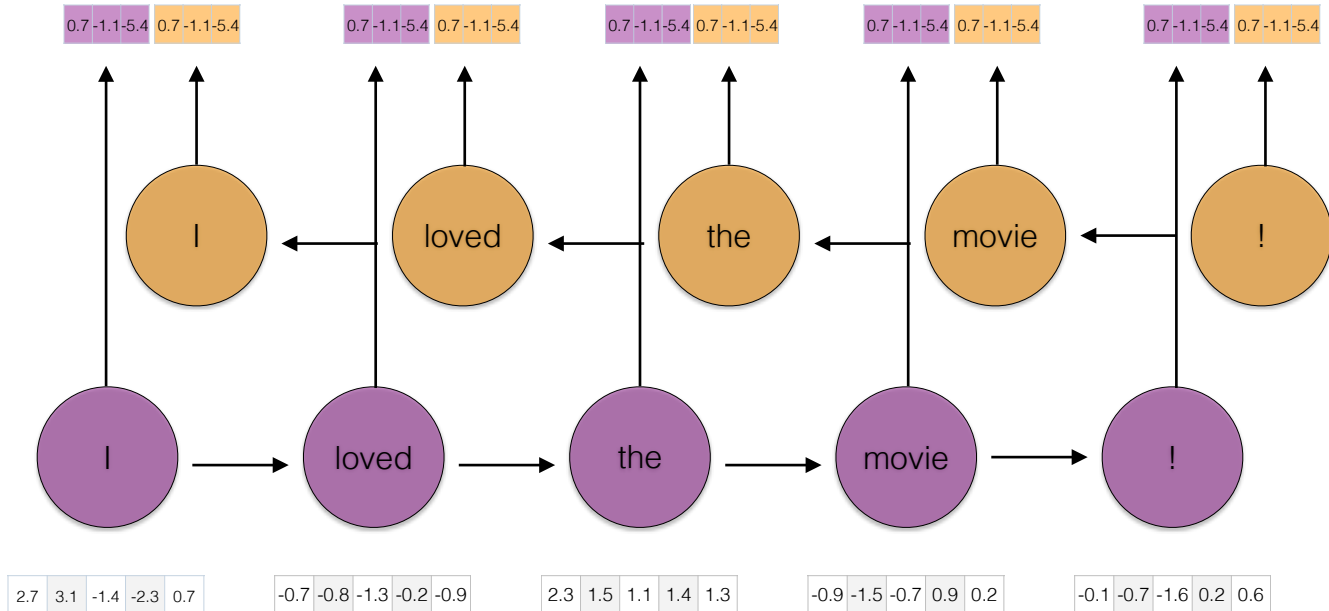
-0.7 -0.8 -1.3 -0.2 -0.9

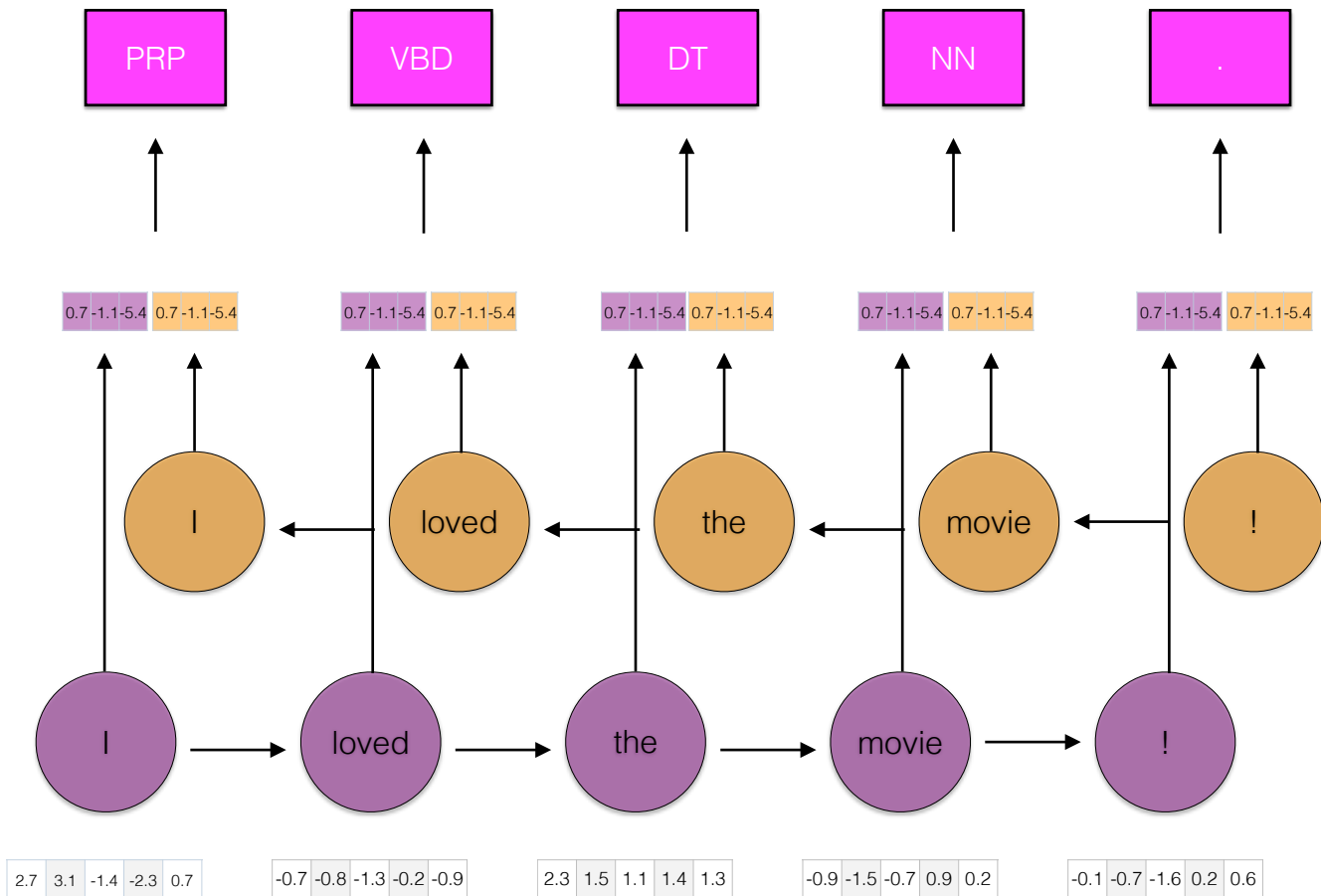
2.3 1.5 1.1 1.4 1.3

-0.9 -1.5 -0.7 0.9 0.2

-0.1 -0.7 -1.6 0.2 0.6

Bidirectional RNN

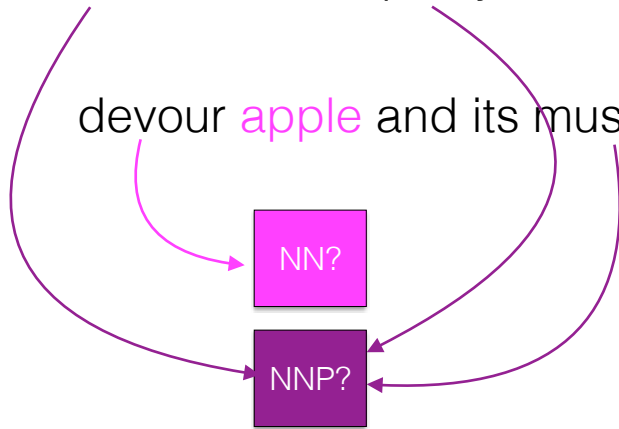




RNNs for POS tagging

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?

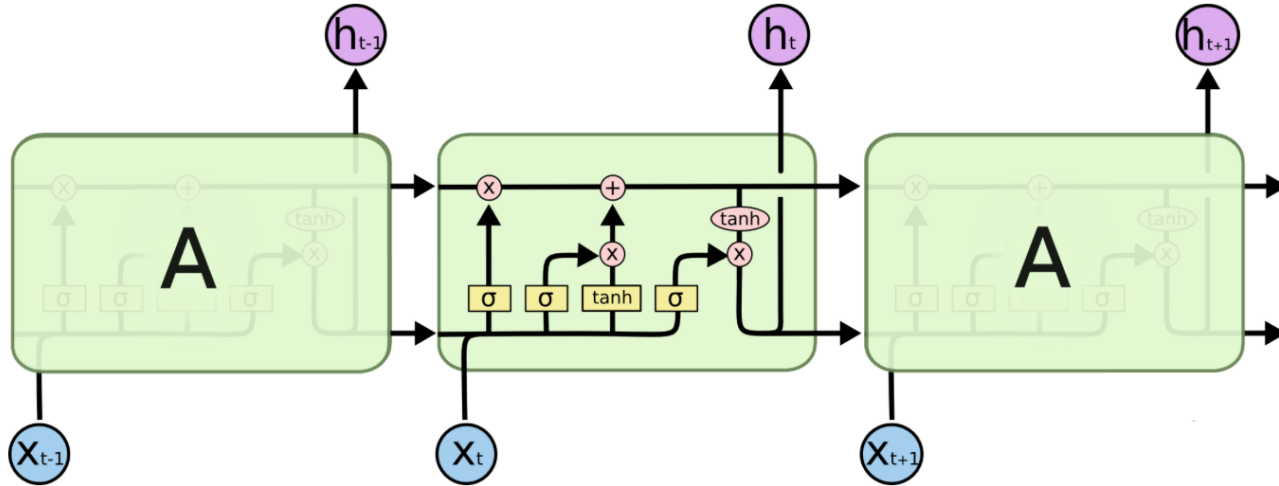
RNNs

- 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)

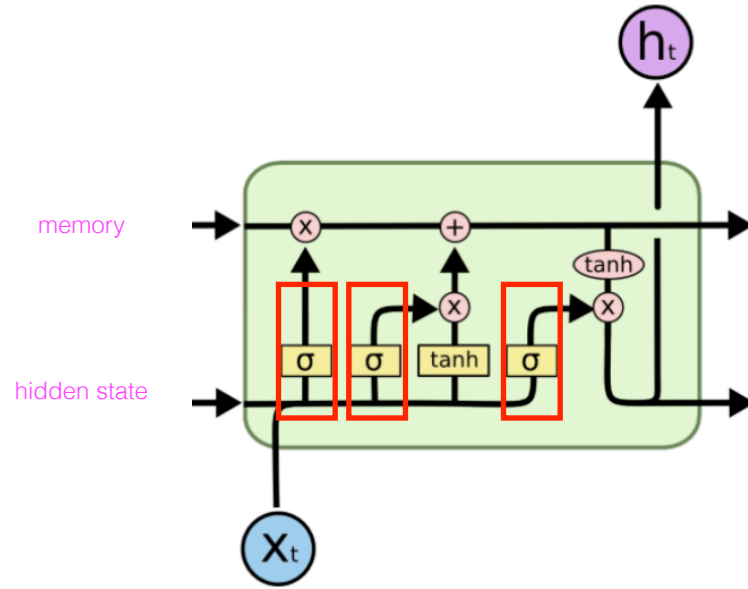
- Designed to account for the vanishing gradient problem
- Basic idea: split the s vector propagated between time steps into a **memory** component and a **hidden state** component

LSTMs



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
-----	-----	------	------	-----

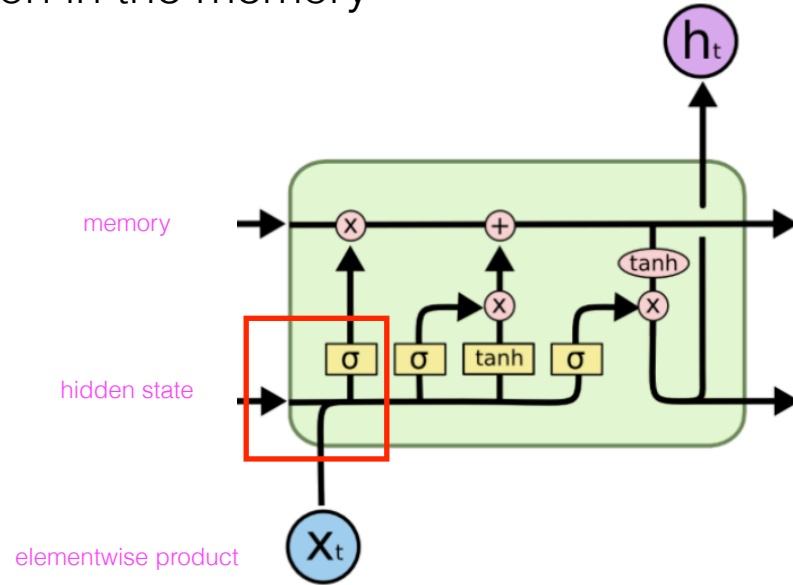
gate

0.01	0.99	0.5	0.98	0.01
------	------	-----	------	------

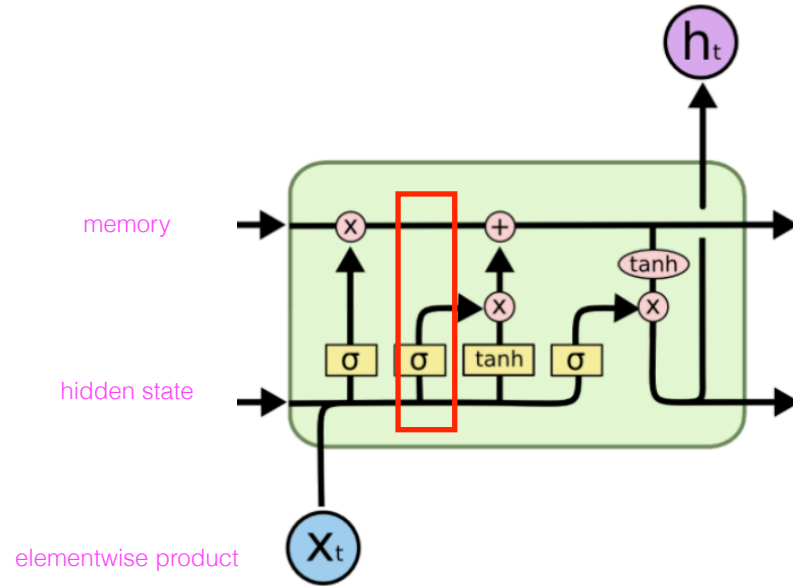
output

0.03	1.4	-0.35	-1.38	0.08
------	-----	-------	-------	------

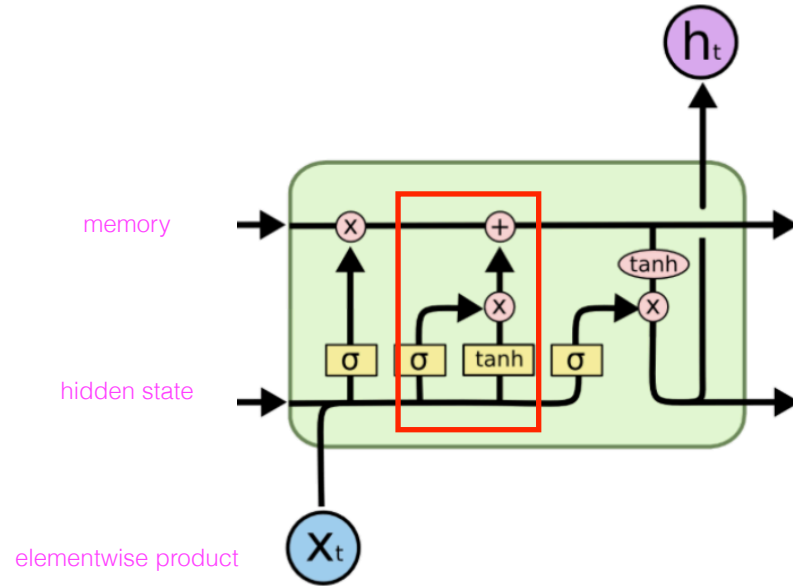
Forget gate: as a function of the previous hidden state and current input, forget information in the memory



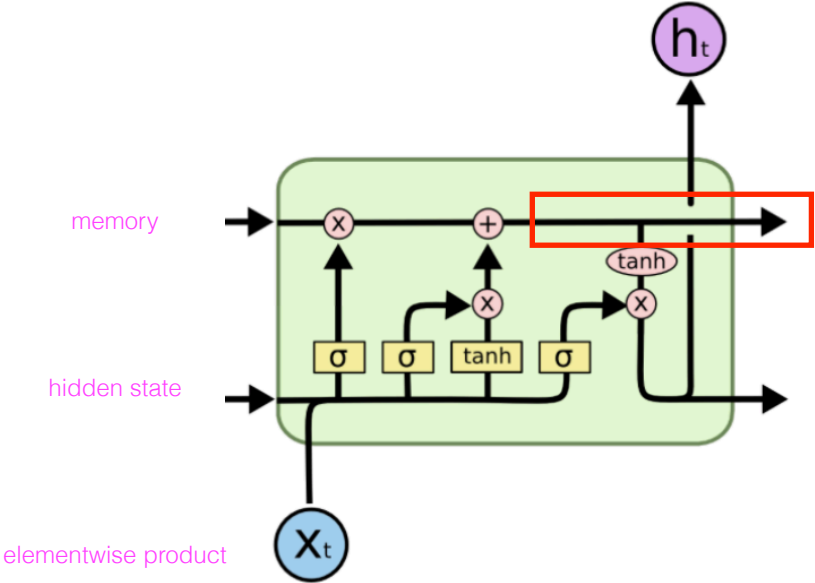
Input gate (but forget some information about the current observation)



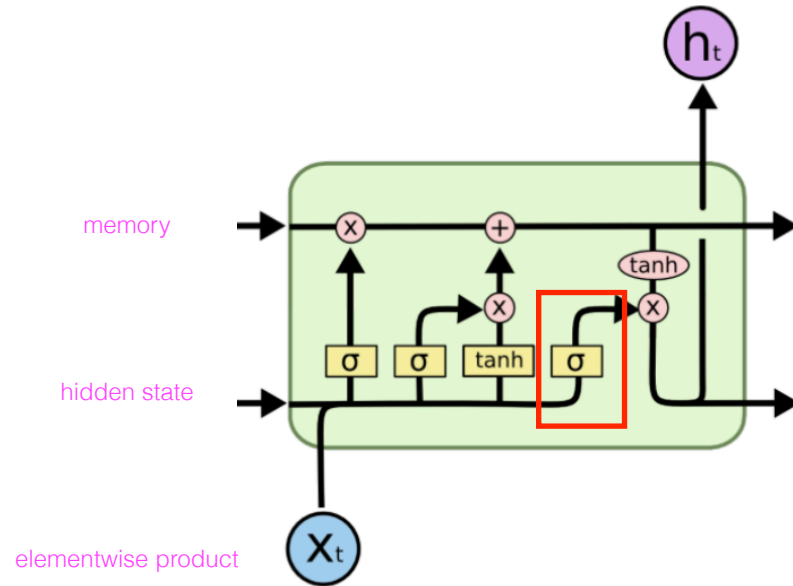
Update the memory (but forget some information about the current observation)



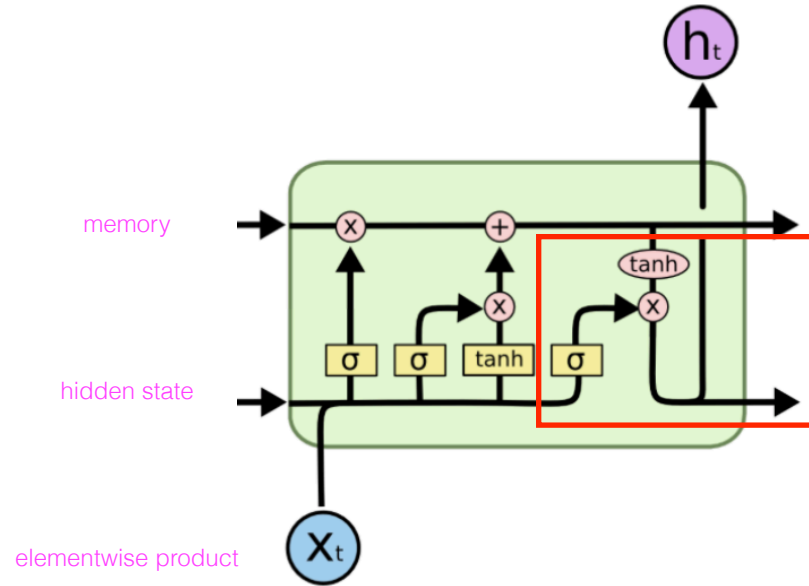
The memory passes directly to the next state



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

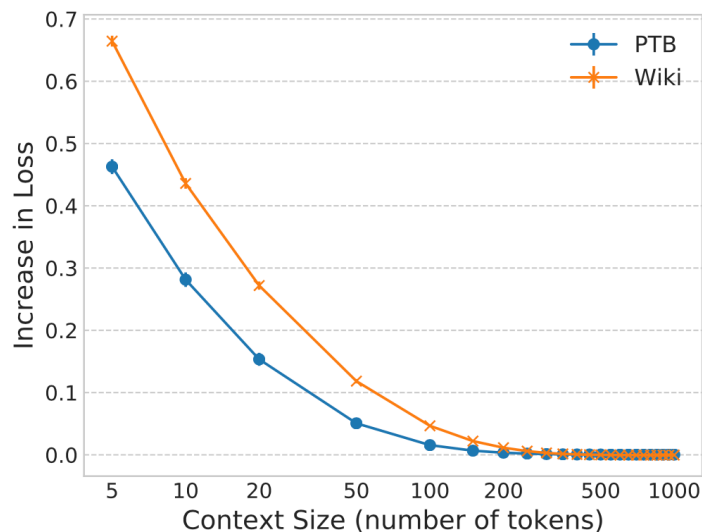


The hidden state is updated with the current observation and new context.



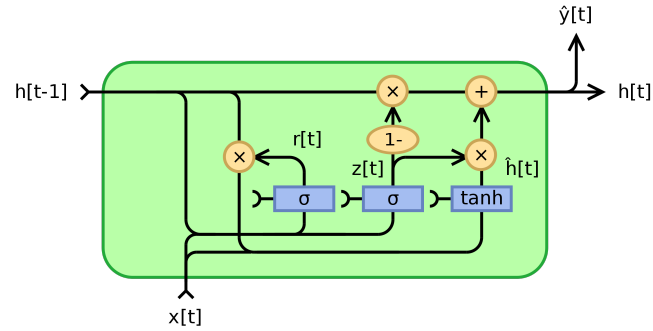
How much context?

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



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 HMM or MEMM?
- 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	$\prod_{i=1}^N P(x_i y_i) P(y_i y_{i-1})$	Markov assumption	no
MEMM	$\prod_{i=1}^N P(y_i y_{i-1}, x, \beta)$	Markov assumption	yes
RNN	$\prod_{i=1}^N P(y_i x_{1:i}, \beta)$	none	distributed

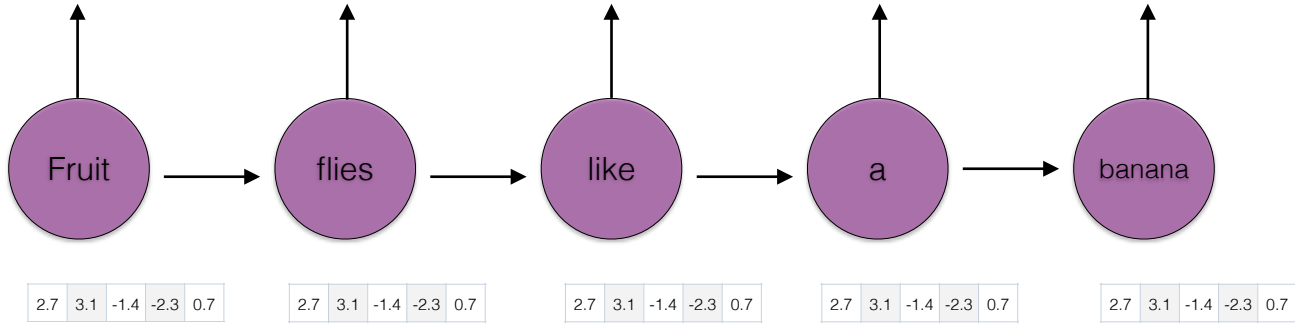
NN

VBZ

VB

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.



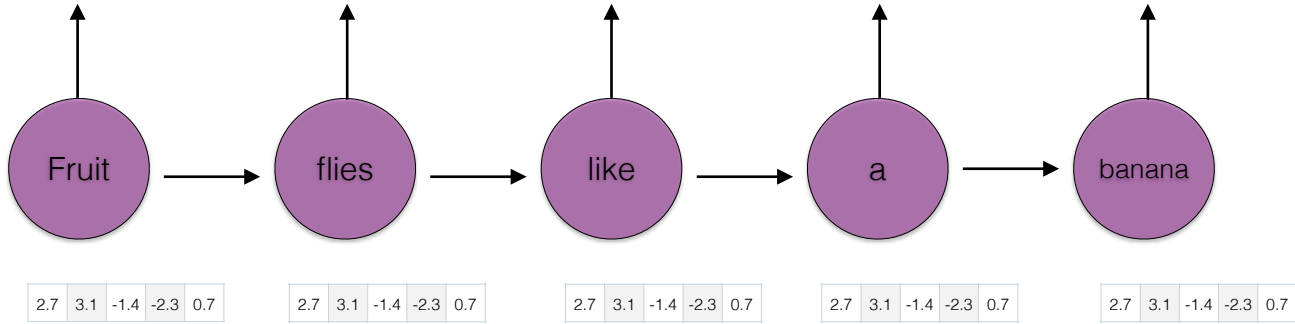
NN

VBZ

VB

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
$e_{1,1}$				

The

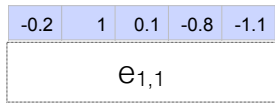
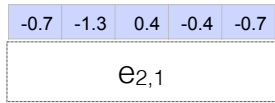
0.3	0.3	-1.7	0.7	-1.1
$e_{1,2}$				

dog

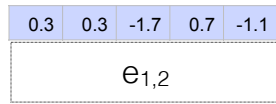
1.6	-0.3	-0.9	-0.7	0.2
$e_{1,3}$				

barked

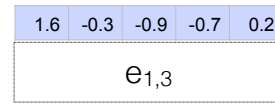
The value for time step j at layer i is the result of attention over all time steps in the previous layer $i-1$



The



dog



barked

-0.7	-1.3	0.4	-0.4	-0.7
$e_{2,1}$				

-0.2	1	0.1	-0.8	-1.1
$e_{1,1}$				

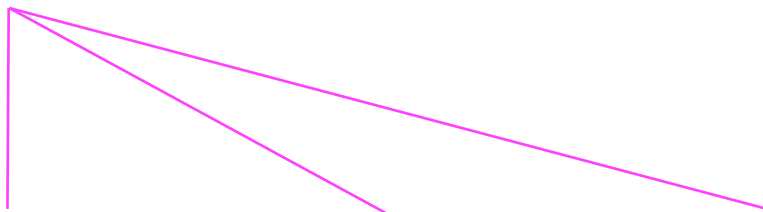
0.3	0.3	-1.7	0.7	-1.1
$e_{1,2}$				

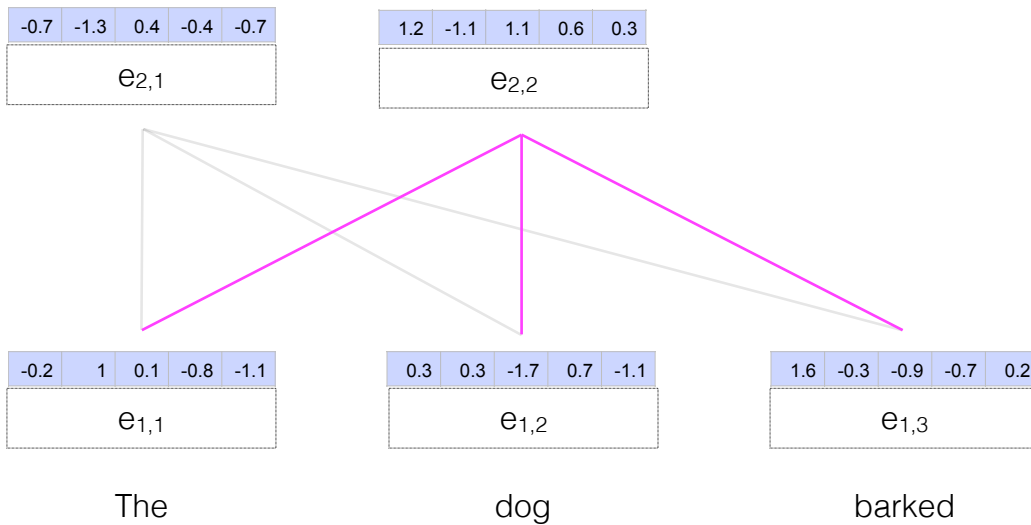
1.6	-0.3	-0.9	-0.7	0.2
$e_{1,3}$				

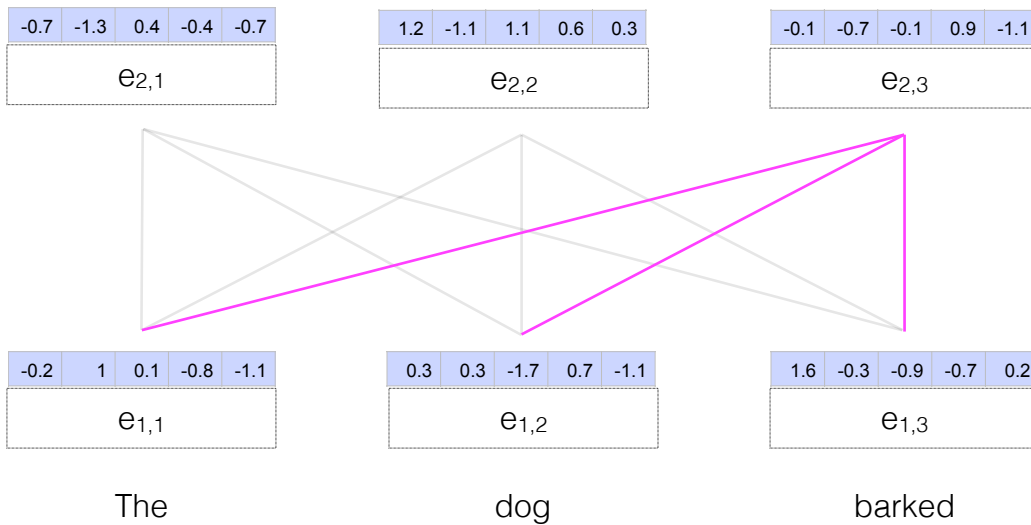
The

dog

barked







-0.2	0.3	2.1	1.2	0.6
$e_{3,1}$				



-0.7	-1.3	0.4	-0.4	-0.7
$e_{2,1}$				

1.2	-1.1	1.1	0.6	0.3
$e_{2,2}$				

-0.1	-0.7	-0.1	0.9	-1.1
$e_{2,3}$				



-0.2	1	0.1	-0.8	-1.1
$e_{1,1}$				

0.3	0.3	-1.7	0.7	-1.1
$e_{1,2}$				

1.6	-0.3	-0.9	-0.7	0.2
$e_{1,3}$				

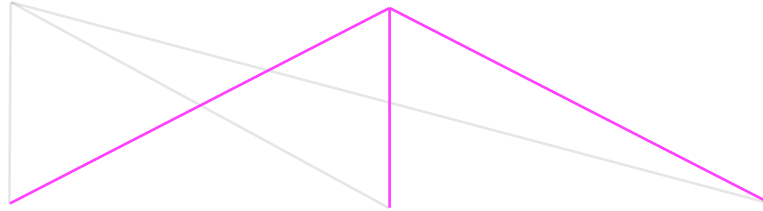
The

dog

barked

-0.2	0.3	2.1	1.2	0.6
$e_{3,1}$				

-1.8	-0.2	-2.4	-0.2	-0.1
$e_{3,2}$				



-0.7	-1.3	0.4	-0.4	-0.7
$e_{2,1}$				

1.2	-1.1	1.1	0.6	0.3
$e_{2,2}$				

-0.1	-0.7	-0.1	0.9	-1.1
$e_{2,3}$				



-0.2	1	0.1	-0.8	-1.1
$e_{1,1}$				

0.3	0.3	-1.7	0.7	-1.1
$e_{1,2}$				

1.6	-0.3	-0.9	-0.7	0.2
$e_{1,3}$				

The

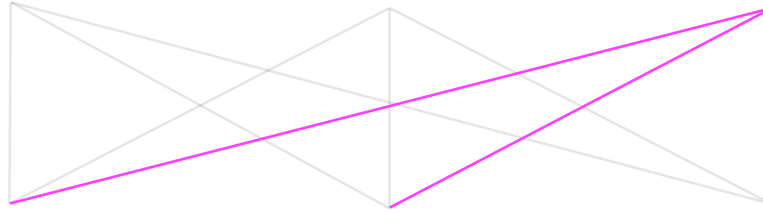
dog

barked

-0.2	0.3	2.1	1.2	0.6
$e_{3,1}$				

-1.8	-0.2	-2.4	-0.2	-0.1
$e_{3,2}$				

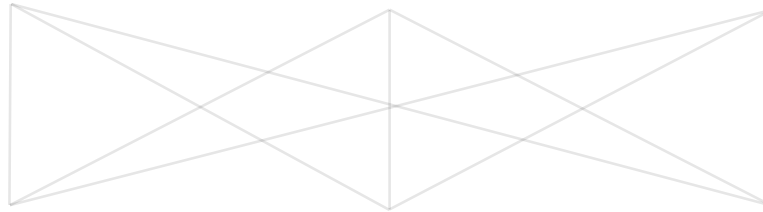
-0.9	-1.5	-0.7	0.9	0.2
$e_{3,3}$				



-0.7	-1.3	0.4	-0.4	-0.7
$e_{2,1}$				

1.2	-1.1	1.1	0.6	0.3
$e_{2,2}$				

-0.1	-0.7	-0.1	0.9	-1.1
$e_{2,3}$				



-0.2	1	0.1	-0.8	-1.1
$e_{1,1}$				

0.3	0.3	-1.7	0.7	-1.1
$e_{1,2}$				

1.6	-0.3	-0.9	-0.7	0.2
$e_{1,3}$				

The

dog

barked

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

-0.2	0.3	2.1	1.2	0.6
e _{3,1}				

-1.8	-0.2	-2.4	-0.2	-0.1
e _{3,2}				

-0.9	-1.5	-0.7	0.9	0.2
e _{3,3}				

-0.7	-1.3	0.4	-0.4	-0.7
e _{2,1}				

1.2	-1.1	1.1	0.6	0.3
e _{2,2}				

-0.1	-0.7	-0.1	0.9	-1.1
e _{2,3}				

-0.2	1	0.1	-0.8	-1.1
e _{1,1}				

0.3	0.3	-1.7	0.7	-1.1
e _{1,2}				

1.6	-0.3	-0.9	-0.7	0.2
e _{1,3}				

The

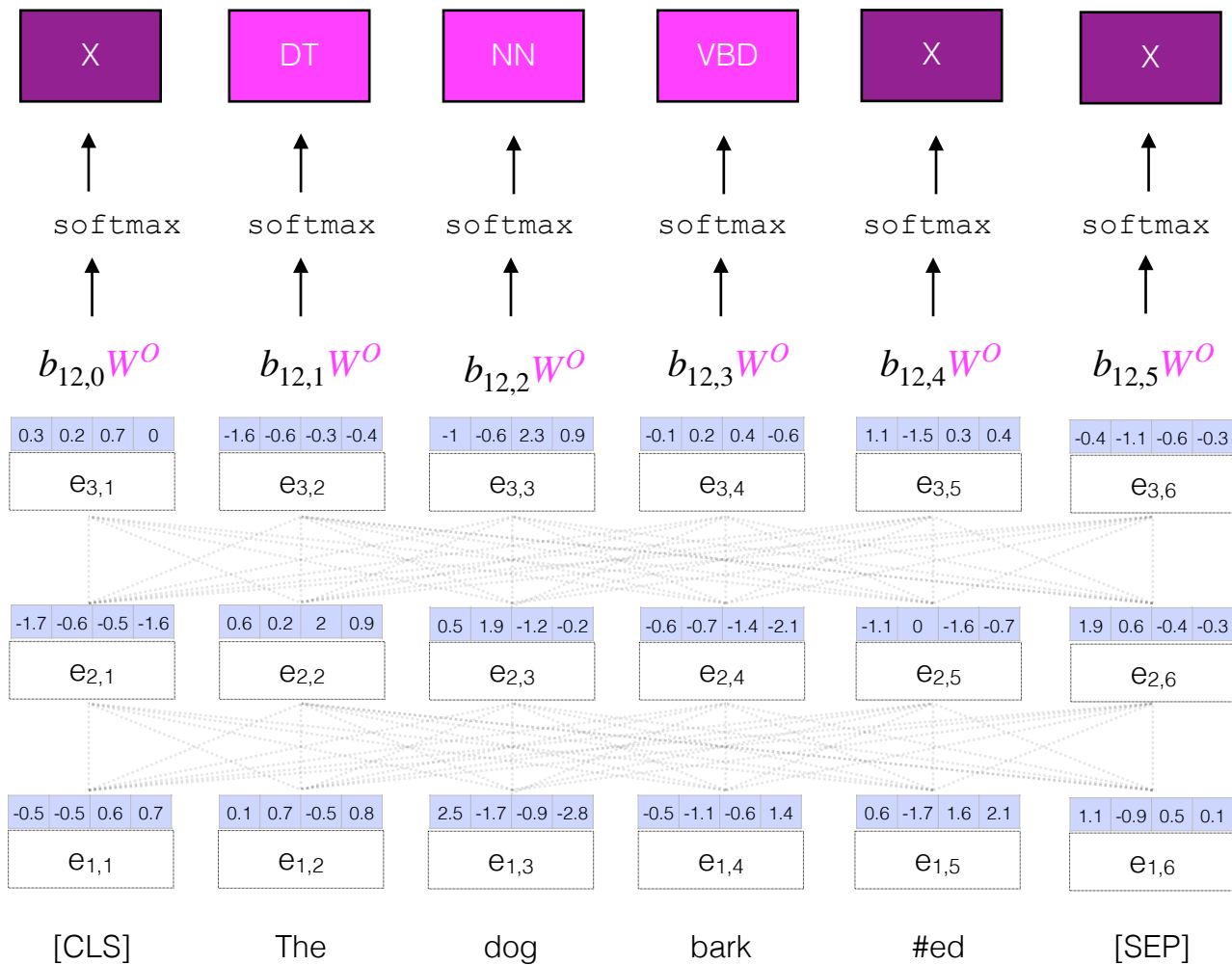
dog

barked

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**.





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.