

**10707**

# **Deep Learning**

Russ Salakhutdinov

Machine Learning Department

[rsalakhu@cs.cmu.edu](mailto:rsalakhu@cs.cmu.edu)

# Language Modeling

# Natural Language Processing

- Natural language processing is concerned with tasks involving language data
  - we will focus on text data NLP
- Much like for computer vision, we can design neural networks specifically adapted to the processing of text data
  - main issue: text data is inherently **high dimensional**

# Natural Language Processing

- Typical preprocessing steps of text data
  - Form vocabulary of words that maps words to a unique ID
  - Different criteria can be used to select which words are part of the vocabulary
  - Pick **most frequent words** and ignore **uninformative** words from a user-defined short list (ex.: “the”, “a”, etc.)
  - All words not in the vocabulary will be mapped to a special “**out-of-vocabulary**”
- Typical vocabulary sizes will vary between 100,000 and 1,000,000

# Vocabulary

- Example:

"the"
"cat"
"and"
"the"
"dog"
"play"
"."

Word	<i>w</i>
"the"	1
"and"	2
"dog"	3
"."	4
"oov"	5



1
5
2
1
3
5
4

- We will note word IDs with the symbol  $w$

- we can think of  $w$  as a **categorical feature** for the original word
- we will sometimes refer to  $w$  as a word, for simplicity

# One-Hot Encoding

- From its word ID, we get a basic representation of a word through the **one-hot encoding** of the ID
  - the **one-hot vector** of an ID is a vector filled with 0s, except for a 1 at the position associated with the ID
  - For vocabulary size D=10, the one-hot vector of word ID w=4 is:
$$e(w) = [ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 ]$$
- A one-hot encoding makes no assumption about **word similarity**
- This is a natural representation to start with, though a poor one

# One-Hot Encoding

- The major problem with the one-hot representation is that it is very high-dimensional
  - the dimensionality of  $e(w)$  is the size of the vocabulary
  - a typical vocabulary size is  $\approx 100,000$
  - a window of 10 words would correspond to an input vector of **at least 1,000,000 units!**
- This has 2 consequences:
  - vulnerability to **overfitting** (millions of inputs means millions of parameters to train)
  - computationally **expensive**

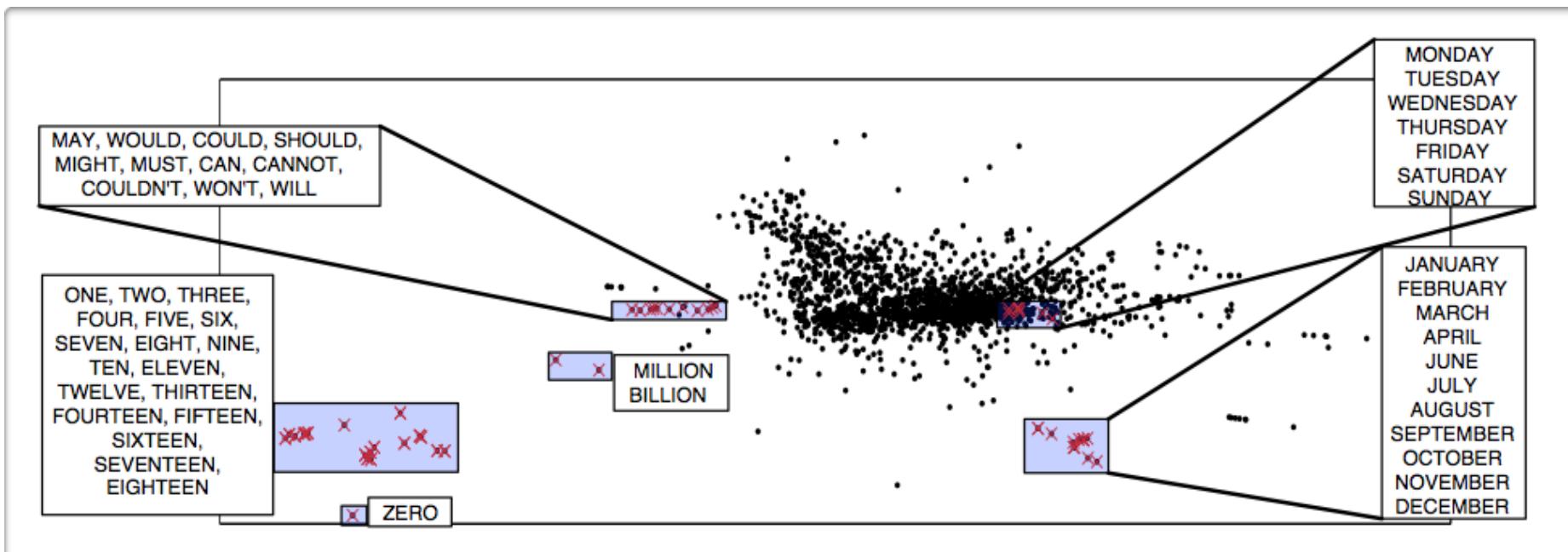
# Continuous Representation of Words

- Each word  $w$  is associated with a real-valued vector  $C(w)$

Word	$w$	$C(w)$
" the "	1	[ 0.6762, -0.9607, 0.3626, -0.2410, 0.6636 ]
" a "	2	[ 0.6859, -0.9266, 0.3777, -0.2140, 0.6711 ]
" have "	3	[ 0.1656, -0.1530, 0.0310, -0.3321, -0.1342 ]
" be "	4	[ 0.1760, -0.1340, 0.0702, -0.2981, -0.1111 ]
" cat "	5	[ 0.5896, 0.9137, 0.0452, 0.7603, -0.6541 ]
" dog "	6	[ 0.5965, 0.9143, 0.0899, 0.7702, -0.6392 ]
" car "	7	[ -0.0069, 0.7995, 0.6433, 0.2898, 0.6359 ]
...	...	...

# Continuous Representation of Words

- We would like the distance  $\|C(w) - C(w')\|$  to reflect **meaningful similarities** between words



(from Blitzer et al. 2004)

# Continuous Representation of Words

- Learn a continuous representation of words
  - we could then use these representations as input to a neural network
- We learn these representations by **gradient descent**
  - we don't only update the neural network parameters
  - we also update **each representation  $C(w)$**  in the input  $x$  with a gradient step:

$$C(w) \leftarrow C(w) - \alpha \nabla_{C(w)} l$$

where  $l$  is the **loss function** optimized by the neural network

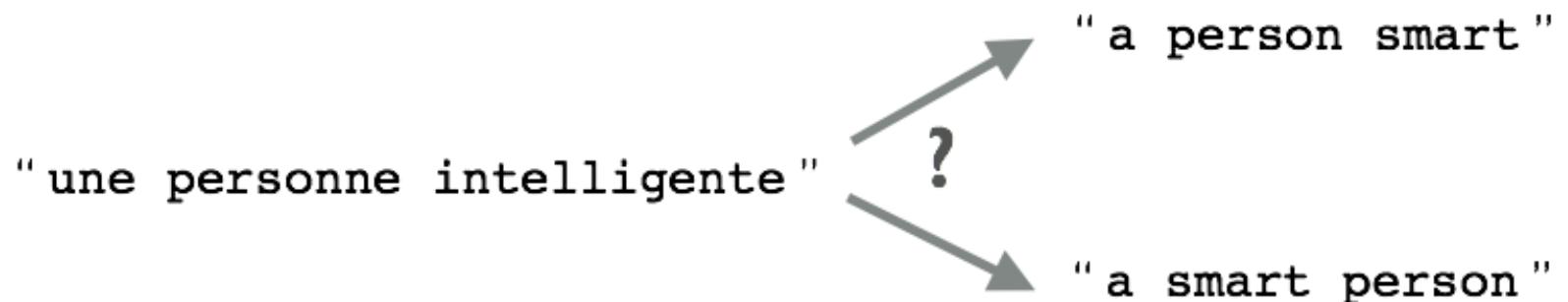
# Continuous Representation of Words

- Let  $C$  be a matrix whose rows are the representations  $C(w)$ 
  - obtaining  $C(w)$  corresponds to the multiplication  $e(w)^\top C$
  - view differently, we are **projecting**  $e(w)$  onto the columns of  $C$
  - this is a **continuous transformation**, through which we can propagate gradients
- In practice, we implement  $C(w)$  with a lookup table, not with a multiplication

# Language Modeling

$$p(w_1, \dots, w_T)$$

- language modeling is the task of learning a language model that assigns **high probabilities** to well formed sentences
- plays a crucial role in speech recognition and machine translation systems



# Language Modeling

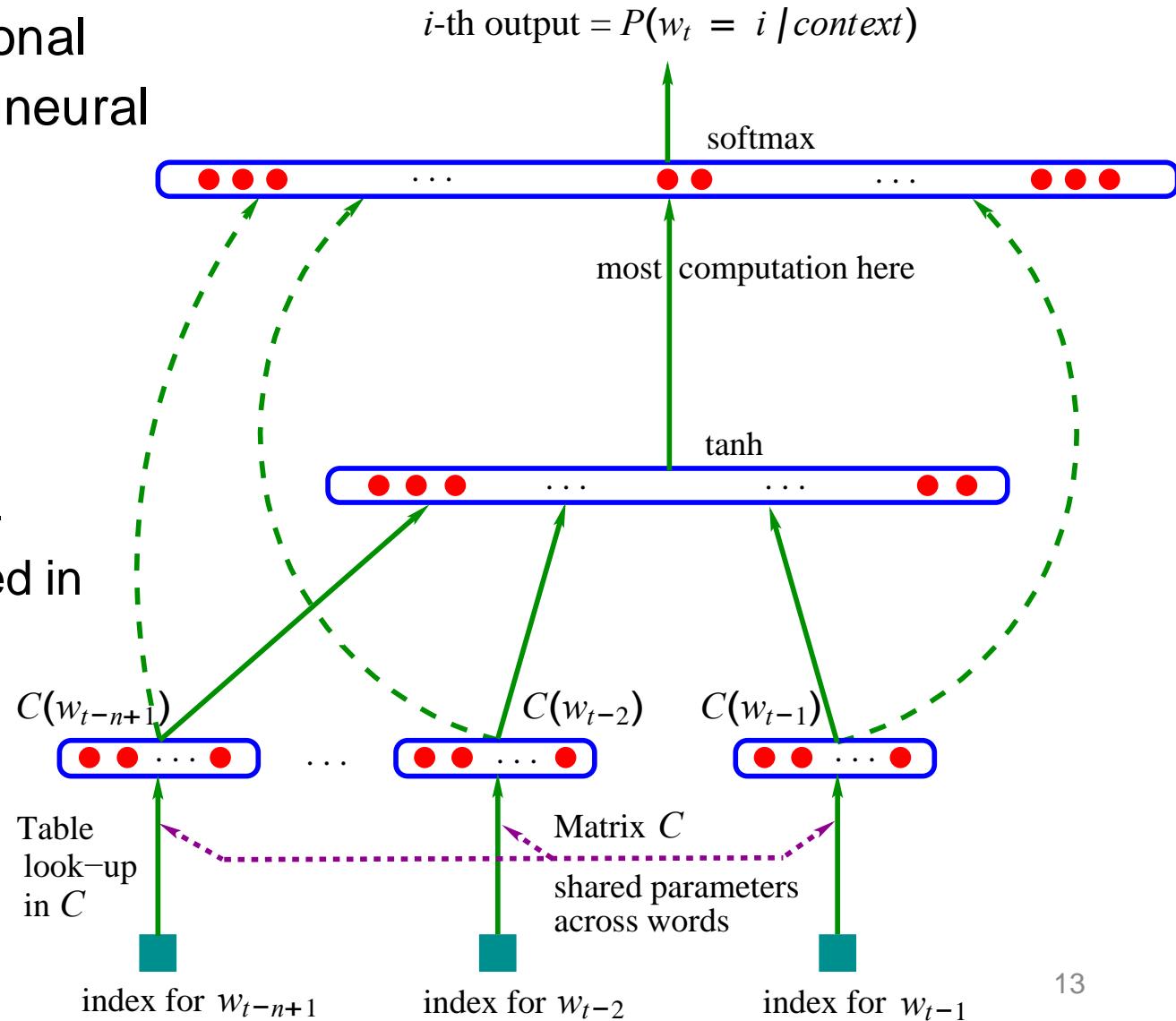
$$p(w_1, \dots, w_T) = \prod_{t=1}^T p(w_t | w_{t-(n-1)}, \dots, w_{t-1})$$

- the  $t^{\text{th}}$  word was generated based only on the  $n-1$  previous words
- we will refer to  $w_{t-(n-1)}, \dots, w_{t-1}$  as the context

# Neural Language Model

- Model the conditional distributions with a neural network:

- learn word representations to allow transfer to n-grams not observed in training corpus



# Neural Language Model

- Can potentially **generalize** to contexts not seen in training set
  - Example:  $P(\text{"eating"} | \text{"the"}, \text{"cat"}, \text{"is"})$
  - Imagine 4-gram [**“the”**, **“cat”**, **“is”**, **“eating”**] is not in training corpus, but [**“the”**, **“dog”**, **“is”**, **“eating”**] is
  - If the word representations of **“cat”** and **“dog”** are similar, then the neural network will be able to generalize to the case of **“cat”**

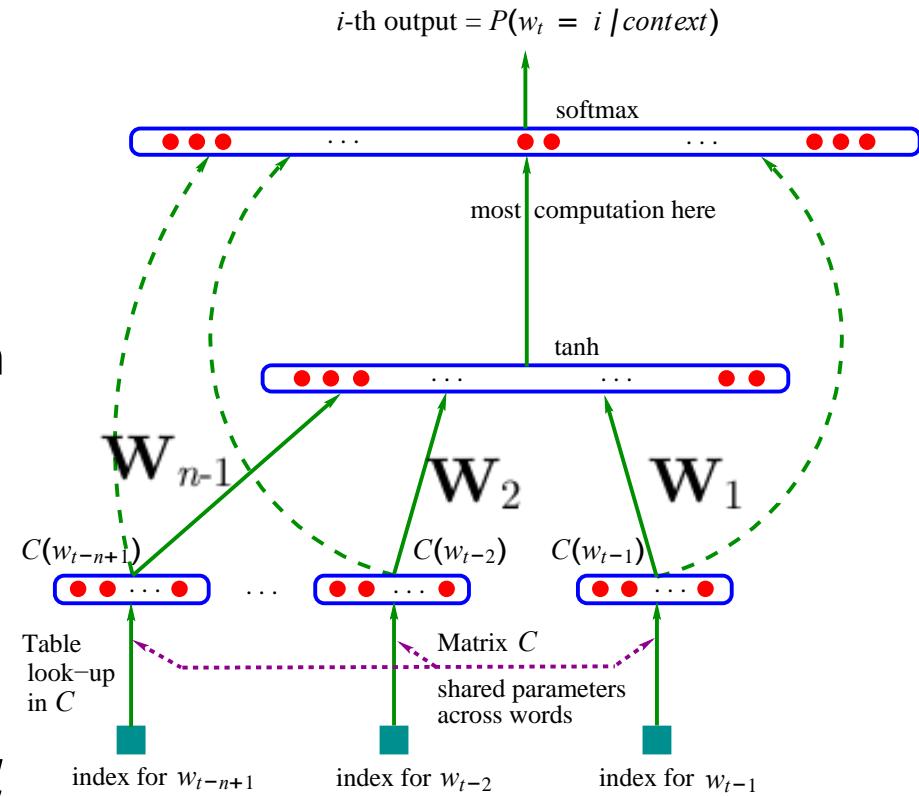
# Neural Language Model

- We know how to propagate gradients in such a network

$$\nabla_{\mathbf{a}(\mathbf{x})} l$$

- let's note the submatrix connecting  $w_{t-i}$  and the hidden layer as  $\mathbf{W}_i$
- The gradient wrt  $C(w)$  for any  $w$  is

$$\nabla_{C(w)} l = \sum_{i=1}^{n-1} \mathbf{1}_{(w_{t-i}=w)} \mathbf{W}_i^\top \nabla_{\mathbf{a}(\mathbf{x})} l$$



# Performance Evaluation

- In language modeling, a common evaluation metric is the **perplexity**
  - it is simply the exponential of the average negative log-likelihood
- Evaluation on Brown Corpus
  - n-gram model (Kneser-Ney smoothing): 321
  - neural network language model: 276
  - neural network + n-gram: 252

# How About Generating Sentences!

Input



Output

A man skiing down the snow covered mountain with a dark sky in the background.

# How About Generating Sentences!

Input



Output

A man skiing down the snow covered mountain with a dark sky in the background.

We want to model:

$$p(w_1, w_2, \dots, w_n) =$$

$$p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)\dots p(w_n|w_1, w_2, \dots, w_{n-1})$$

# Caption Generation with NLM



a car is parked in  
the middle of nowhere .



a wooden table and chairs  
arranged in a room .



a ferry boat on a marina  
with a group of people .



there is a cat sitting on a shelf .



a little boy with a bunch  
of friends on the street .

# Caption Generation with NLM



the two birds are trying  
to be seen in the water .  
**(can't count)**



a giraffe is standing next  
to a fence in a field .  
**(hallucination)**



a parked car while  
driving down the road .  
**(contradiction)**

# Caption Generation with NLM



the two birds are trying  
to be seen in the water .  
**(can't count)**



a giraffe is standing next  
to a fence in a field .  
**(hallucination)**



a parked car while  
driving down the road .  
**(contradiction)**



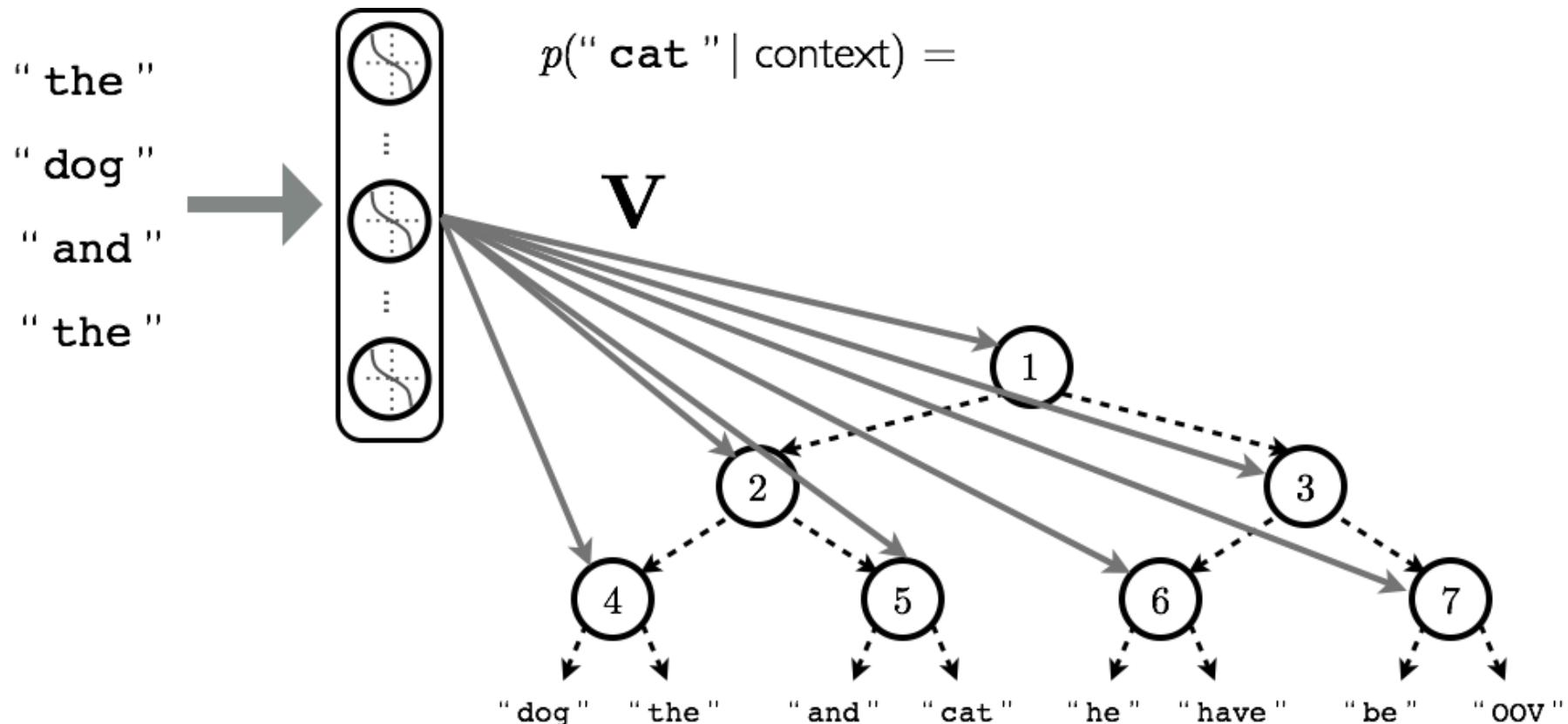
the handlebars are trying  
to ride a bike rack .  
**(nonsensical)**



a woman and a bottle of wine  
in a garden .  
**(gender)**

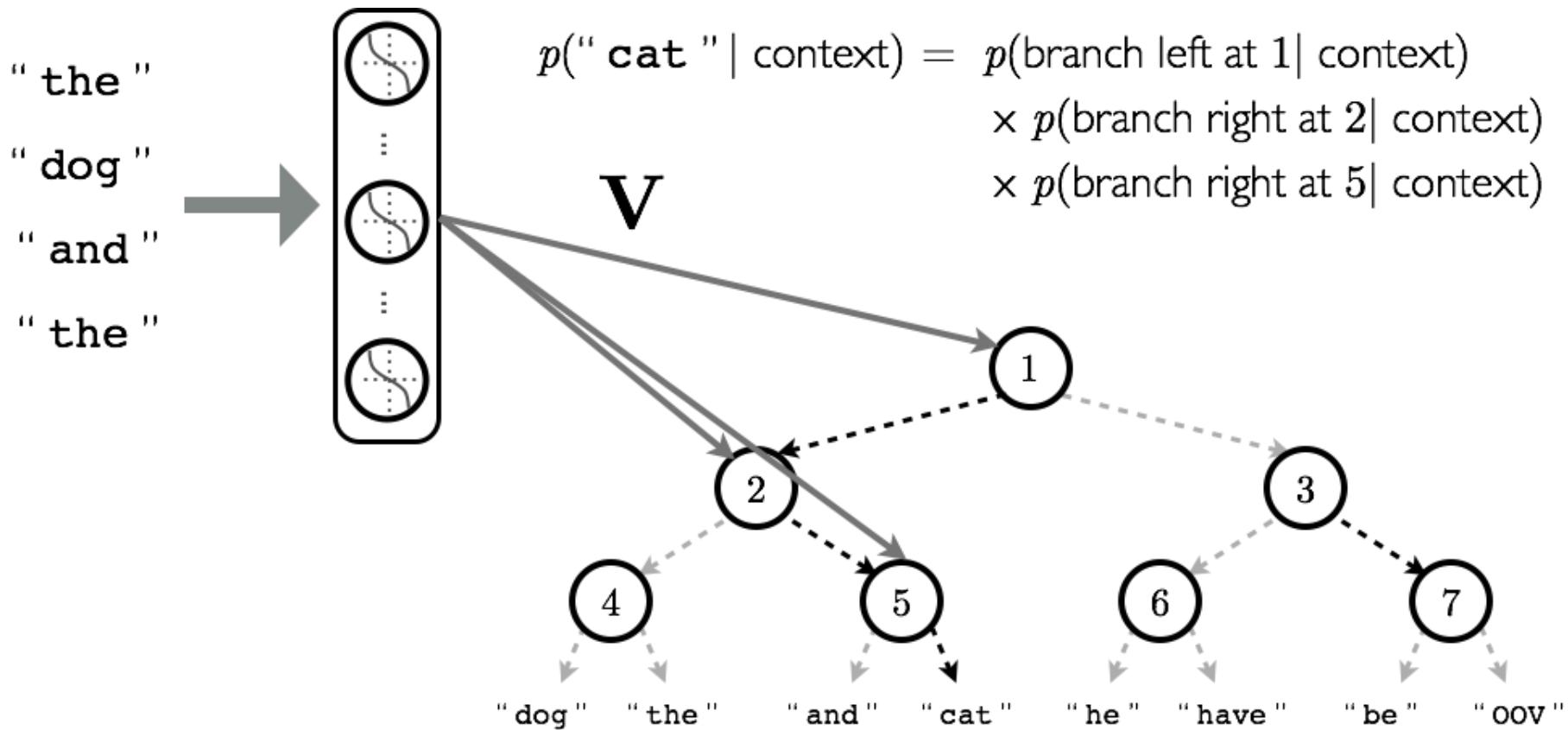
# Hierarchical Output Layer

- Example: [“ the ”, “ dog ”, “ and ”, “ the ”, “ cat ” ]



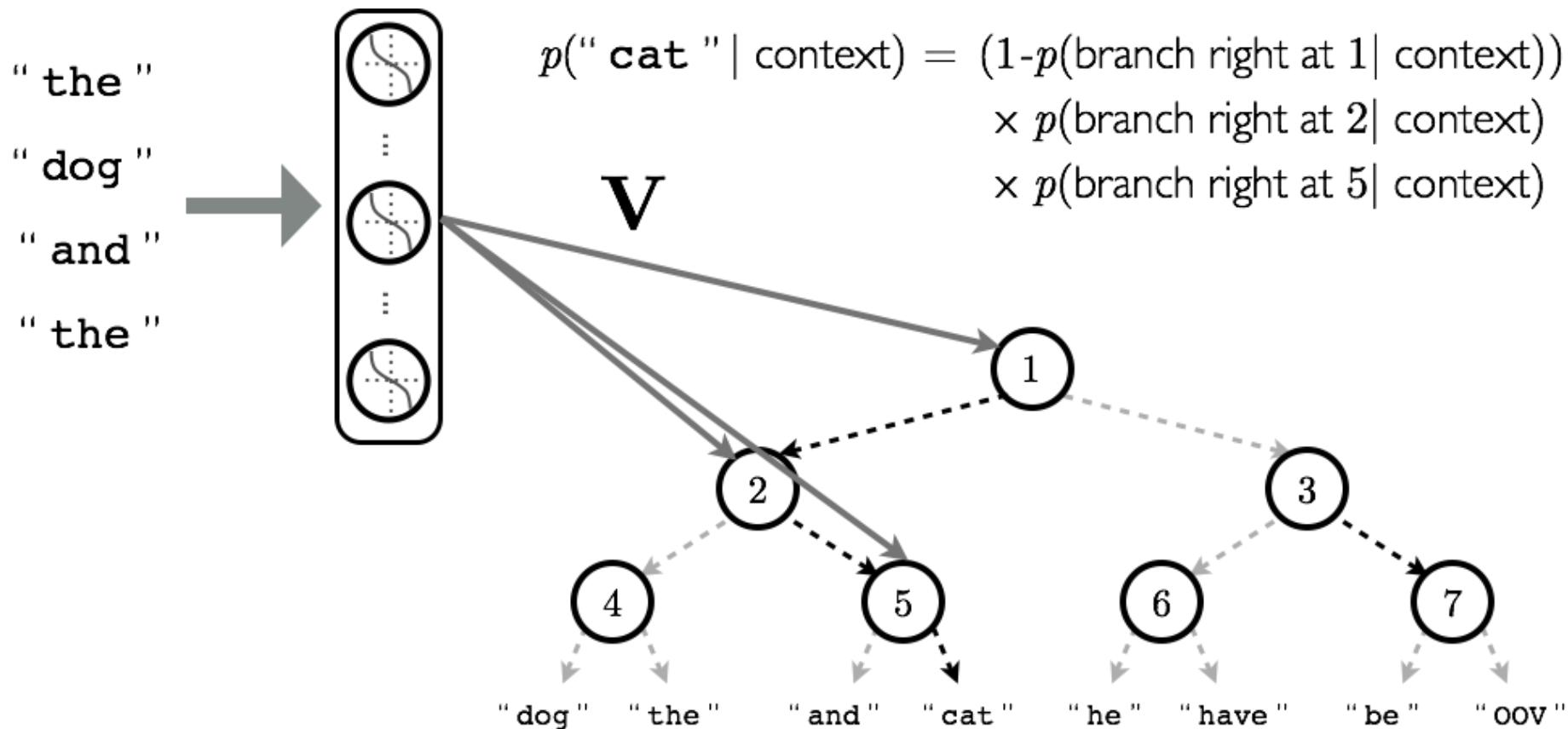
# Hierarchical Output Layer

- Example: ["the", "dog", "and", "the", "cat"]



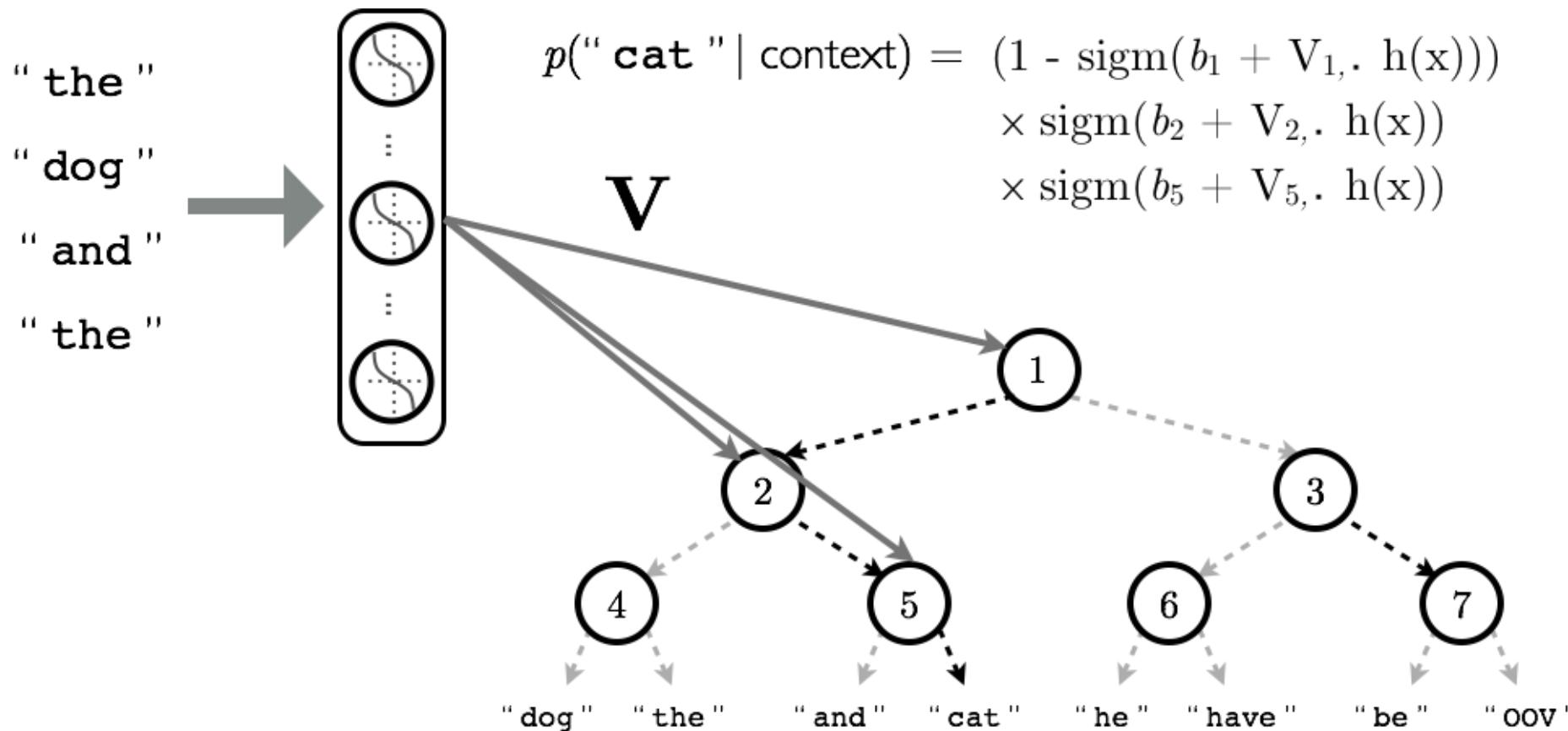
# Hierarchical Output Layer

- Example: ["the", "dog", "and", "the", "cat"]



# Hierarchical Output Layer

- Example: ["the", "dog", "and", "the", "cat"]



# Hierarchical Output Layer

- How to define the word hierarchy?

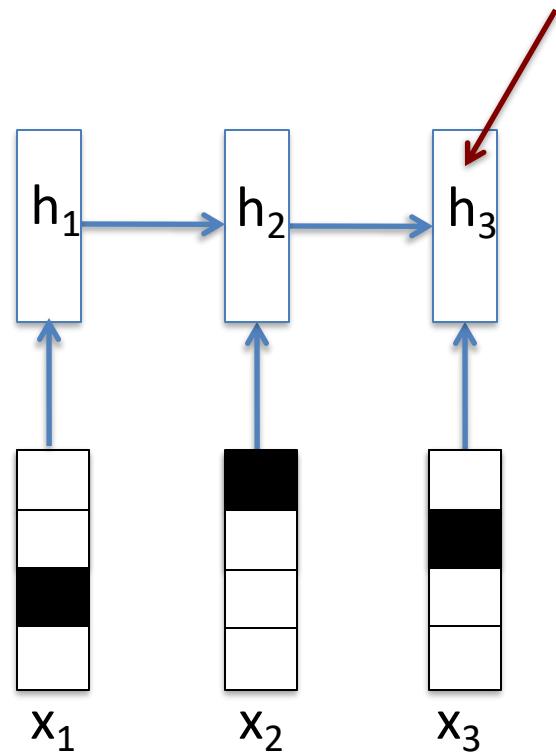
- can use a randomly generated tree
- can use existing linguistic resources, such as WordNet
- can learn the hierarchy using a recursive partitioning strategy

A Scalable Hierarchical Distributed Language Model<sup>[1]</sup>  
Mnih and Hinton, 2008

They report a speedup of 100x, without performance decrease

# Encoding Sentences via Recurrent Neural Network

Sentence  
Representation

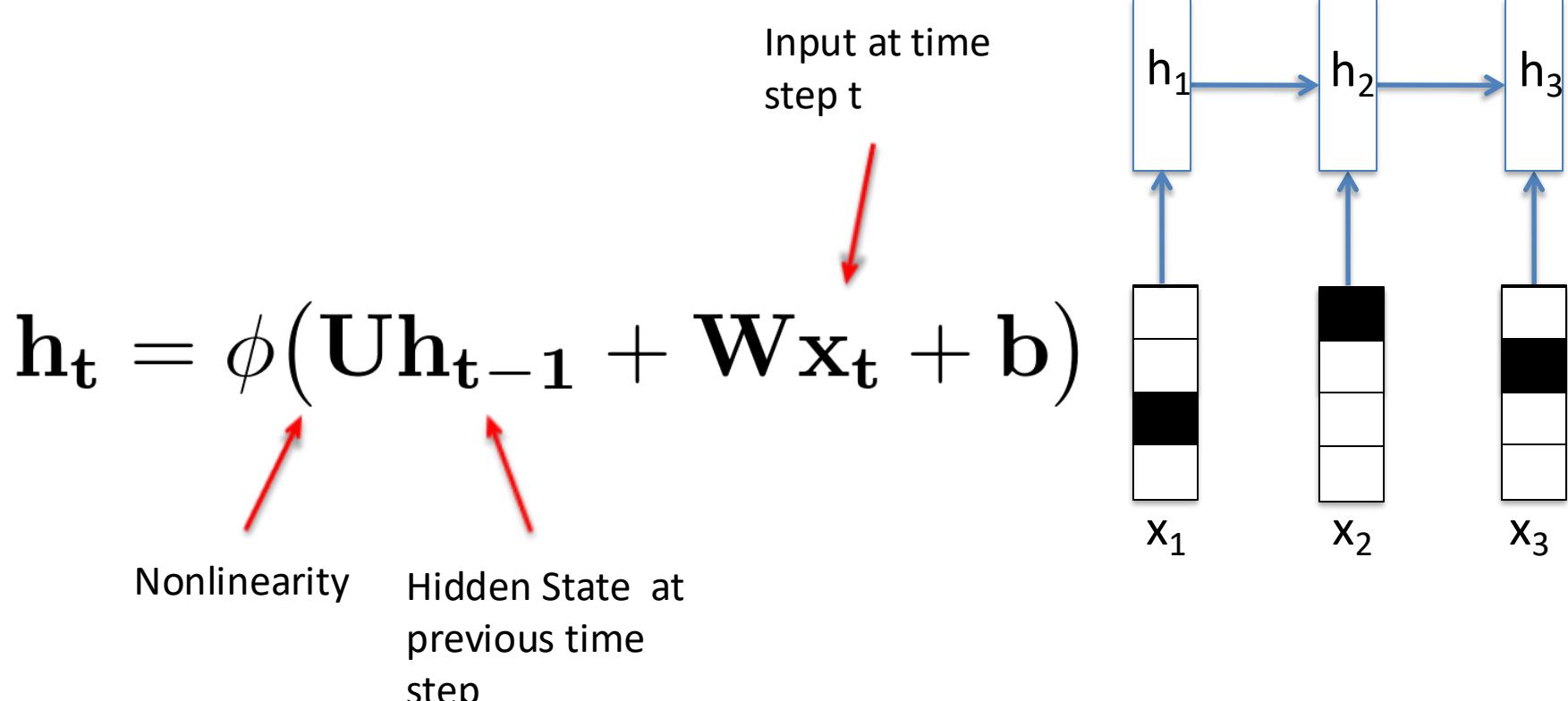


1-of-K encoding of words

Recurrent Neural Network

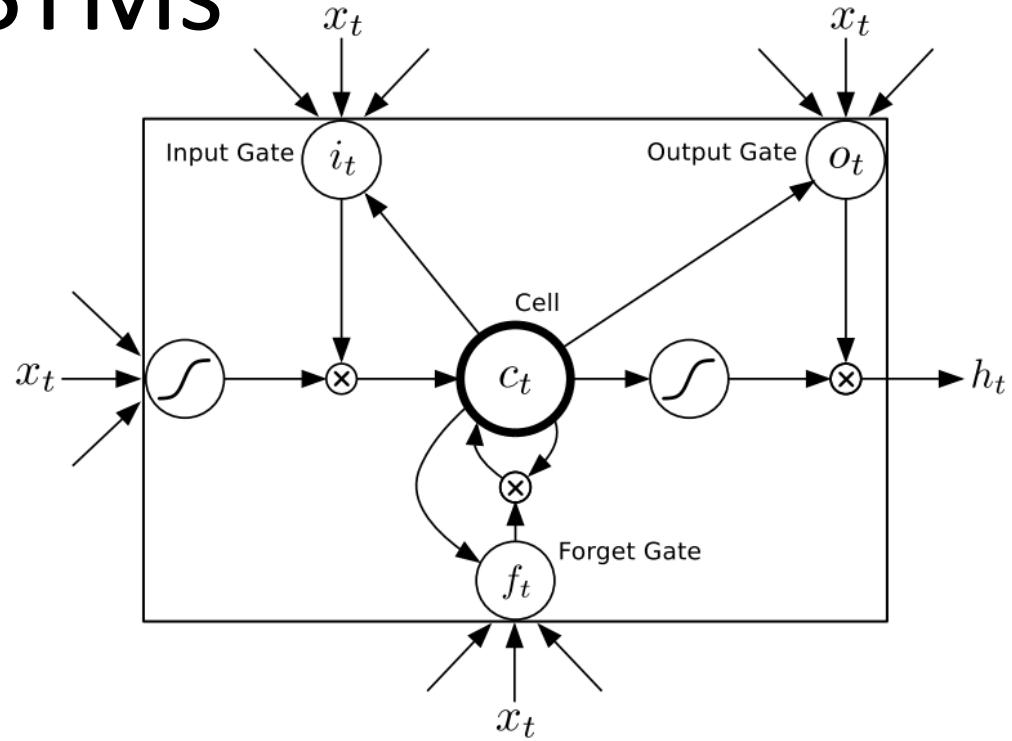
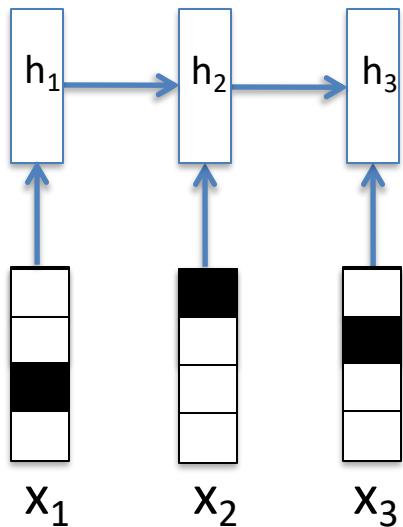
# Recurrent Neural Network

- Replace

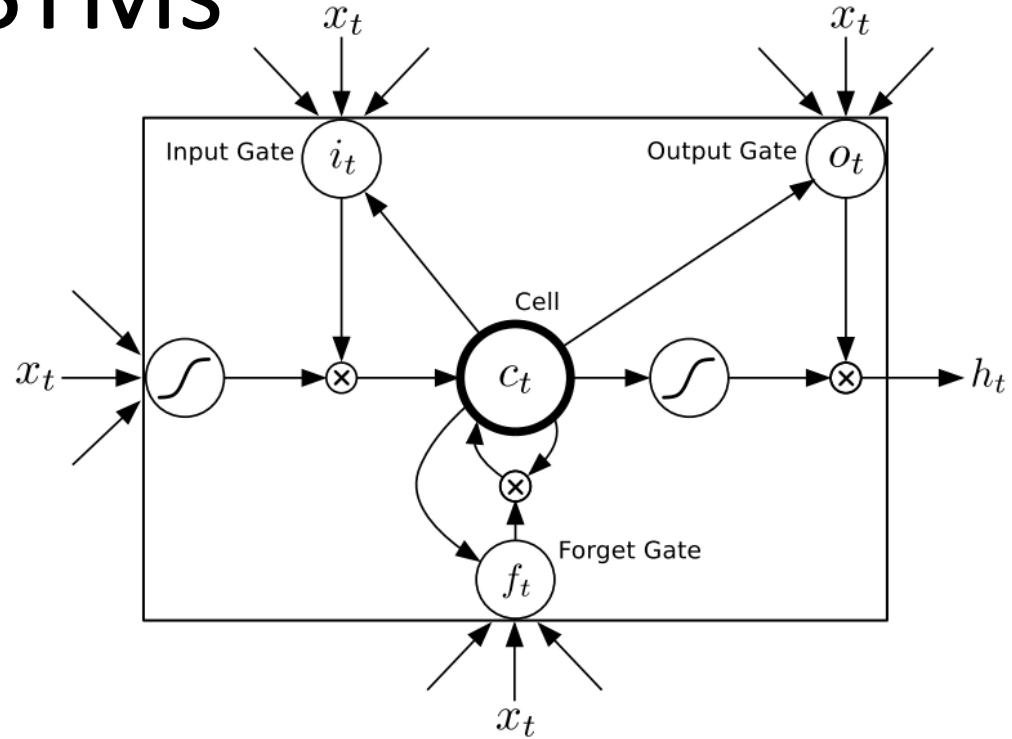
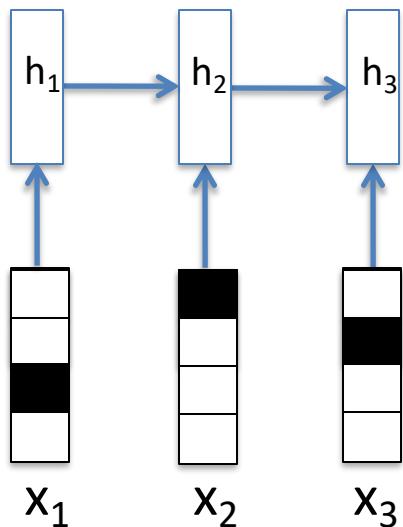


- Can be viewed as a deep neural network with tied weights.

# LSTMs

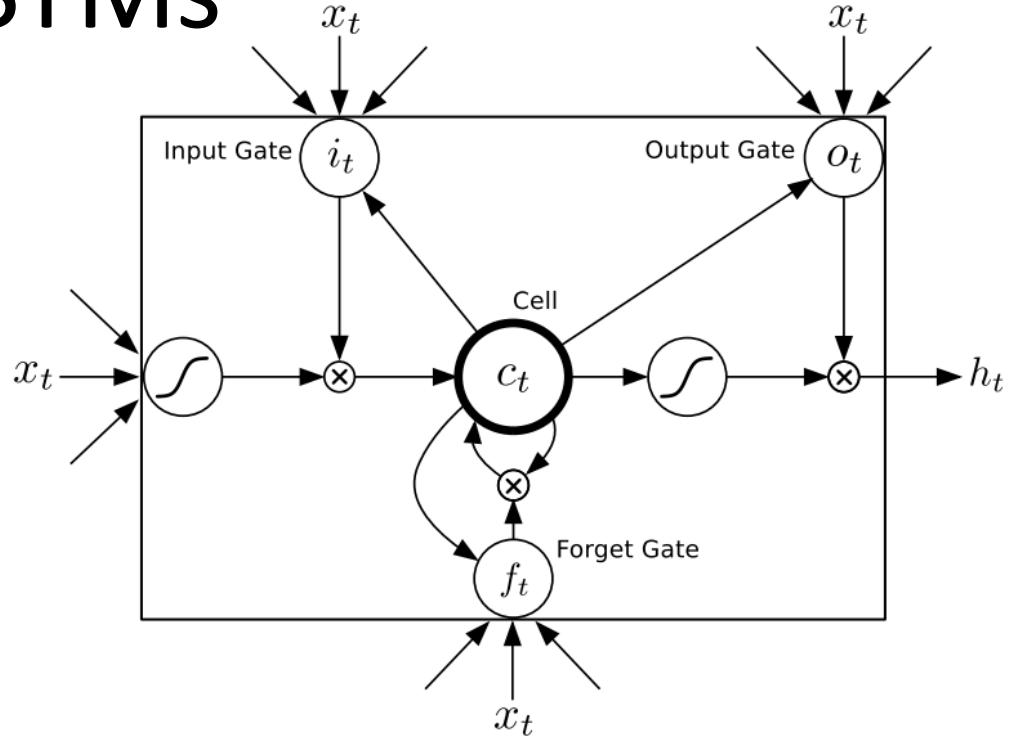
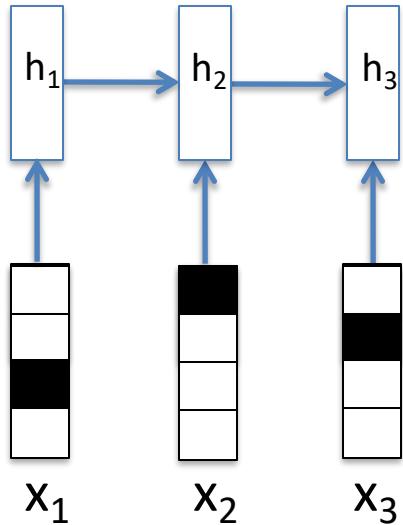


# LSTMs



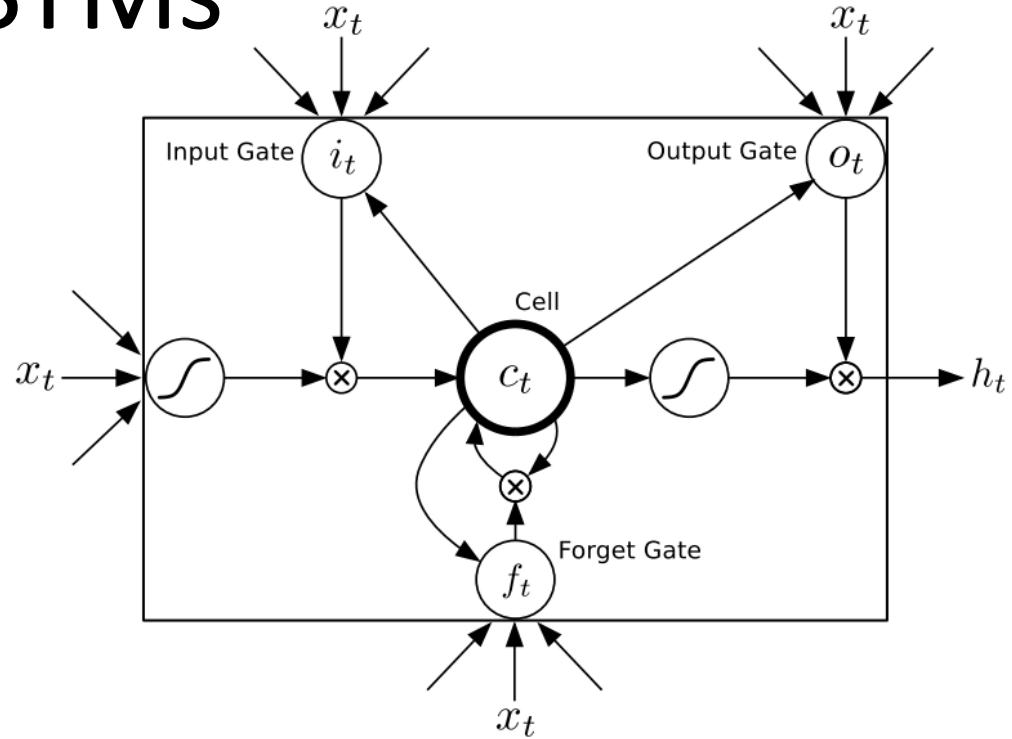
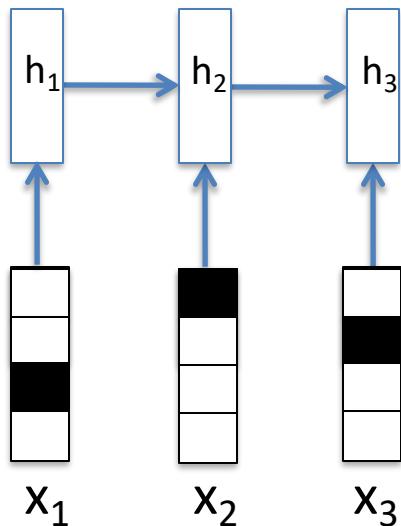
$$i_t = \sigma (W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i),$$

# LSTMs



$$\begin{aligned} i_t &= \sigma (W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \\ f_t &= \sigma (W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \end{aligned}$$

# LSTMs

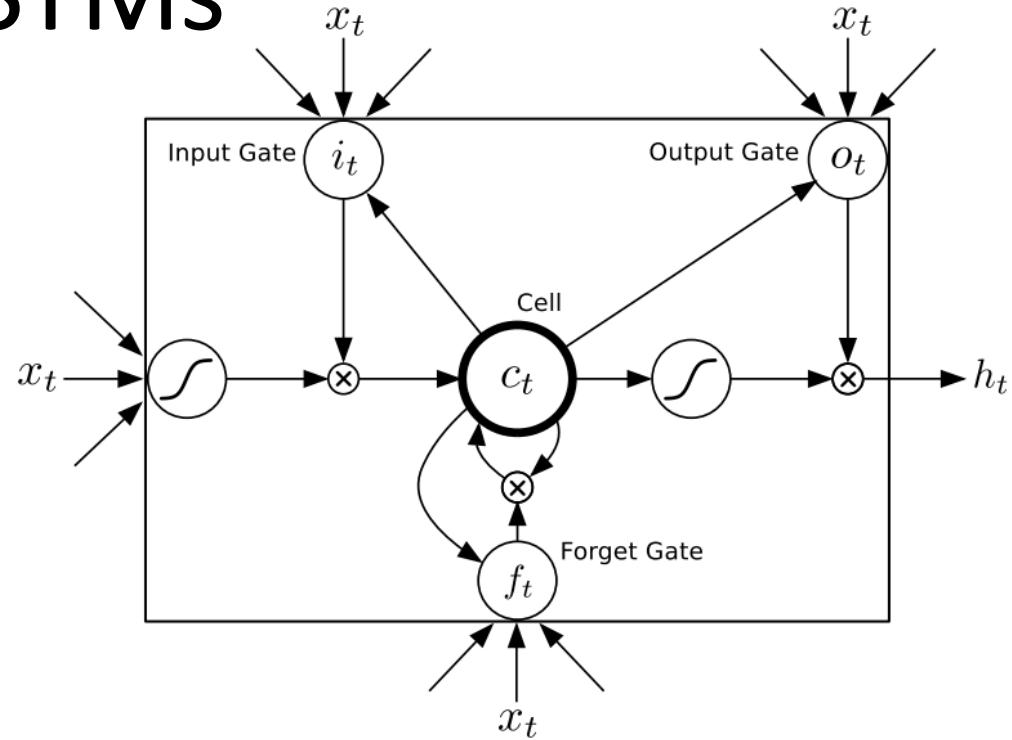
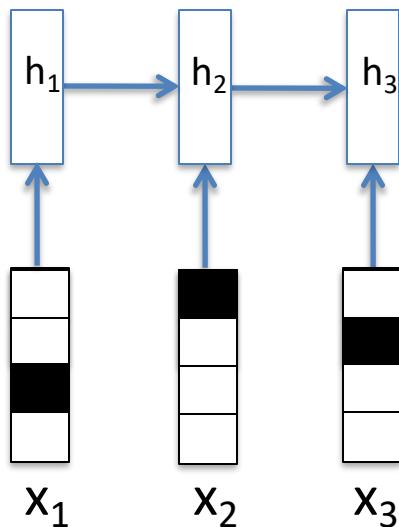


$$\mathbf{i}_t = \sigma (W_{xi} \mathbf{x}_t + W_{hi} \mathbf{h}_{t-1} + W_{ci} \mathbf{c}_{t-1} + \mathbf{b}_i),$$

$$\mathbf{f}_t = \sigma (W_{xf} \mathbf{x}_t + W_{hf} \mathbf{h}_{t-1} + W_{cf} \mathbf{c}_{t-1} + \mathbf{b}_f),$$

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tanh (W_{xc} \mathbf{x}_t + W_{hc} \mathbf{h}_{t-1} + \mathbf{b}_c),$$

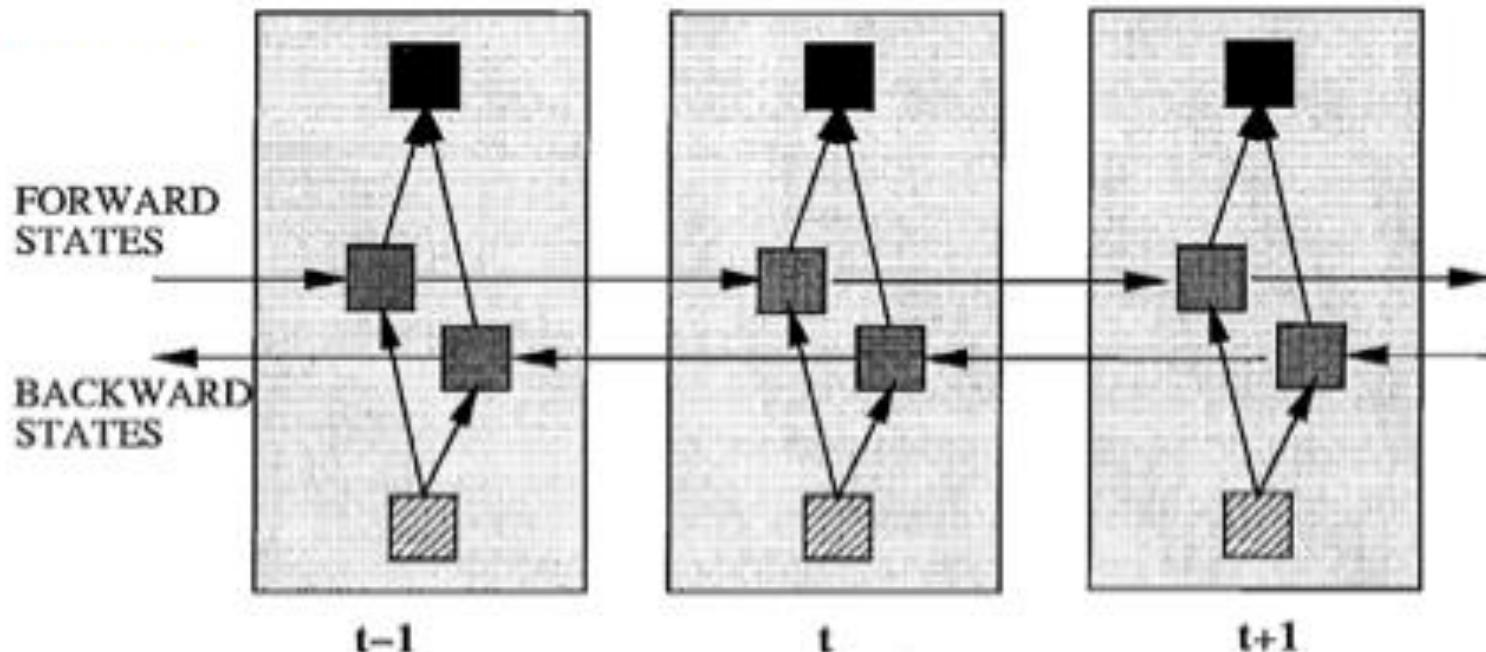
# LSTMs



$$\begin{aligned}
 \mathbf{i}_t &= \sigma(W_{xi}\mathbf{x}_t + W_{hi}\mathbf{h}_{t-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i), \\
 \mathbf{f}_t &= \sigma(W_{xf}\mathbf{x}_t + W_{hf}\mathbf{h}_{t-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f), \\
 \mathbf{c}_t &= \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \tanh(W_{xc}\mathbf{x}_t + W_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c), \\
 \mathbf{o}_t &= \sigma(W_{xo}\mathbf{x}_t + W_{ho}\mathbf{h}_{t-1} + W_{co}\mathbf{c}_t + \mathbf{b}_o), \\
 \mathbf{h}_t &= \mathbf{o}_t \tanh(\mathbf{c}_t).
 \end{aligned}$$

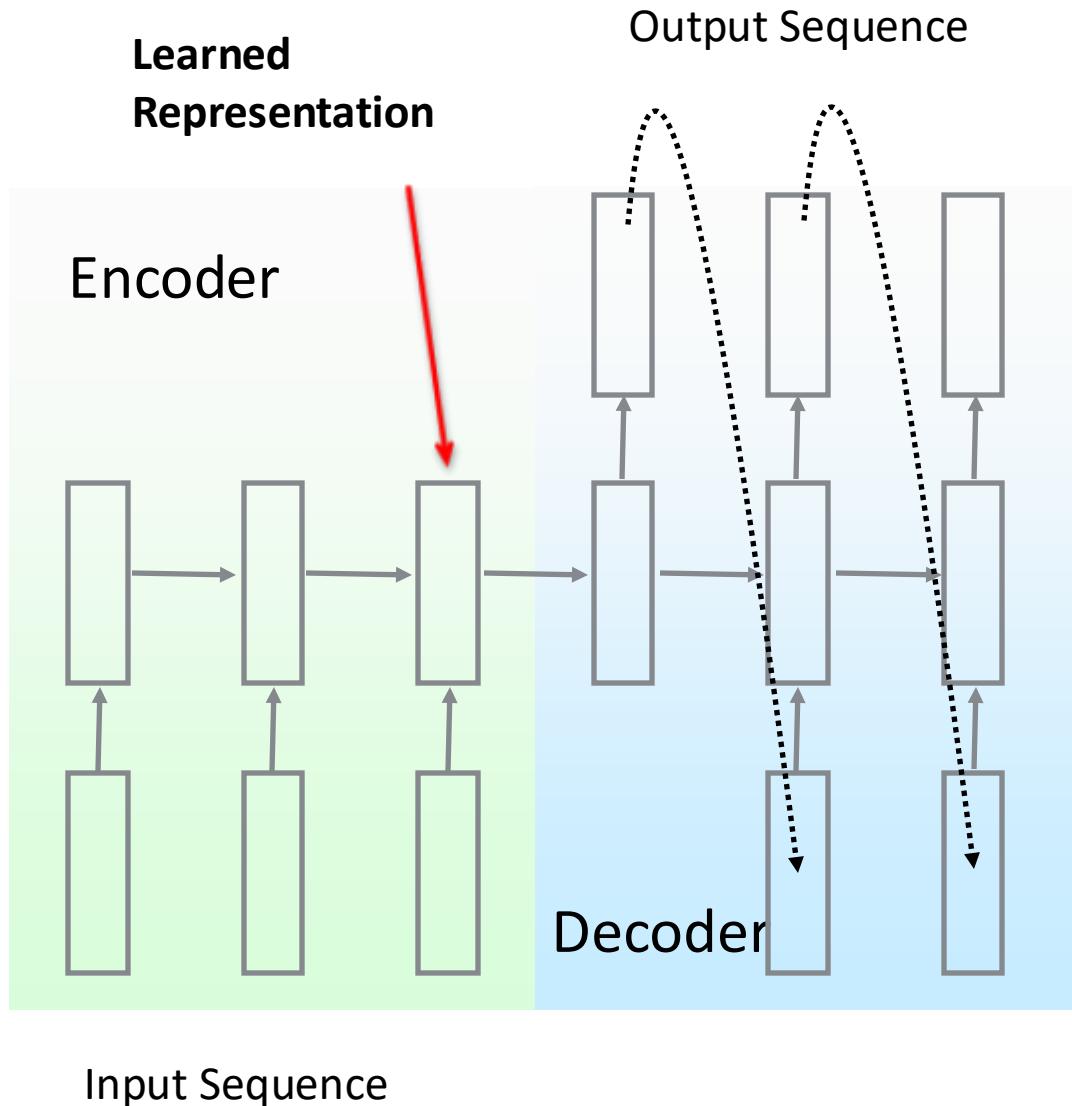
# Bidirectional RNNs

**Bidirectional RNNs (Schuster and Paliwal, 1997)**



- Heavily used in language modeling.

# Sequence to Sequence Learning

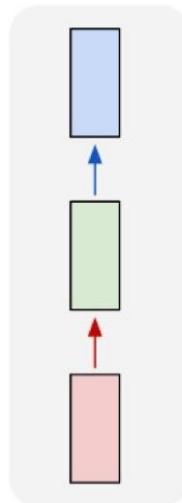


- RNN Encoder-Decoders for Machine Translation (Sutskever et al. 2014; Cho et al. 2014; Kalchbrenner et al. 2013, Srivastava et.al., 2015)

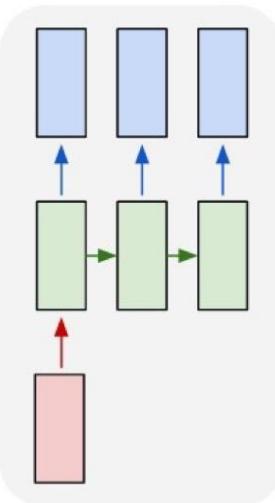
# Sequence to Sequence Models

- Natural language processing is concerned with tasks involving language data

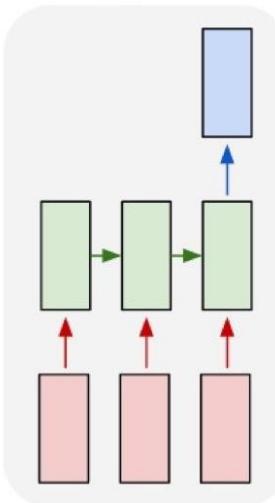
one to one



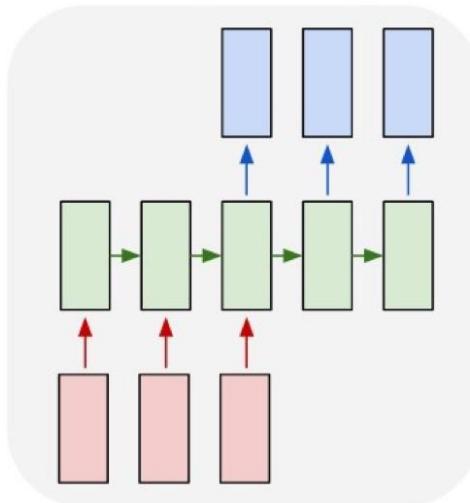
one to many



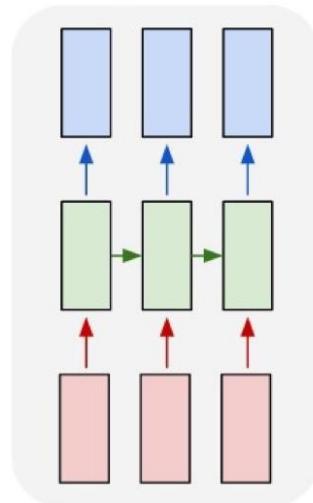
many to one



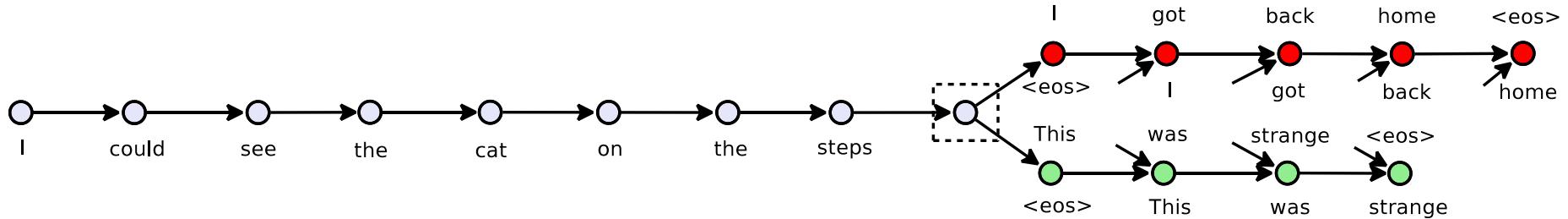
many to many



many to many

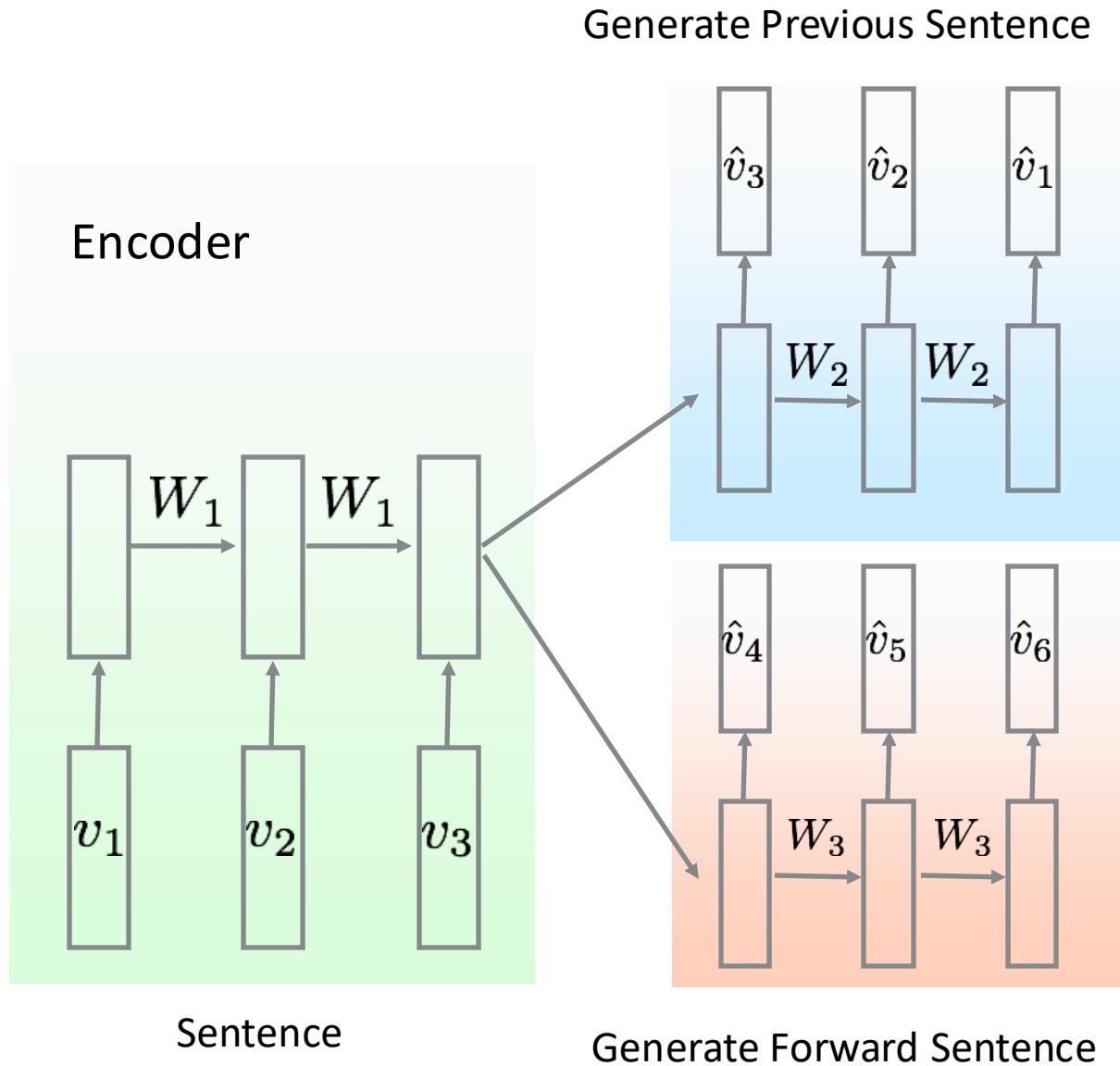


# Skip-Thought Model



- Given a tuple  $(s_{i-1}, s_i, s_{i+1})$  of contiguous sentences:
  - the sentence  $s_i$  is encoded using LSTM.
  - the sentence  $s_i$  attempts to reconstruct the previous sentence and next sentence  $s_{i+1}$ .
- The input is the sentence triplet:
  - I got back home.
  - I could see the cat on the steps.
  - This was strange.

# Skip-Thought Model

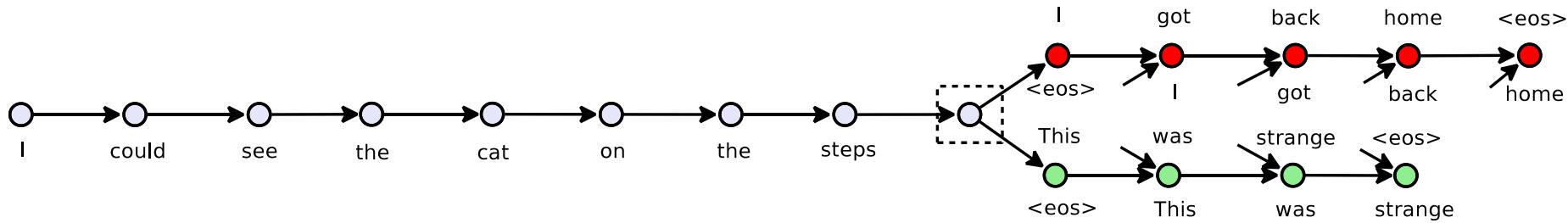


# Learning Objective

- We are given a tuple  $(s_{i-1}, s_i, s_{i+1})$  of contiguous sentences.
- **Objective:** The sum of the log-probabilities for the next and previous sentences conditioned on the encoder representation:

$$\sum_t \underbrace{\log P(w_{i+1}^t | w_{i+1}^{<t}, \mathbf{h}_i)}_{\text{Forward sentence}} + \underbrace{\log P(w_{i-1}^t | w_{i-1}^{<t}, \mathbf{h}_i)}_{\text{Previous sentence}}$$

representation of  
encoder ↓



# Book 11K corpus

# of books	# of sentences	# of words	# of unique words
11,038	74,004,228	984,846,357	1,316,420

- Query sentence along with its nearest neighbor from 500K sentences using cosine similarity:
  - He ran his hand inside his coat, double-checking that the unopened letter was still there.
  - He slipped his hand between his coat and his shirt, where the folded copies lay in a brown envelope.

# Semantic Relatedness

- SemEval 2014 Task 1: semantic relatedness SICK dataset:  
Given two sentences, produce a score of how semantically related these sentences are based on human generated scores (1 to 5).
- The dataset comes with a predefined split of 4500 training pairs, 500 development pairs and 4927 testing pairs.
- Using skip-thought vectors for each sentence, we simply train a linear regression to predict semantic relatedness.
  - For pair of sentences, we compute component-wise features between pairs (e.g.  $|u-v|$ ).

# Semantic Relatedness

	Method	$r$	$\rho$	MSE
SemEval 2014 sub- missions	Illinois-LH [18]	0.7993	0.7538	0.3692
	UNAL-NLP [19]	0.8070	0.7489	0.3550
	Meaning Factory [20]	0.8268	0.7721	0.3224
	ECNU [21]	0.8414	—	—
Results reported by Tai et.al.	Mean vectors [22]	0.7577	0.6738	0.4557
	DT-RNN [23]	0.7923	0.7319	0.3822
	SDT-RNN [23]	0.7900	0.7304	0.3848
	LSTM [22]	0.8528	0.7911	0.2831
	Bidirectional LSTM [22]	0.8567	0.7966	0.2736
	Dependency Tree-LSTM [22]	<b>0.8676</b>	<b>0.8083</b>	<b>0.2532</b>
Ours	uni-skip	0.8477	0.7780	0.2872
	bi-skip	0.8405	0.7696	0.2995
	combine-skip	0.8584	0.7916	0.2687
	combine-skip+COCO	0.8655	0.7995	0.2561

- Our models outperform all previous systems from the SemEval 2014 competition. This is remarkable, given the simplicity of our approach and the lack of feature engineering.

# Semantic Relatedness

Sentence 1	Sentence 2	GT	pred
A little girl is looking at a woman in costume	A young girl is looking at a woman in costume	4.7	4.5
A little girl is looking at a woman in costume	The little girl is looking at a man in costume	3.8	4.0
A little girl is looking at a woman in costume	A little girl in costume looks like a woman	2.9	3.5
A sea turtle is hunting for fish	A sea turtle is hunting for food	4.5	4.5
A sea turtle is not hunting for fish	A sea turtle is hunting for fish	3.4	3.8
A man is driving a car	The car is being driven by a man	5	4.9
There is no man driving the car	A man is driving a car	3.6	3.5
A large duck is flying over a rocky stream	A duck, which is large, is flying over a rocky stream	4.8	4.9
A large duck is flying over a rocky stream	A large stream is full of rocks, ducks and flies	2.7	3.1
A person is performing acrobatics on a motorcycle	A person is performing tricks on a motorcycle	4.3	4.4
A person is performing tricks on a motorcycle	The performer is tricking a person on a motorcycle	2.6	4.4
Someone is pouring ingredients into a pot	Someone is adding ingredients to a pot	4.4	4.0
Nobody is pouring ingredients into a pot	Someone is pouring ingredients into a pot	3.5	4.2
Someone is pouring ingredients into a pot	A man is removing vegetables from a pot	2.4	3.6

- Example predictions from the SICK test set. GT is the ground truth relatedness, scored between 1 and 5.
- The last few results: slight changes in sentences result in large changes in relatedness that we are unable to score correctly.

# Paraphrase Detection

- Microsoft Research Paraphrase Corpus: For two sentences one must predict whether or not they are paraphrases.

	Method	Acc	F1	
Recursive Auto- encoders	feats [24]	73.2		The training set contains 4076 sentence pairs (2753 are positive)
	RAE+DP [24]	72.6		
	RAE+feats [24]	74.2		
	RAE+DP+feats [24]	76.8	83.6	
Best published results	FHS [25]	75.0	82.7	The test set contains 1725 pairs (1147 are positive).
	PE [26]	76.1	82.7	
	WDDP [27]	75.6	83.0	
	MTMETRICS [28]	<b>77.4</b>	<b>84.1</b>	
Ours	uni-skip	73.0	81.9	
	bi-skip	71.2	81.2	
	combine-skip	73.0	82.0	
	combine-skip + feats	75.8	83.0	

# Classification Benchmarks

- 5 datasets: movie review sentiment (MR), customer product reviews (CR), subjectivity/objectivity classification (SUBJ), opinion polarity (MPQA) and question-type classification (TREC).

	<b>Method</b>	<b>MR</b>	<b>CR</b>	<b>SUBJ</b>	<b>MPQA</b>	<b>TREC</b>
Bag-of-words	NB-SVM [41]	79.4	<u>81.8</u>	93.2	86.3	
	MNB [41]	79.0	<u>80.0</u>	<u>93.6</u>	86.3	
	cBoW [6]	77.2	79.9	<u>91.3</u>	86.4	87.3
Supervised	GrConv [6]	76.3	81.3	89.5	84.5	88.4
	RNN [6]	77.2	82.3	93.7	90.1	90.2
	BRNN [6]	82.3	82.6	94.2	90.3	91.0
	CNN [4]	81.5	85.0	93.4	89.6	<b>93.6</b>
	AdaSent [6]	<b>83.1</b>	<b>86.3</b>	<b>95.5</b>	<b>93.3</b>	92.4
Paragraph-vector [7]		74.8	78.1	90.5	74.2	91.8
Ours	uni-skip	75.5	79.3	92.1	86.9	91.4
	bi-skip	73.9	77.9	92.5	83.3	89.4
	combine-skip	76.5	80.1	<u>93.6</u>	87.1	<u>92.2</u>
	combine-skip + NB	<u>80.4</u>	81.3	<u>93.6</u>	<u>87.5</u>	