CIS 6930 Topics in Computing for Data Science
Week 6a: Recurrent Neural Networks (2)
Encoder-Decoder Models

9/30/2021 & 10/5/2021

Yoshihiko (Yoshi) Suhara

2pm-3:20pm

This Week & Next Week

- Thu 10/7 Midterm exam (written exam **on campus**)
  - Time: 2pm-3:20pm
  - Location: LBR 252

- Fall Break

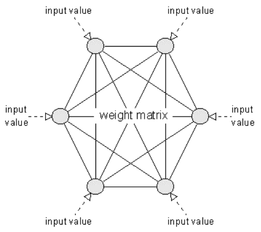
Week 6!

- ~~Week 1: Deep Learning Basics (Thu 9/9)~~
- ~~Week 2: AutoEncoder (Tue 9/14)~~
- ~~Week 3: Convolutional Neural Networks (Thu 9/16)~~
- ~~Week 4: GAN (Tue 9/21)~~
- ~~Week 5: Word embeddings: Word2vec, GloVe (Thu 9/23)~~
- **Week 6: Recurrent Neural Networks (Tue 9/28, Thu 9/30)**
- Week 7: Review/Project pitch & Mid-term (Tue 10/5, Thu 10/7)
- Fall Break
- ...

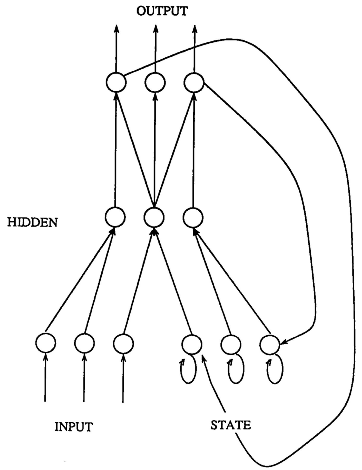
Recap: Recurrent Neural Networks (1)

Recurrent Neural Networks for Sequential Input

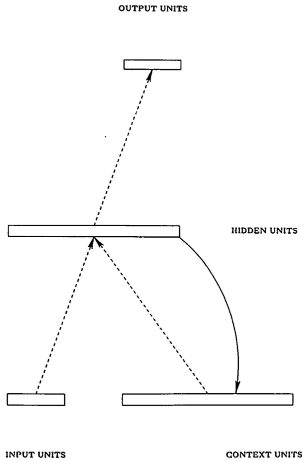
- Inspired by Hopfield Network (1982)
- "Simple" Recurrent Neural Networks
  - Jordan Net (1986)
  - **Elman Net (1990)**



Hopfield Network

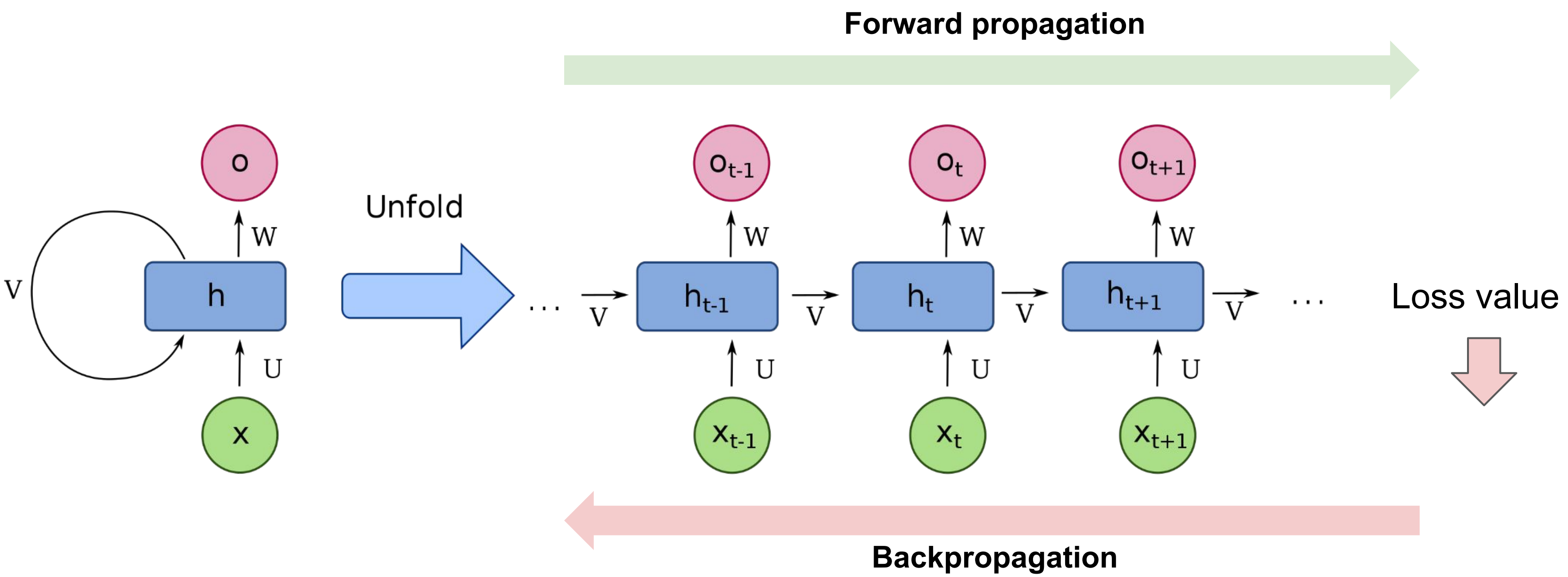


Jordan Net



Elman Net

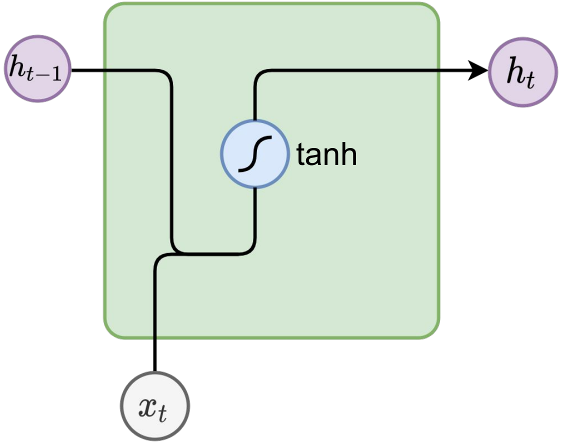
Unfolding RNN
Backpropagation Through Time (BPTT)



RNN Cell for Sequential Data

RNN = Elman Net

- Input value + previous hidden state → Next hidden state



$$h_t = \tanh(\underline{W_{ih}} x_t + b_{ih} + \underline{W_{hh}} h_{(t-1)} + b_{hh})$$

$$W_h\left[x_t ; h_{(t-1)}\right] + b_h$$

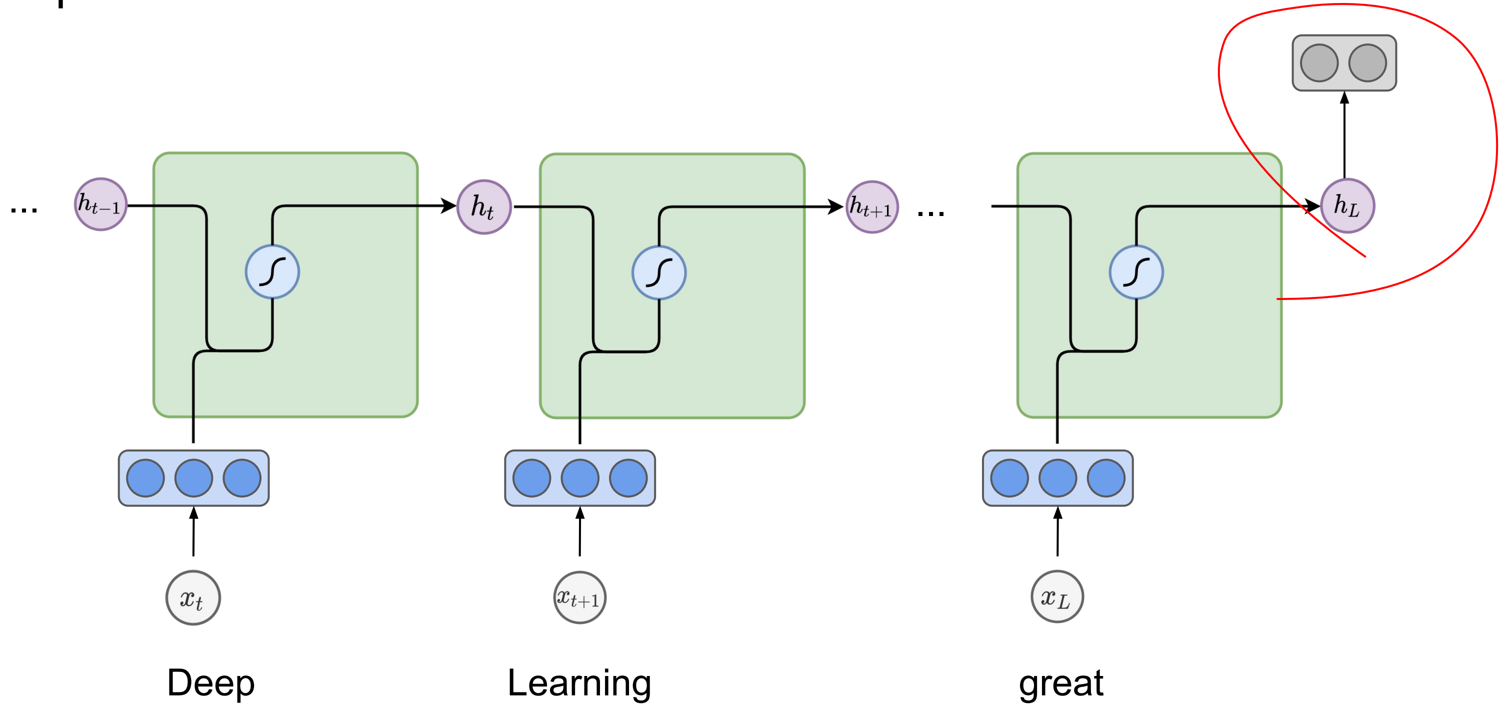
Another notation

The figures are based on the following blog article (highly recommended!)
[Illustrated Guide to LSTM's and GRU's: A step by step explanation]

Text Classification Example

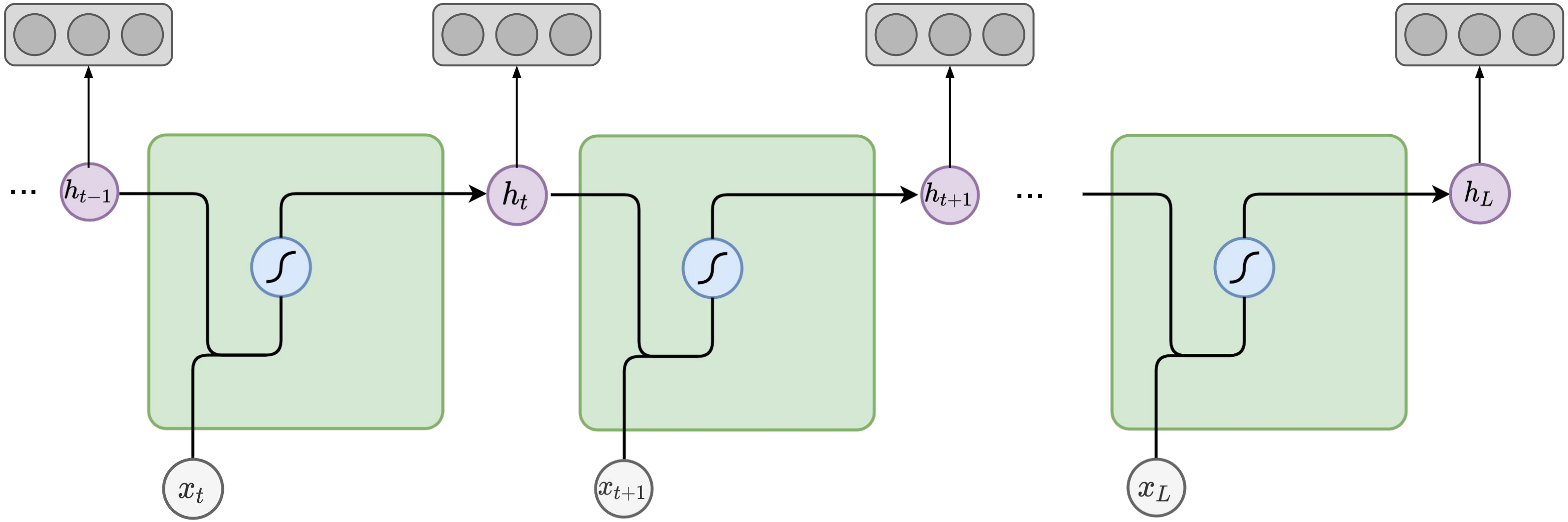
- Input: “Deep Learning is great”
- Label: positive

Any other options?



Where to Attach Output Layer?

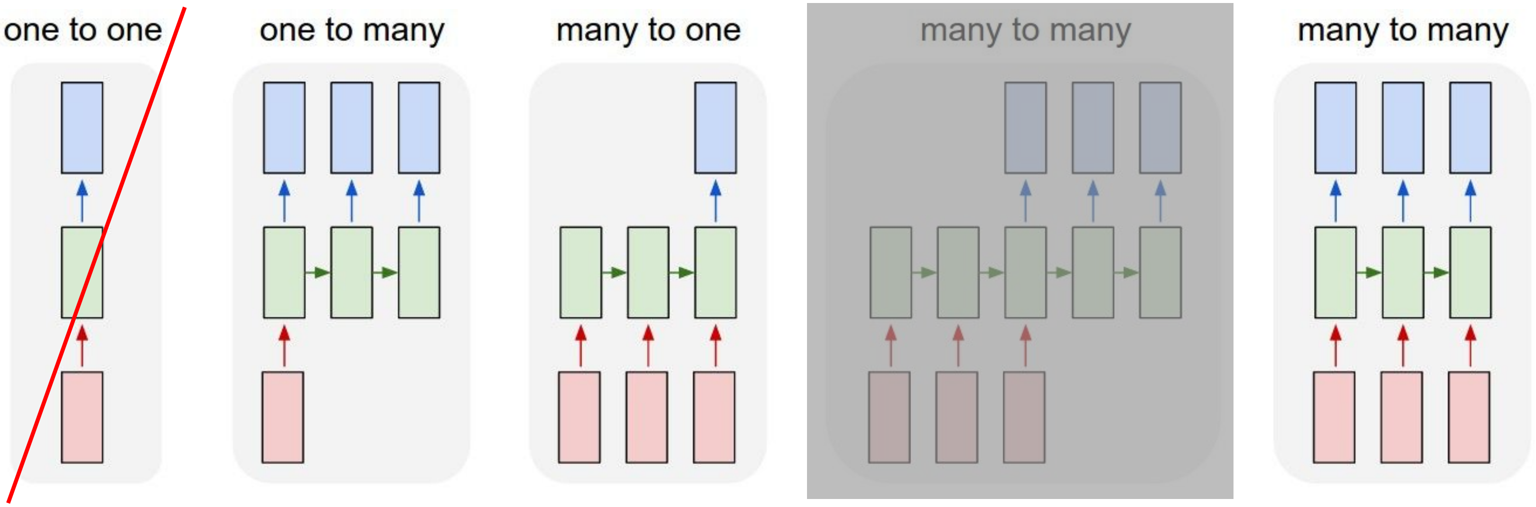
- Where?



Any other options?

→ RNN for **sequential tagging**

RNN Application Patterns



Single-input Generation?

Classification

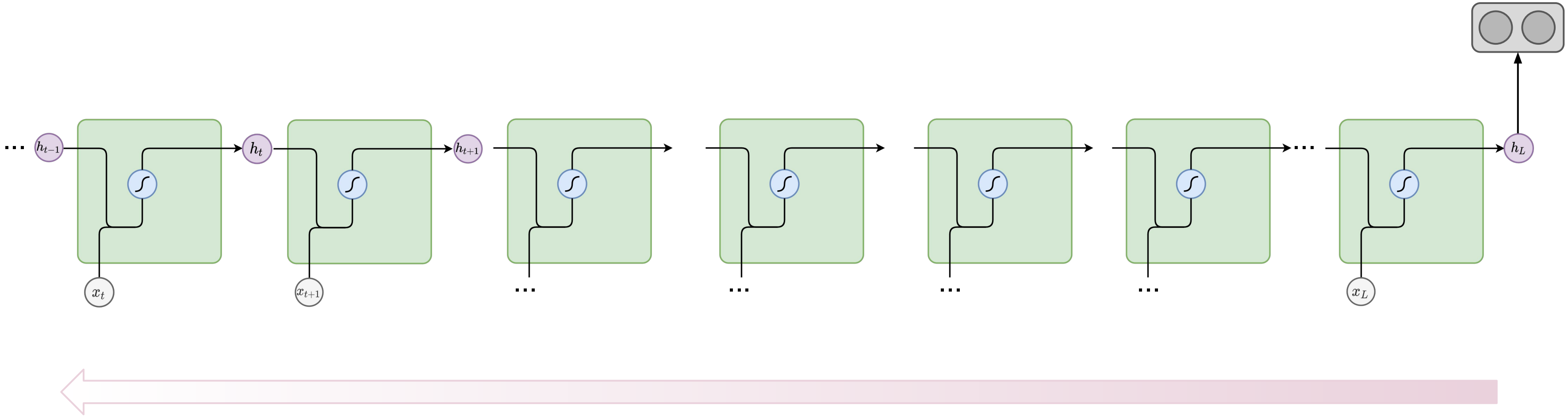
Encoder-decoder model
(aka seq2seq)

Sequential tagging

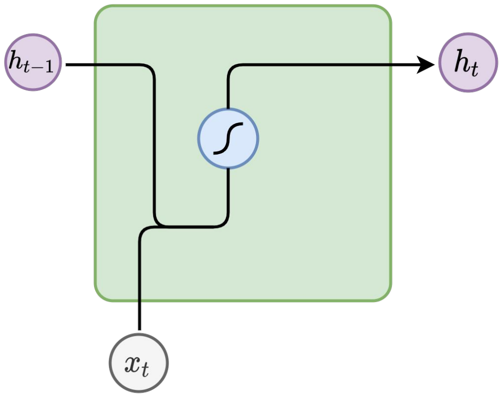
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

What’s the issue with RNNs?

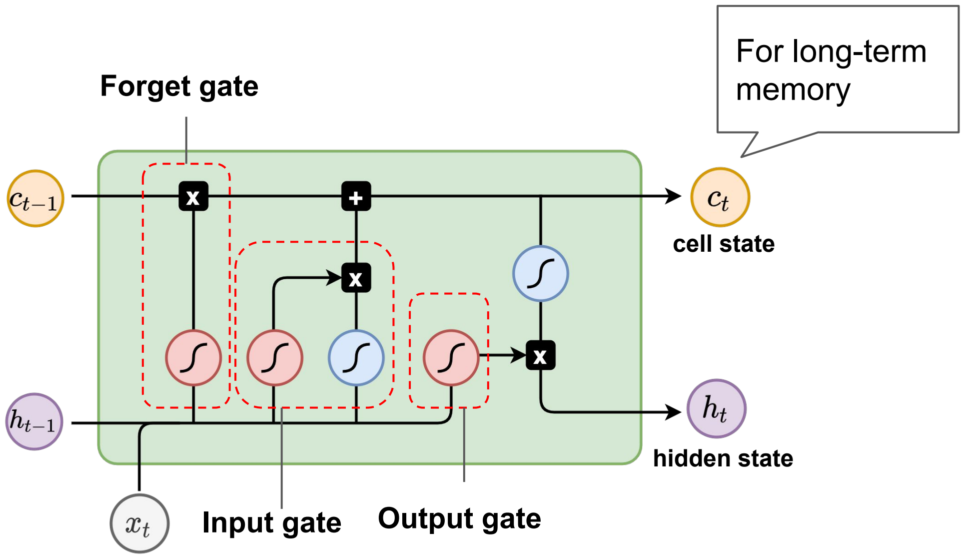
- Not a very good design for **a long sequence**
  - **Later input values** have higher impact on the last hidden state
  - Gradient vanishing problem



LSTM: Cell State + Gating Mechanism



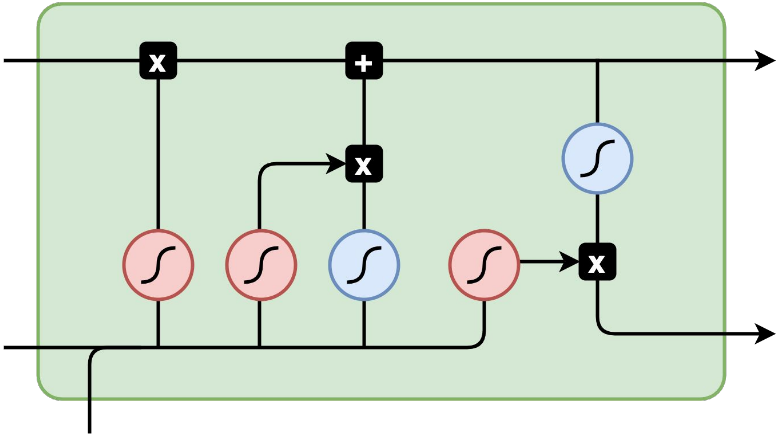
RNN Cell



LSTM Cell

Recap

Long-short Term Memory (LSTM) Cell
[Hochreiter and Schmidhuber 1997]



$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Agenda (1st Half)

- Advanced topics (contd.)
  - GRU
  - Stacked GRU/LSTM
  - Bidirectional GRU/LSTM
- Hands-on Session
- Assignment 3

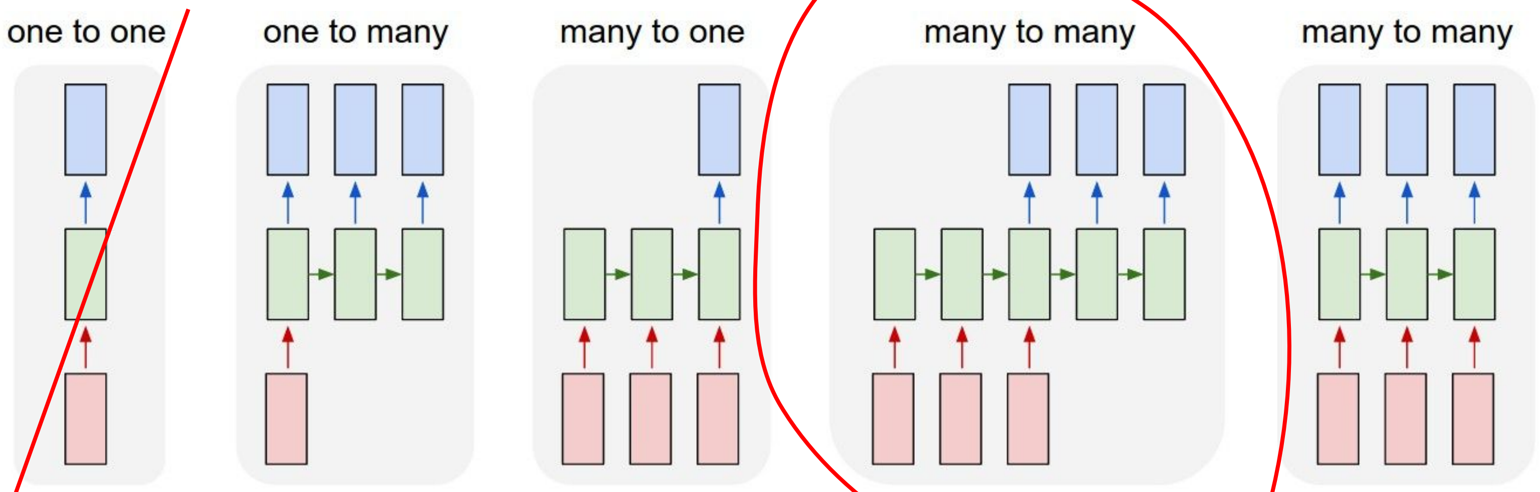
Link to the previous slide deck

- https://github.com/suhara/cis6930-fall2021/blob/main/slides/cis6930-week6a-recurrent-neural-networks-1.pdf

Agenda (2nd Half)

- Encoder-Decoder Models (aka seq2seq Models)
- Decoding Algorithms
- Attention mechanism

RNN Application Patterns



Single-input Generation?

Classification

Encoder-decoder model (aka seq2seq)

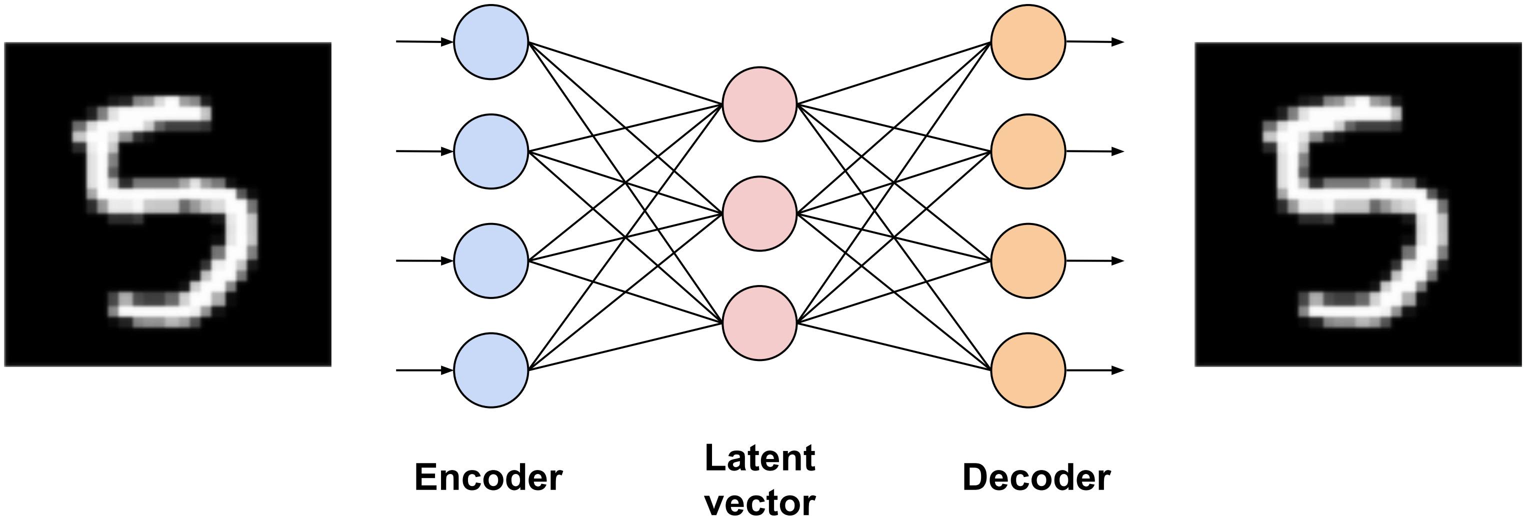
Sequential tagging

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Autoencoders

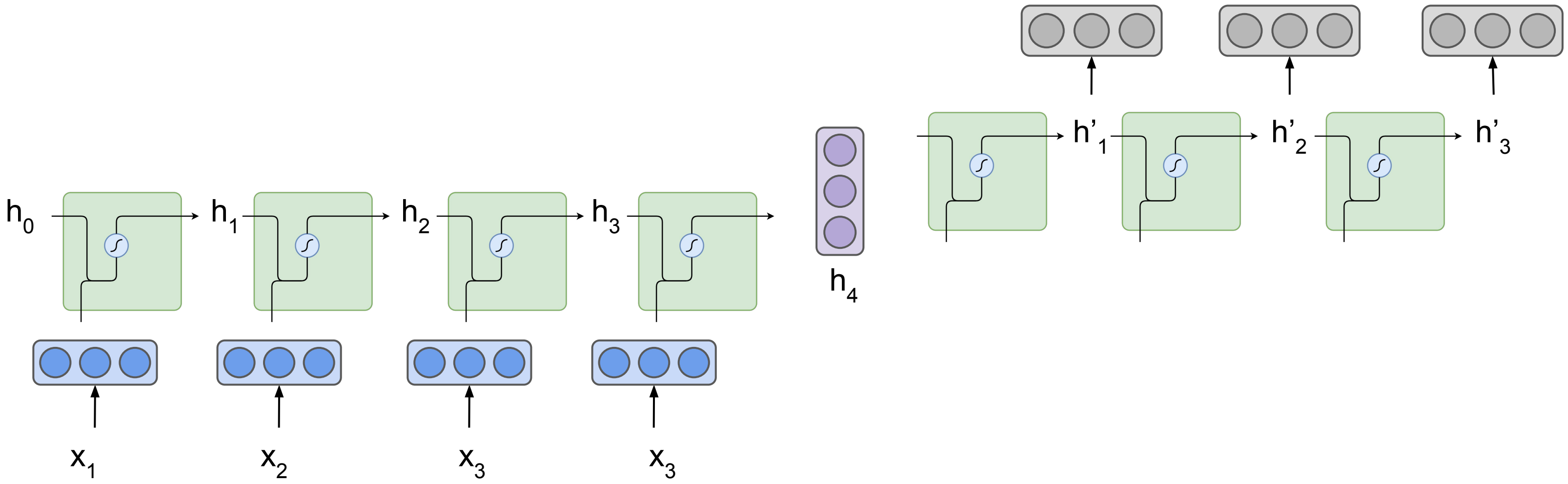
Recap

- **Encoder-decoder models** that **learn to reconstruct the original data**



Encoder-Decoder Model
aka Sequence-to-sequence (seq2seq) Model

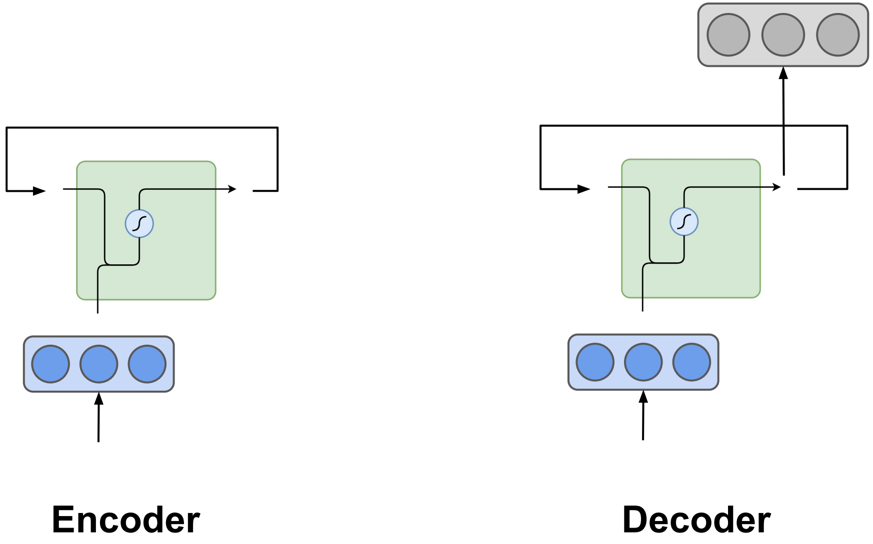
- Input: Variable length sequence
- Output: Variable length sequence



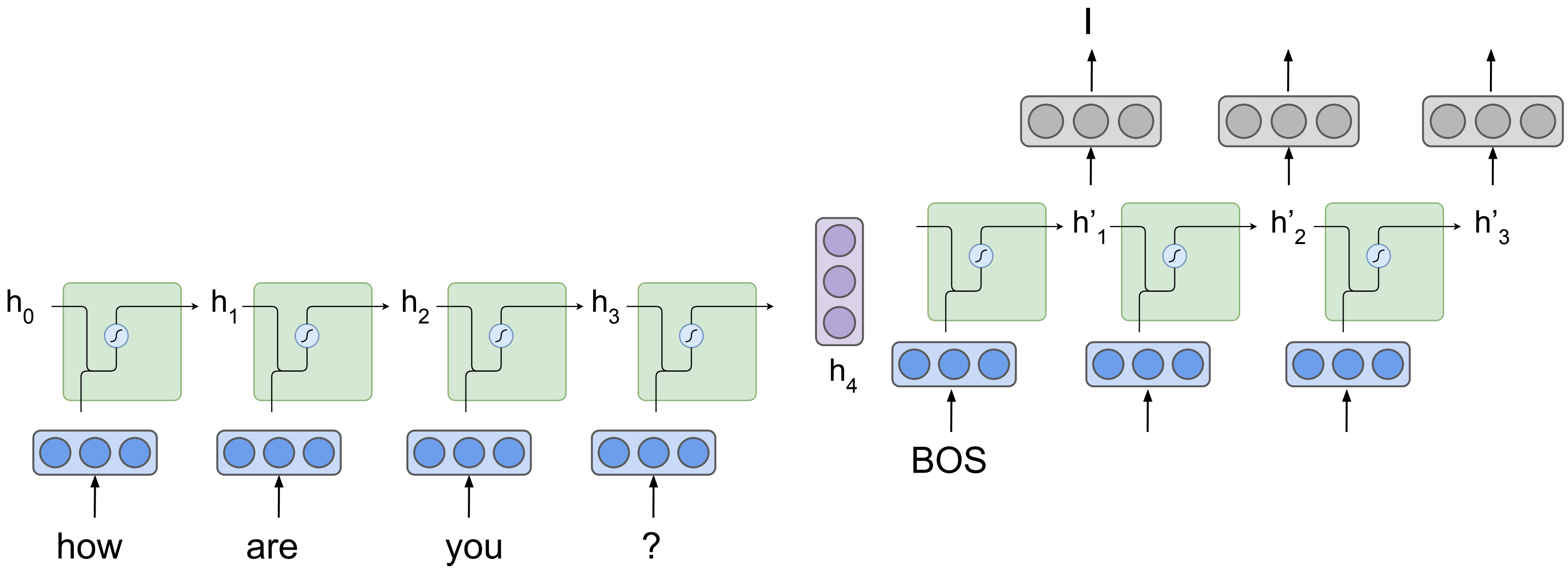
(* RNN Cell can be LSTM/GRU Cell)

Encoder Model + Decoder Model

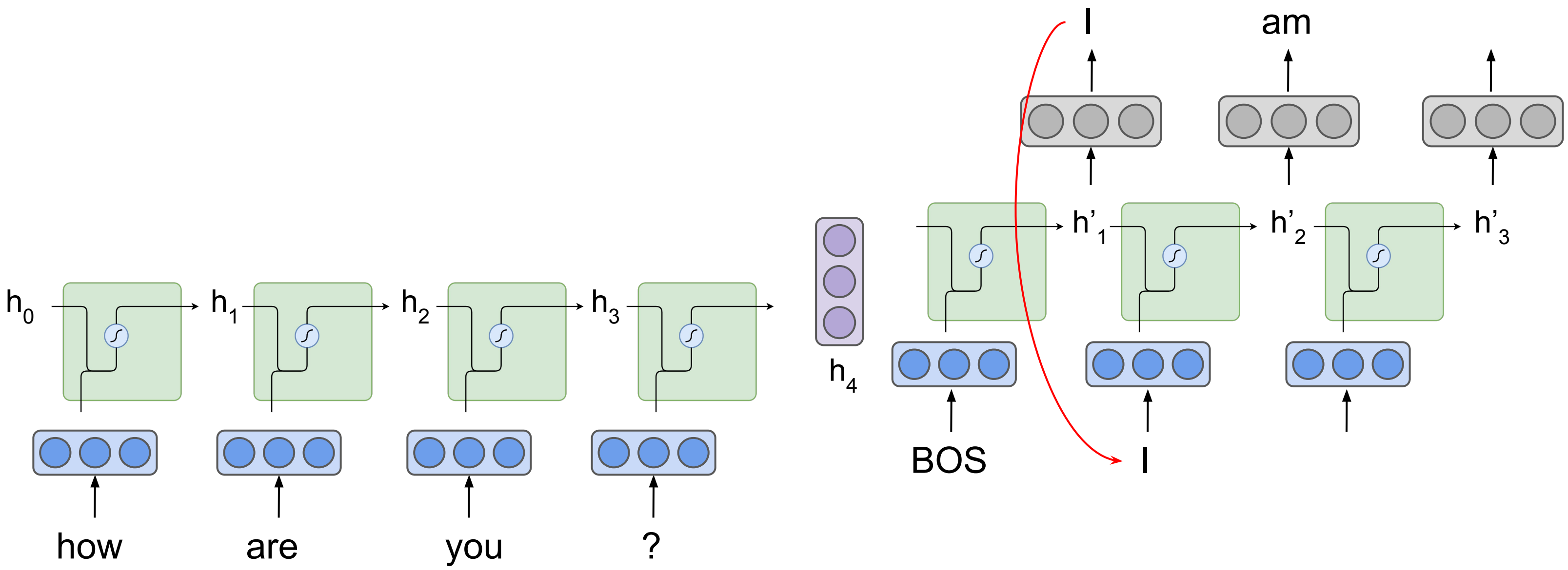
- Two different RNN models



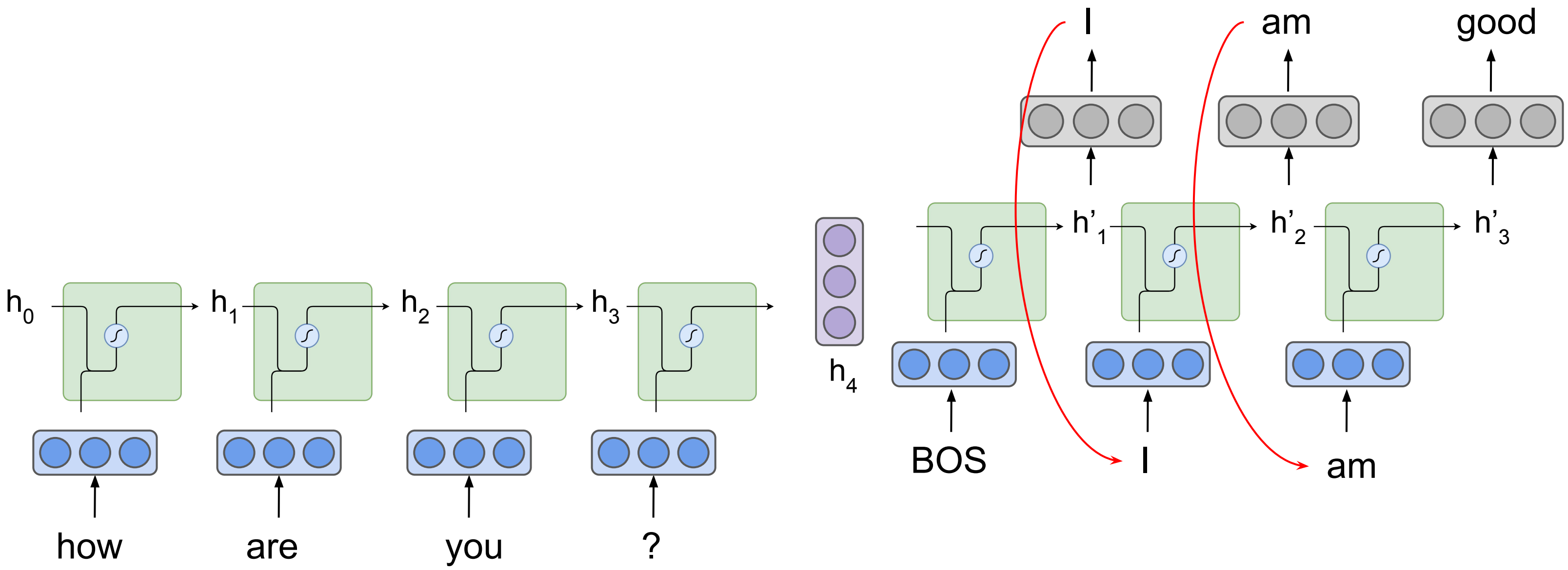
Example 1: Response Generation



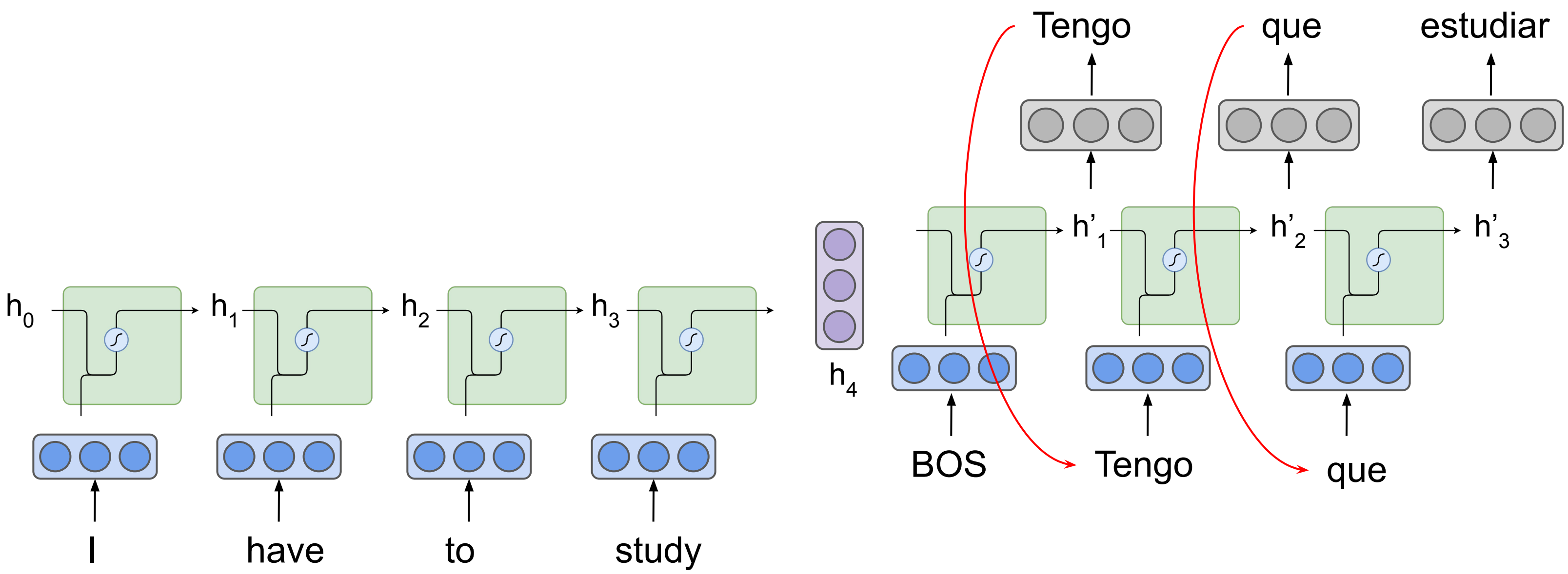
Example 1: Response Generation



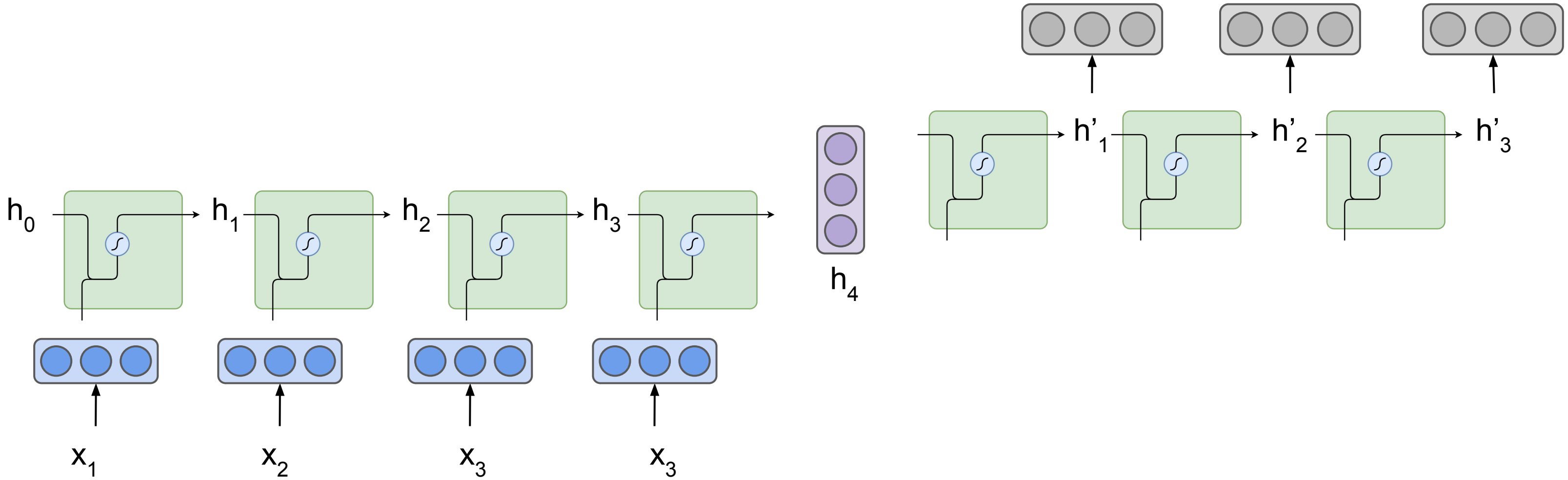
Example 1: Response Generation

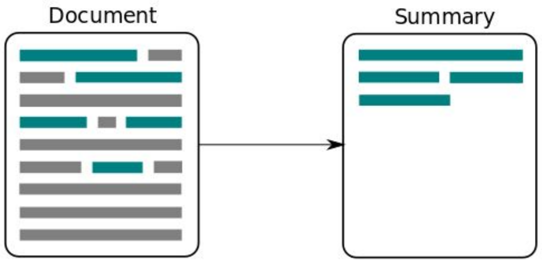


Example 2: Machine Translation



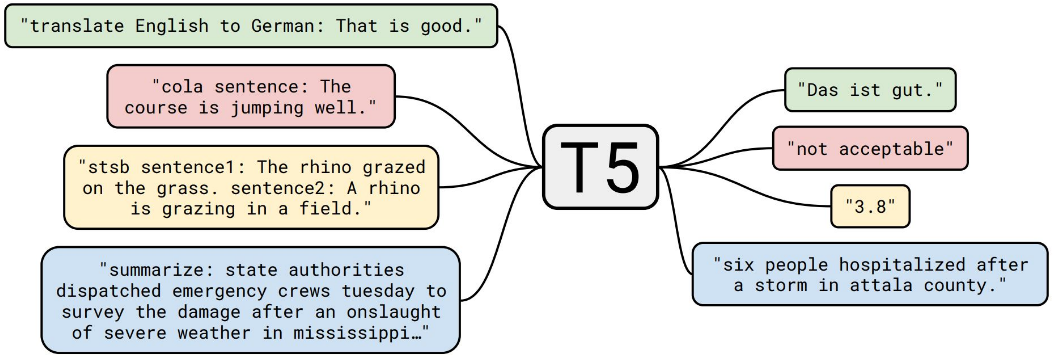
Example 3: Text Summarization





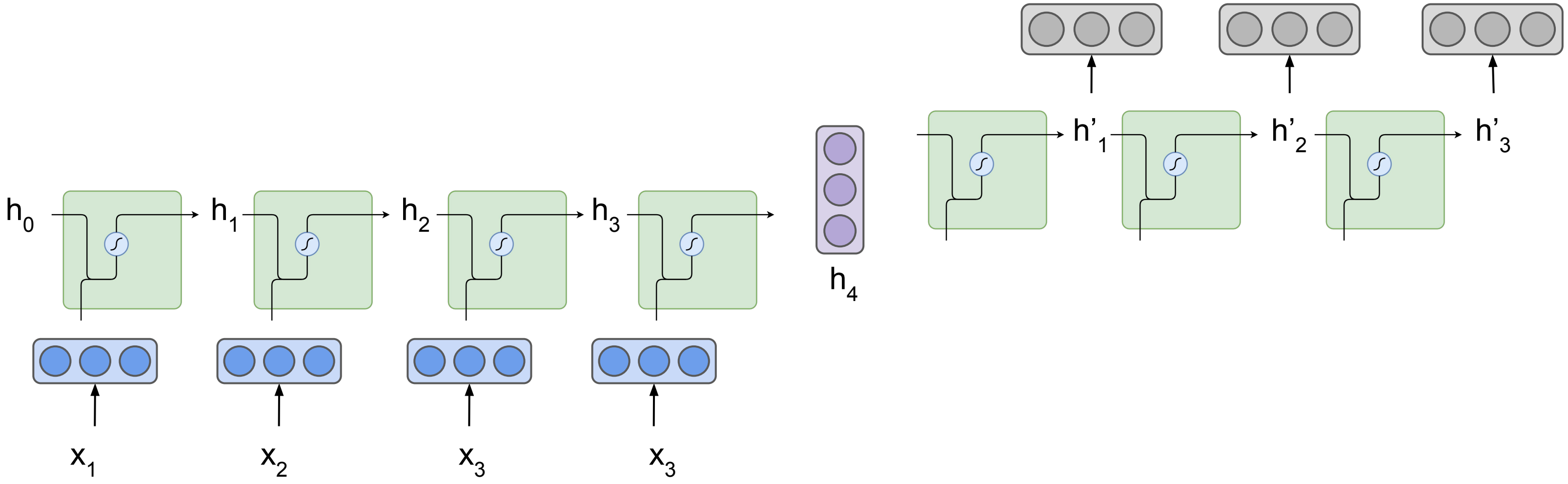
What Can We Do with Encoder-Decoder Models?

- Almost anything! (even classification!)
- …. as long as you have a sufficient amount of **parallel data** (**input-output pairs**)



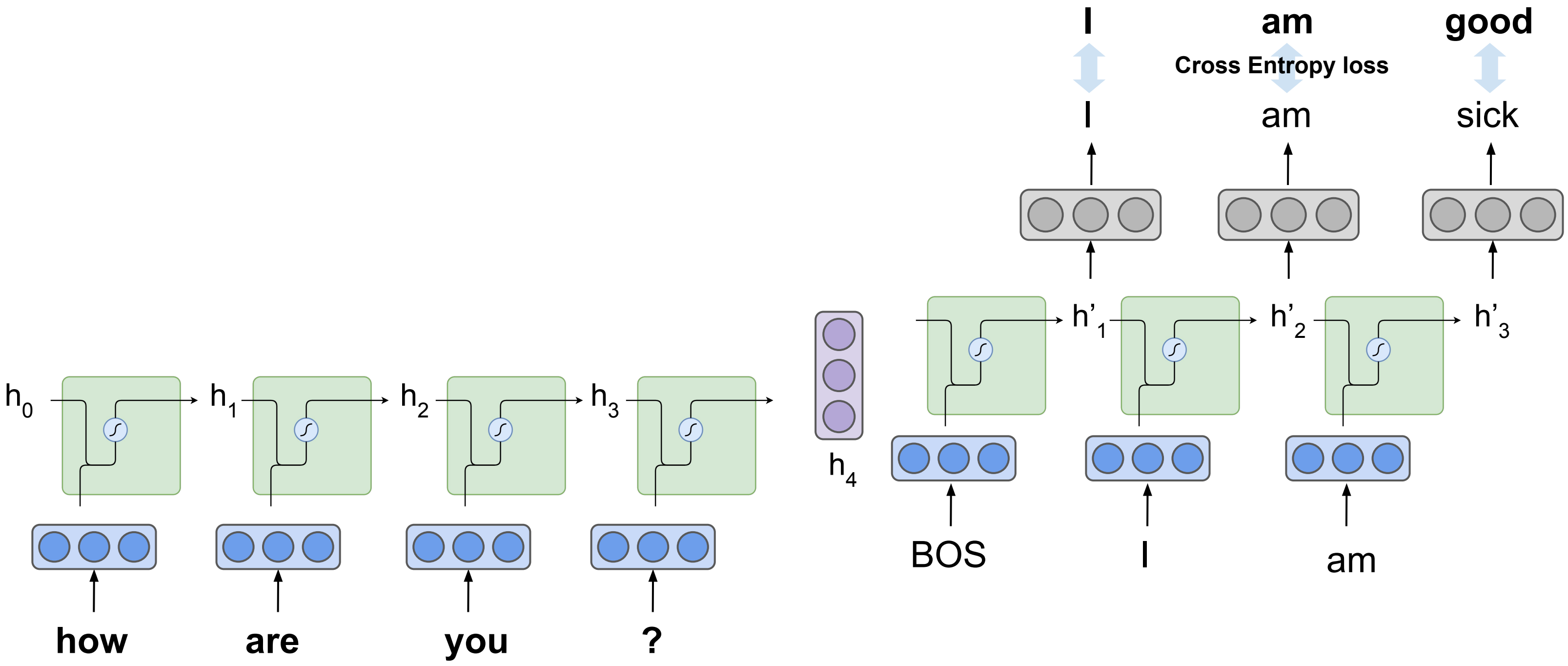
Check: Input & Output

- **Token ID** → One-hot vector → Dense vector → **RNN**
- **RNN** → Output layer + softmax → **Token ID**



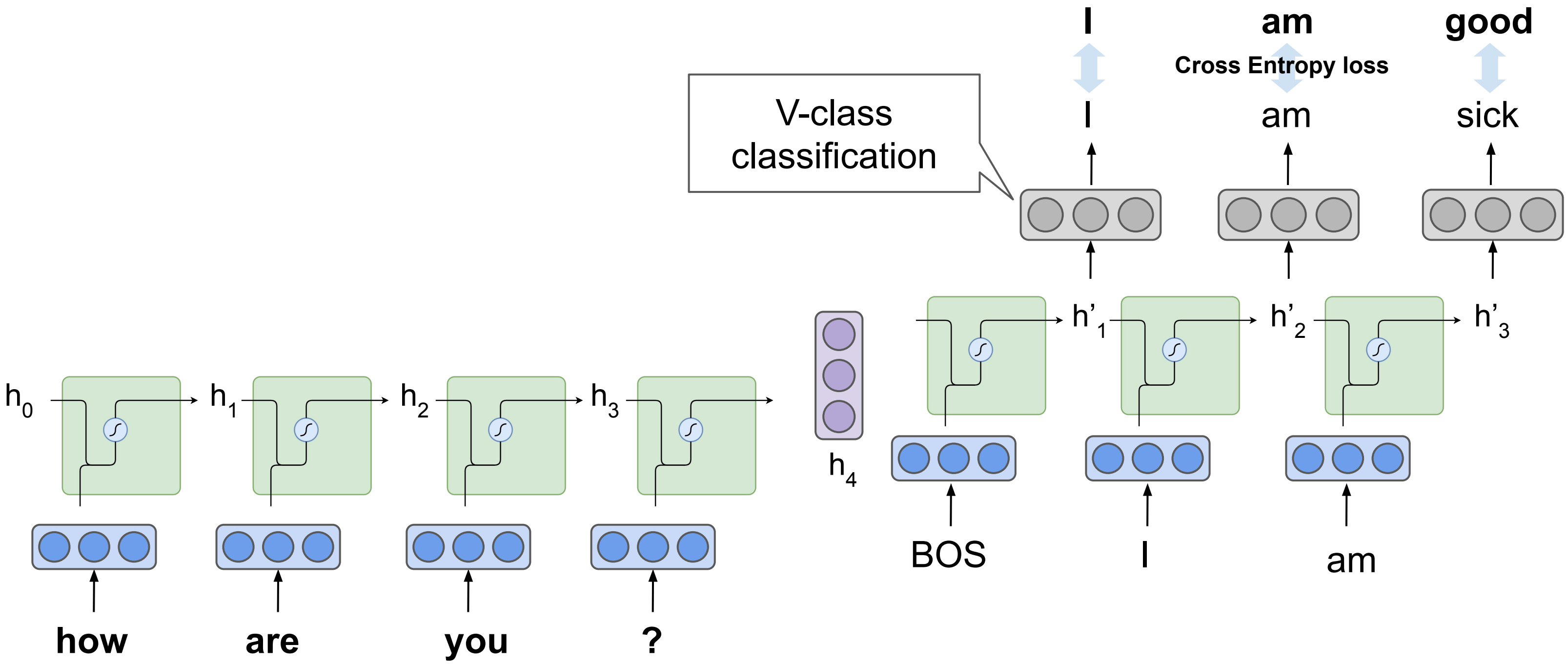
Training Encoder-Decoder Models: Training Data

- Training data: **Pairs of text sequences** (called a **parallel corpus)**
  - e.g., ("How are you?", "I am good")



Training Encoder-Decoder Models: “Teacher Forcing”

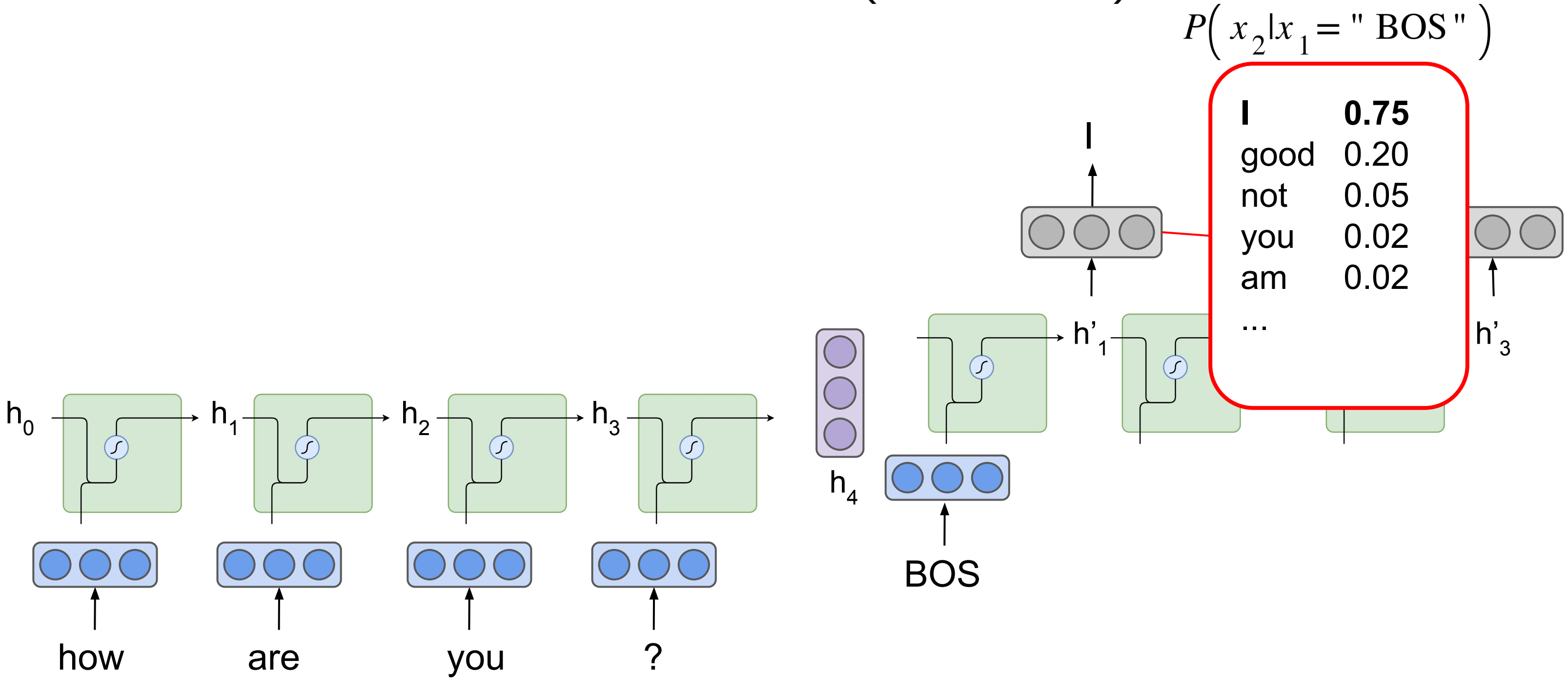
- Train the model to output **ground-truth outputs**



Decoding Algorithms

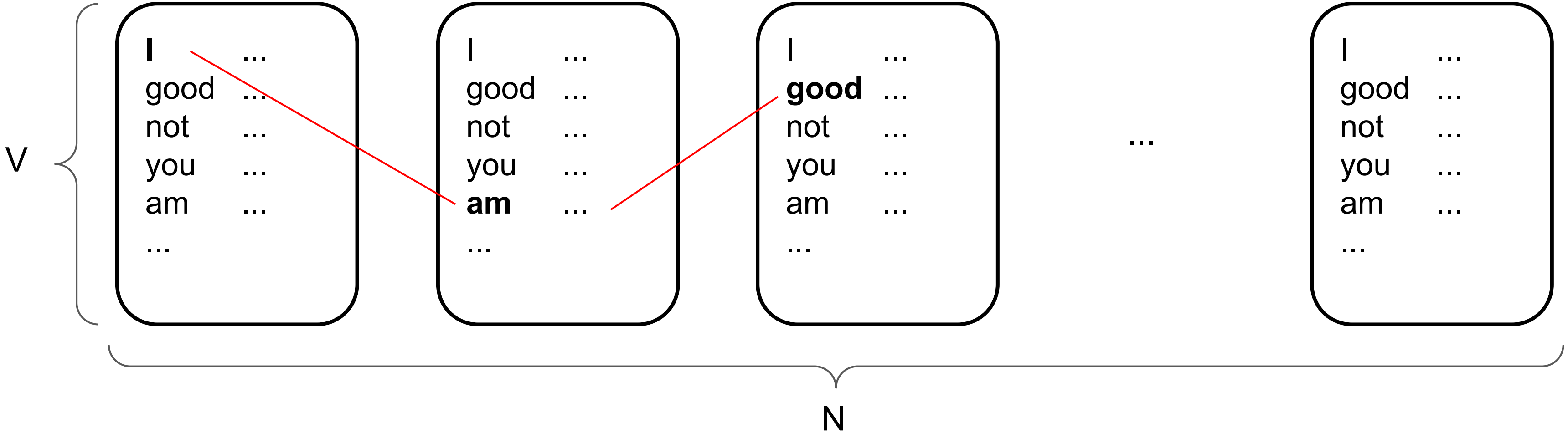
What does the output layer do at each step?

- **Token probability** given the context $P\left(x_t | x_{1:t-1}\right)$



Exhaustive Search is Intractable!

- We want to find the best sequence $\underset{X}{\operatorname{argmax}}\, P(X) = \underset{X}{\operatorname{argmax}} \prod_{t=1}^{N} P\left(x_t | x_{1:t-1}\right)$
- Number of candidates: V^N
  - for a sequence length of N and a vocabulary size of V

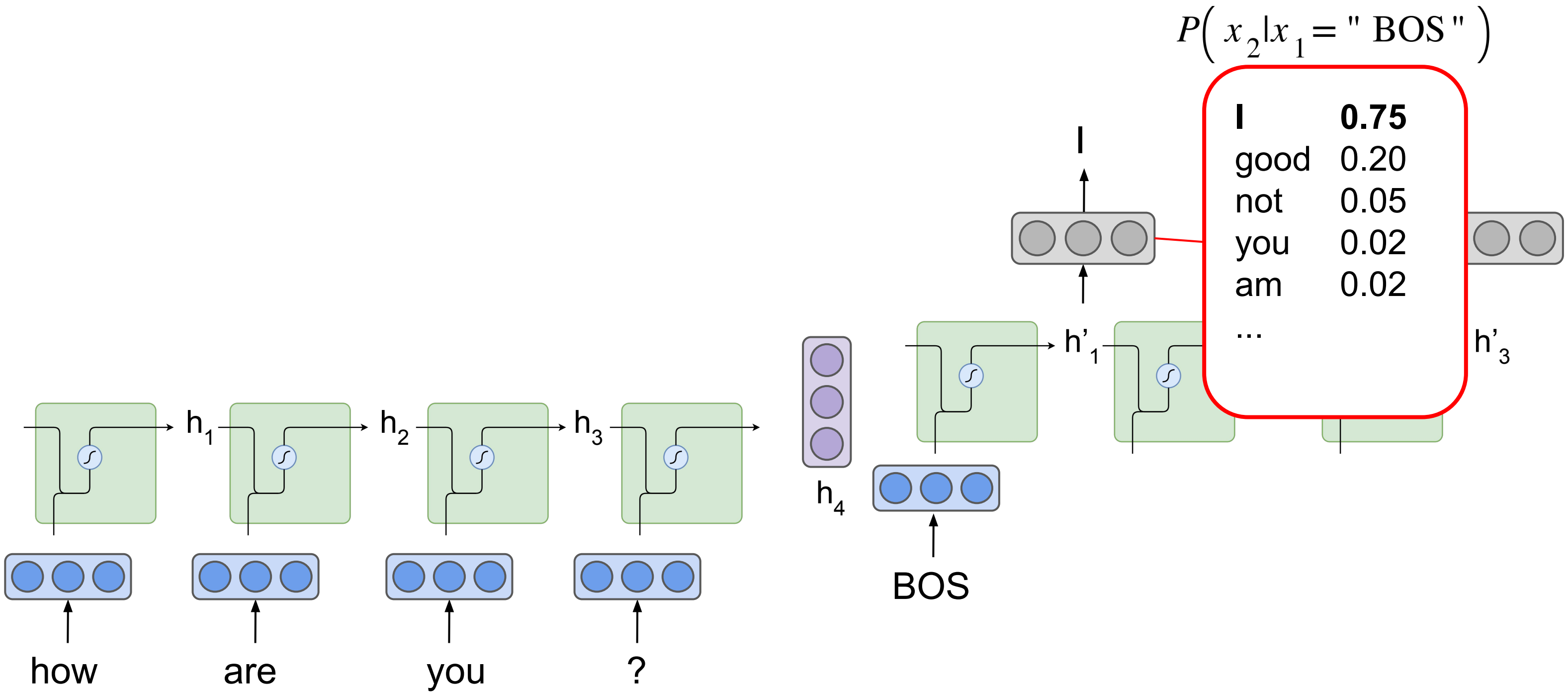


Decoding Algorithm (i.e., Approximate Search)

- Greedy Search
- Beam Search
- Top-k Sampling (covered in Week 8-9)
- Nucleus Sampling (covered in Week 8-9)

Greedy Search

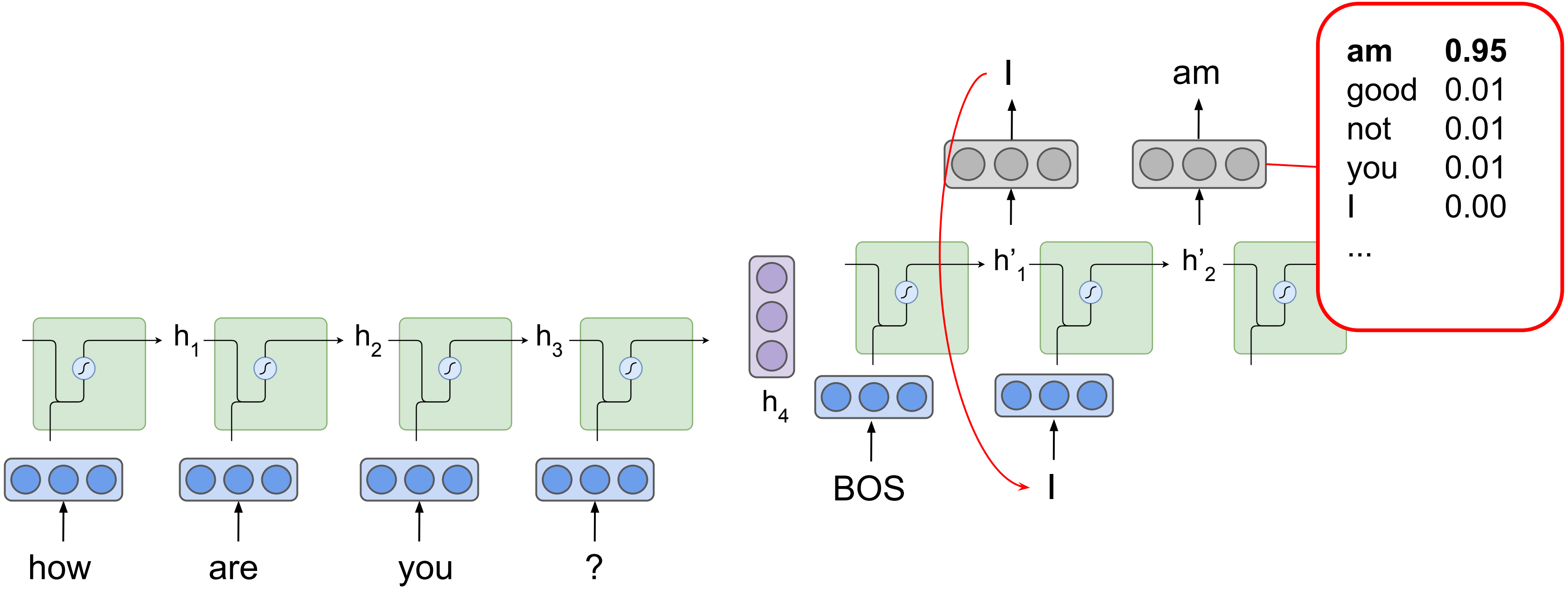
- Always choose the token with the highest probability



Greedy Search

- Always choose the token with the highest probability

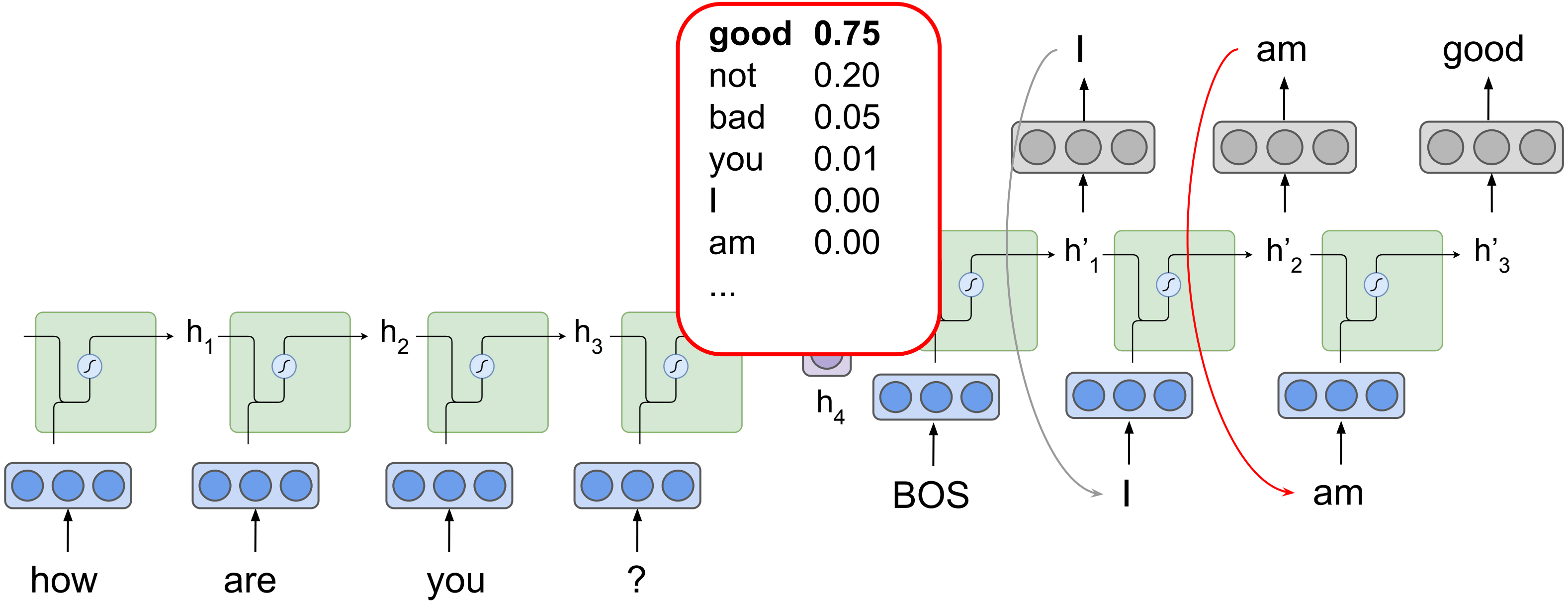
$$P\left(x_3 | x_{1:2} = (\text{"BOS"}, \text{"I"})\right)$$



Greedy Search

- Always choose the token with the highest probability

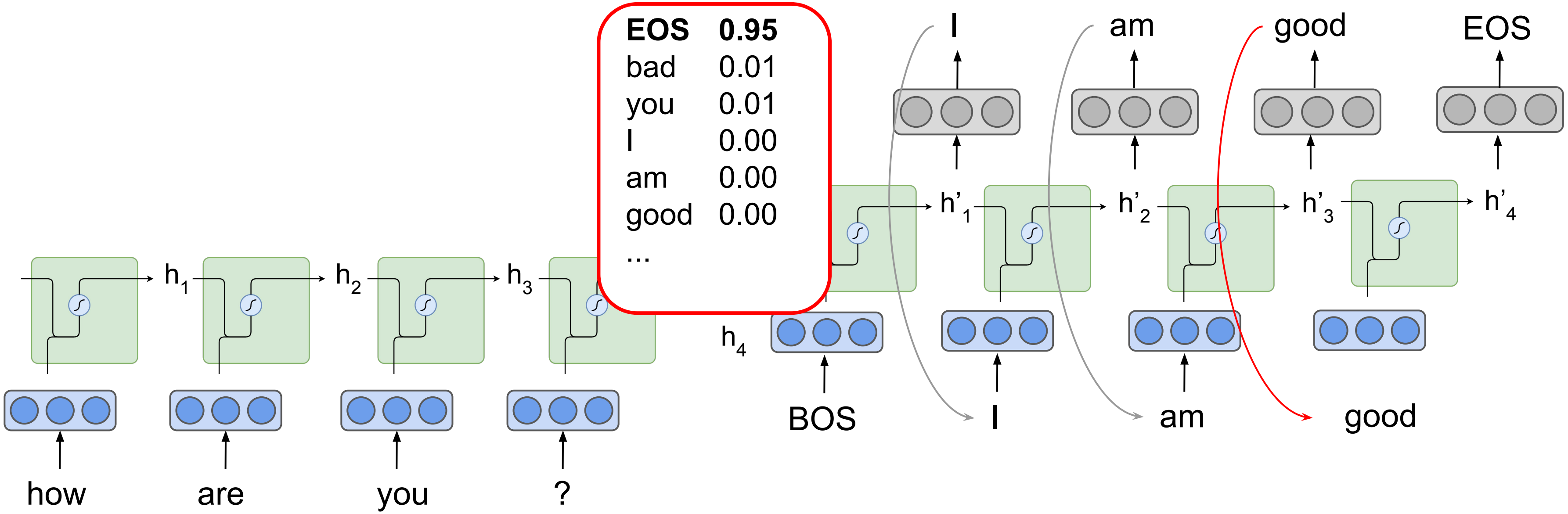
$$P\left(x_4 | x_{1:3} = (\text{"BOS"}, \text{"I"}, \text{"am"})\right)$$



When Does the Decoder Stop?

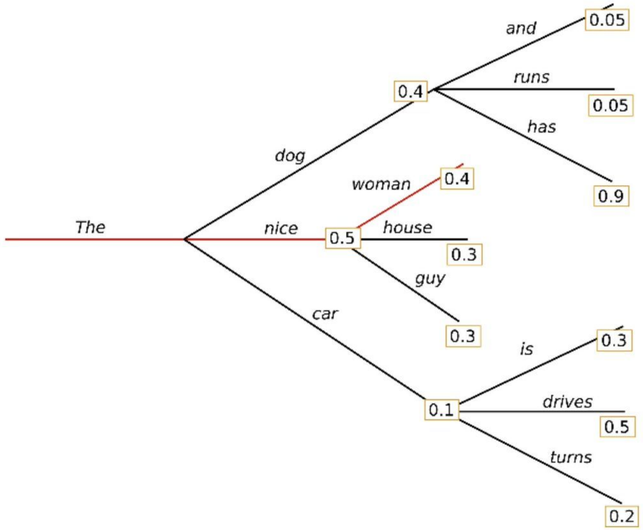
- When it selects the dummy symbol for the end of sequence (e.g., EOS)

$$P\left(x_5 | x_{1:4} = (\text{" BOS "}, \text{" I "}, \text{" am "}, \text{" good "})\right)$$



Beam Search

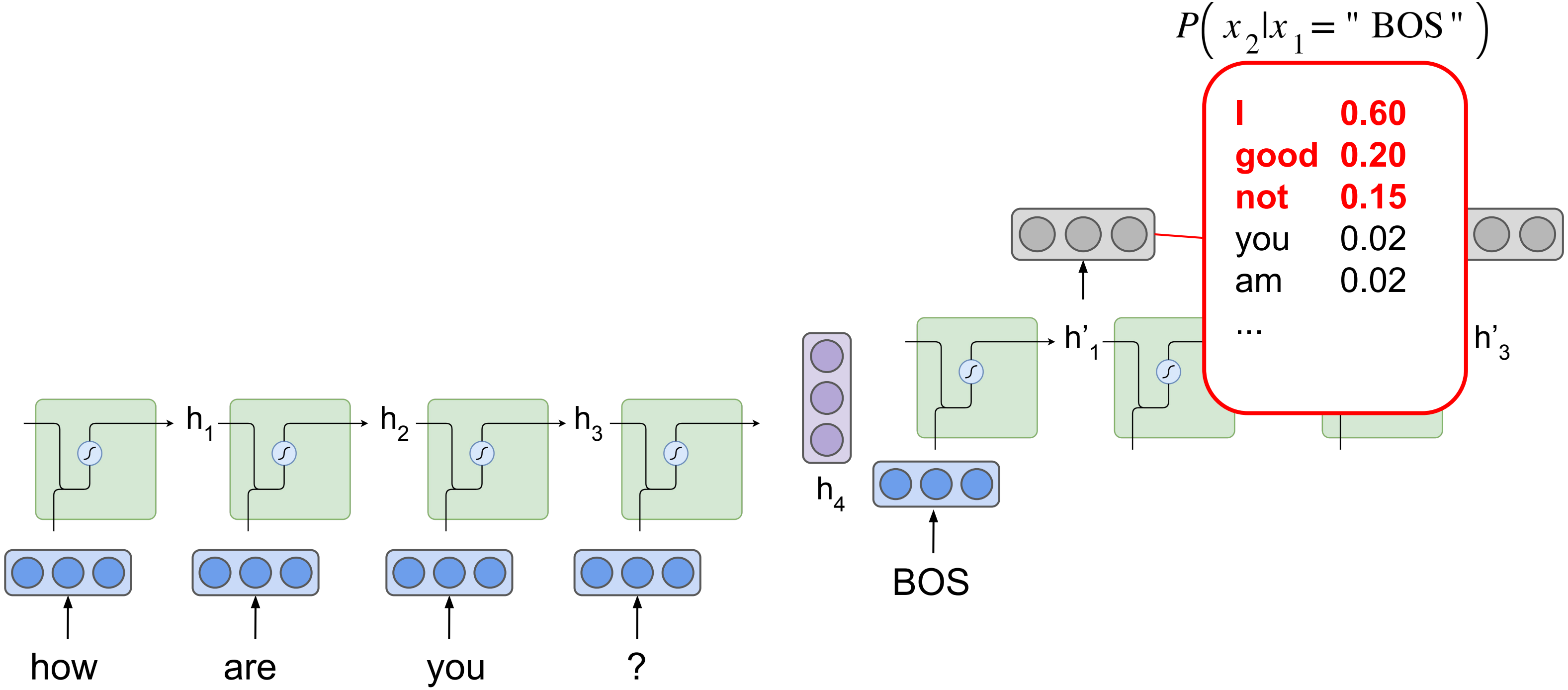
- Select top-k candidates at each step and grow the search tree



Beam Search (k = 3)

k is often called “beam width”

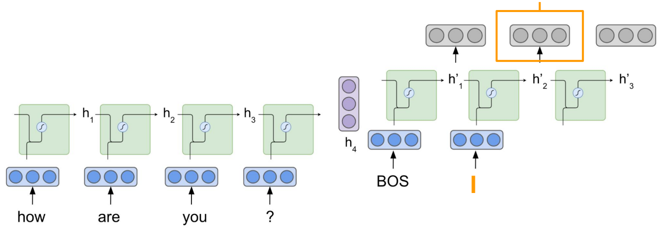
- Choose top-k tokens



Step 2: Calculate Next Token Probability

am	0.80
good	0.05
not	0.01
...	

I	0.40
good	0.20
not	0.19
you	0.02
am	0.02
...	

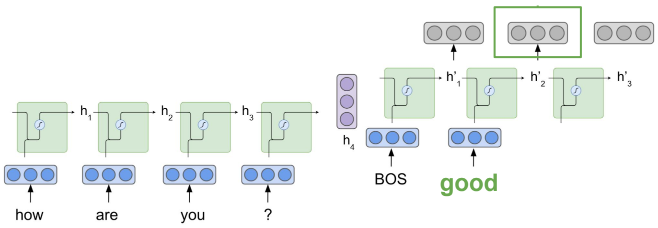


Step 2: Calculate Next Token Probability

I	0.40
good	0.20
not	0.19
you	0.02
am	0.02
...	

am	0.80
good	0.05
not	0.01
...	

EOS	0.80
good	0.02
not	0.01
...	



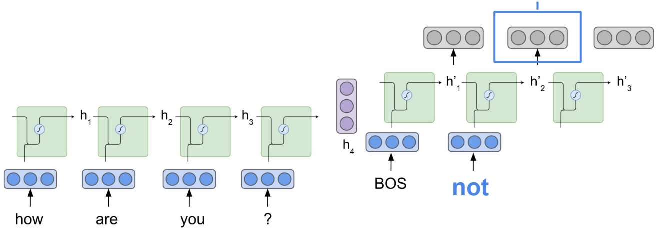
Step 2: Calculate Next Token Probability

I	0.40
good	0.20
not	0.19
you	0.02
am	0.02
...	

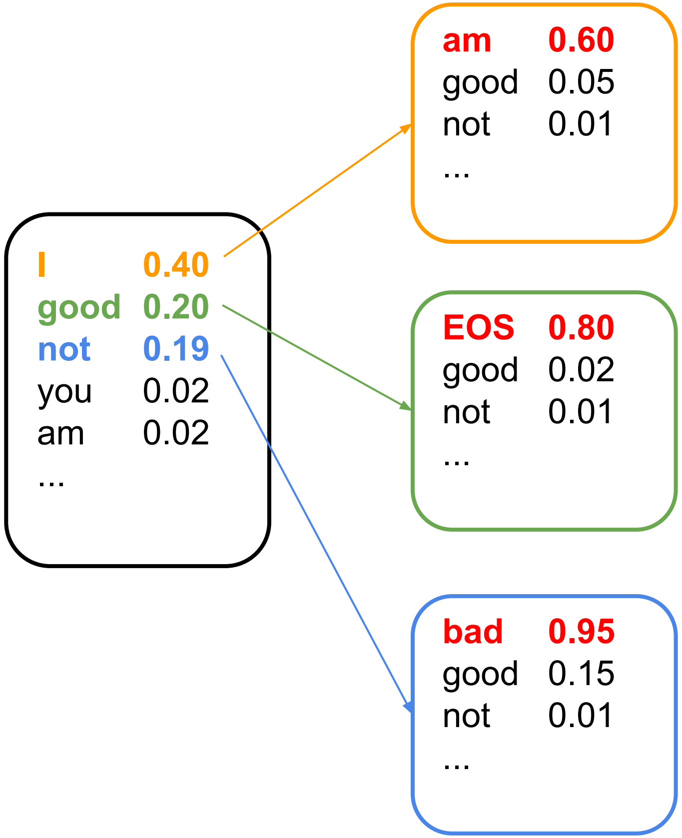
am	0.80
good	0.05
not	0.01
...	

EOS	0.80
good	0.02
not	0.01
...	

bad	0.95
good	0.15
not	0.01
...	



Step 3: Choose Top-3 Candidates



0.40 * 0.60 = 0.24
0.40 * 0.05 = 0.02
0.40 * 0.01 = 0.004

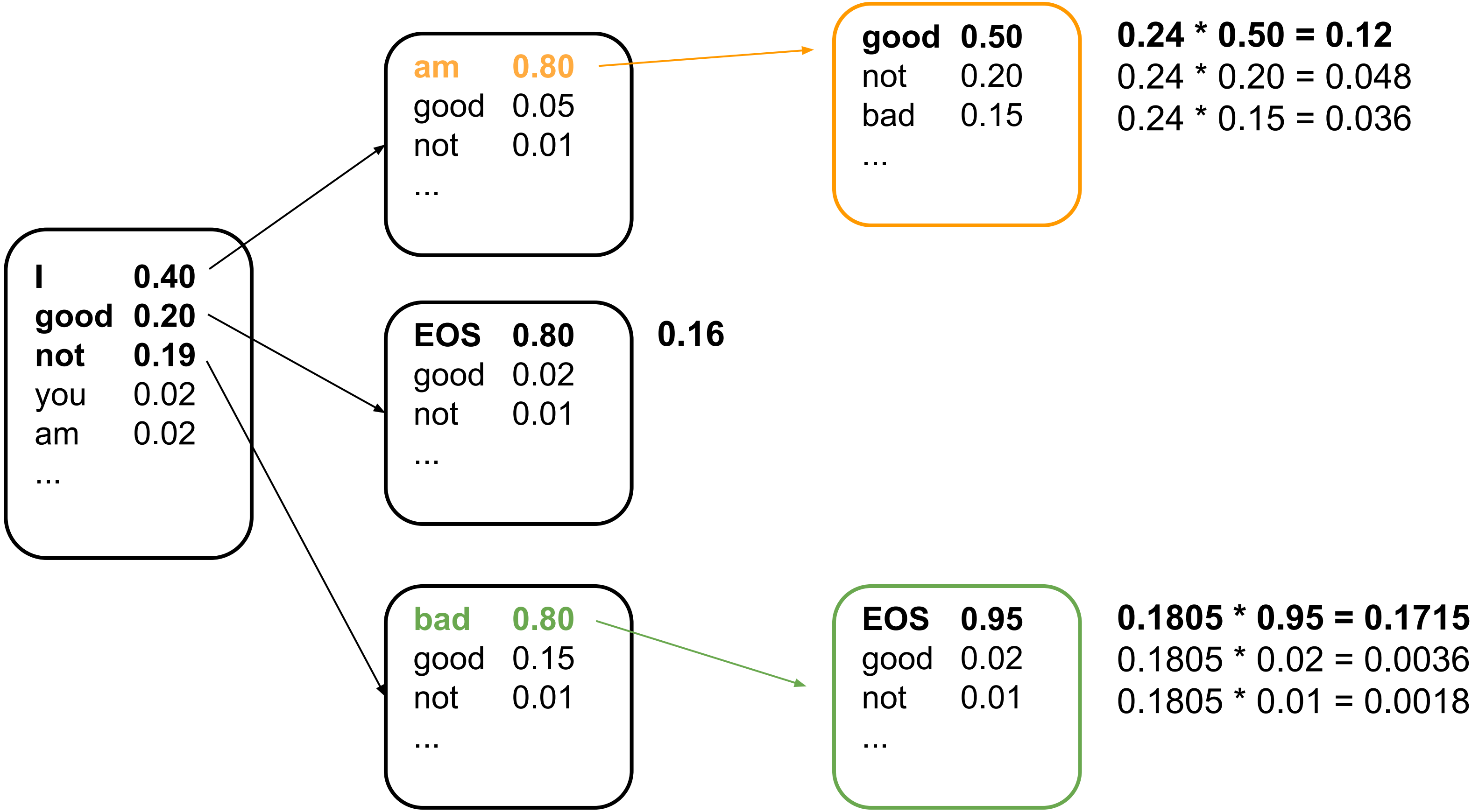
0.20 * 0.80 = 0.16
0.20 * 0.02 = 0.004
0.20 * 0.01 = 0.002

0.19 * 0.95 = 0.1805
0.19 * 0.15 = 0.0285
0.19 * 0.01 = 0.0019

Candidate ranking

1: “I am ...”	0.24
2: “not bad ...”	0.1805
3: “good”	0.16

Repeat the Steps



Update the Candidate List

I	**0.40**
good	**0.20**
not	**0.19**
you	0.02
am	0.02
...	

am	0.80
good	0.05
not	0.01
...	

EOS	**0.80**
good	0.02
not	0.01
...	

0.16

bad	**0.80**
good	0.15
not	0.01
...	

good	**0.50**
not	0.20
bad	0.15
...	

0.24 * 0.50 = 0.12
0.24 * 0.20 = 0.048
0.24 * 0.15 = 0.036

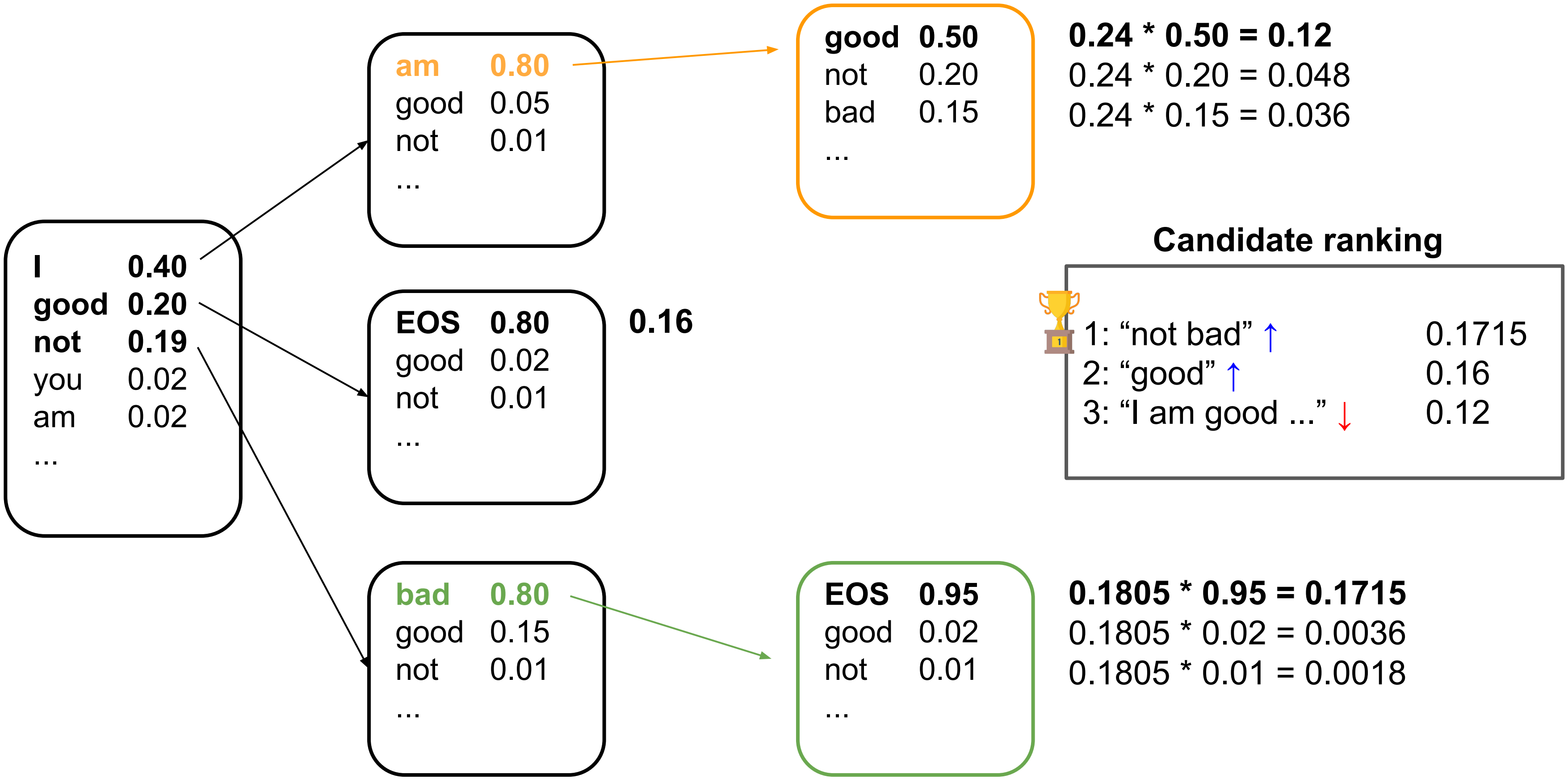
EOS	**0.95**
good	0.02
not	0.01
...	

0.1805 * 0.95 = 0.1715
0.1805 * 0.02 = 0.0036
0.1805 * 0.01 = 0.0018

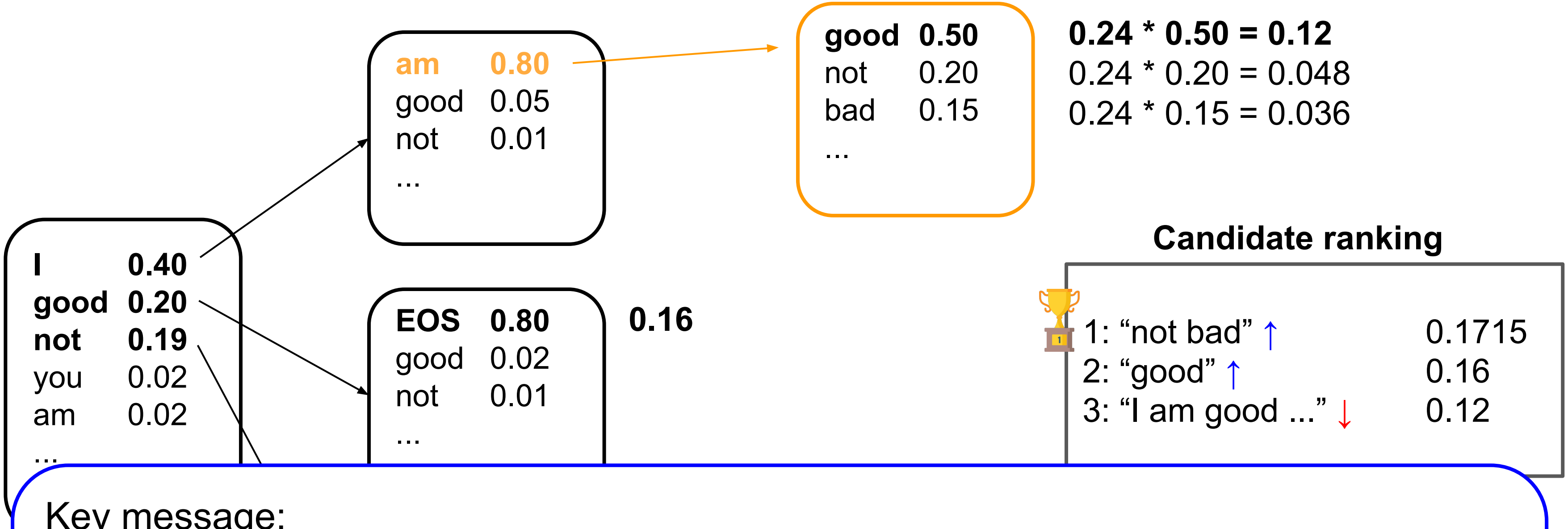
Candidate ranking

1: “not bad” ↑	0.1715
2: “good” ↑	0.16
3: “I am good ...” ↓	0.12

Update the Candidate List



Update the Candidate List



Key message:
The **same model** can return **a different sequence** with a different decoding algorithm

Greedy: “I am good”
Beam Search (k=2): “good”
Beam Search (k=3): “not bad”

Summary: Decoding Algorithm

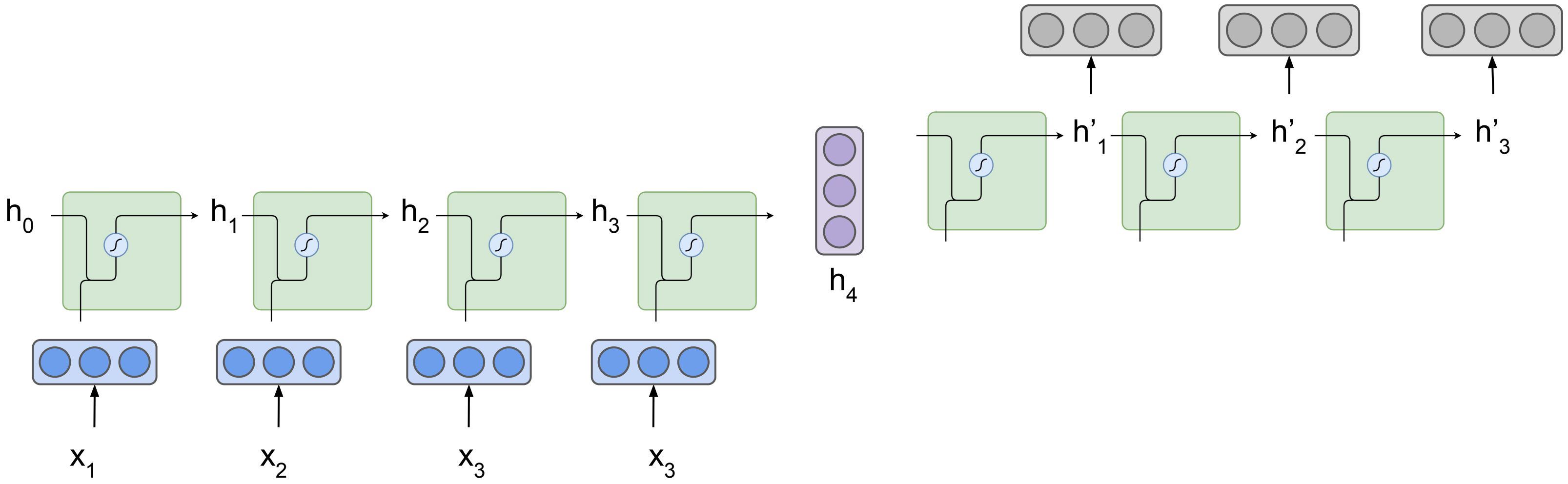
- Greedy Search
  - Time complexity: O(N)
- Beam Search
  - Time complexity: O(k N)
  - Common choice: Beam width = 5 ~ 10

Key Concept

- Encoder-Decoder Model

  = Encoder + Decoder + Decoding algorithm

  Model **Not model**

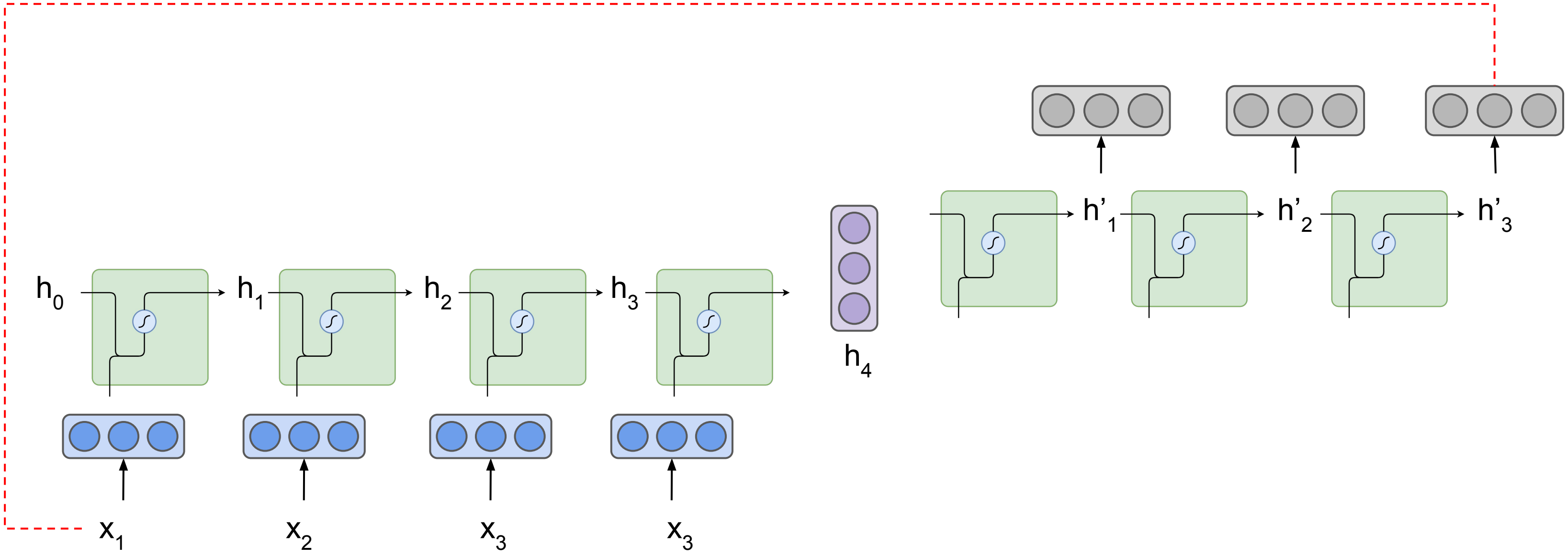


Questions?

Attention Mechanism

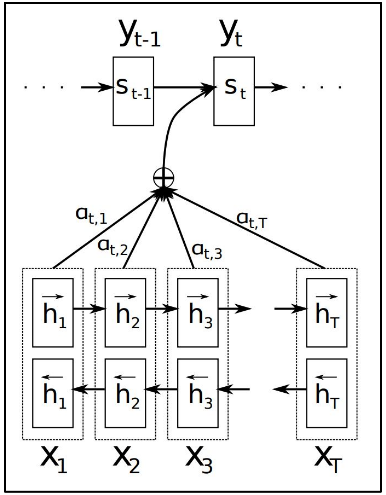
What's the issue with Encoder-Decoder Models?

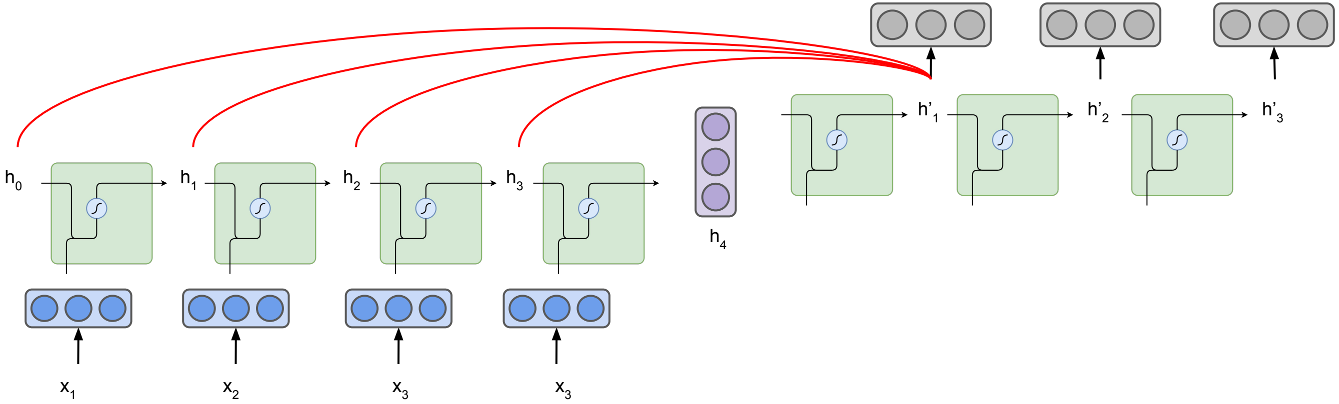
- Not a very good design for **long input/output sequence** even with LSTM/GRU
  - e.g., dependency b/w the beginning of input & the end of output



Attention Mechanism [Bahdanau, Cho, and Bengio ICLR 2016]

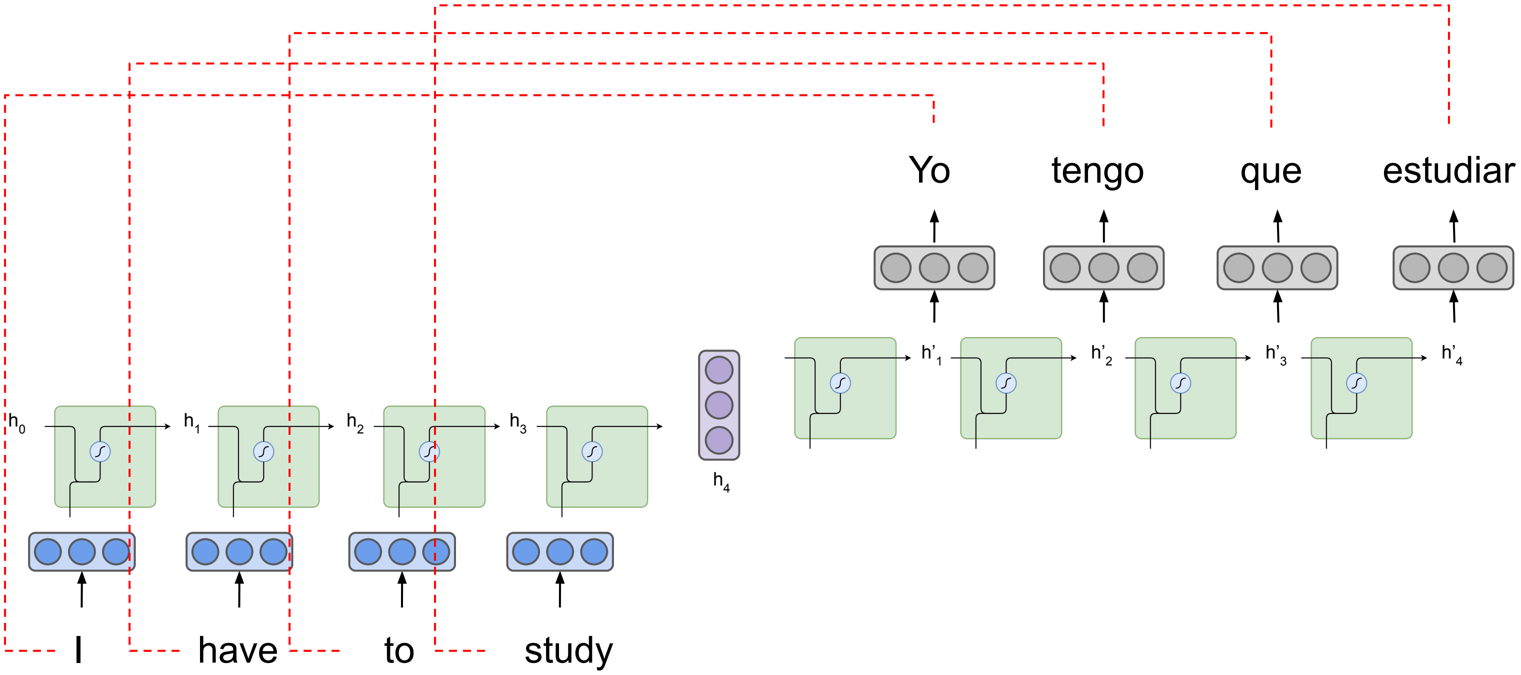
- Additional input to the decoder based on **the alignments b/w encoder and decoder steps**



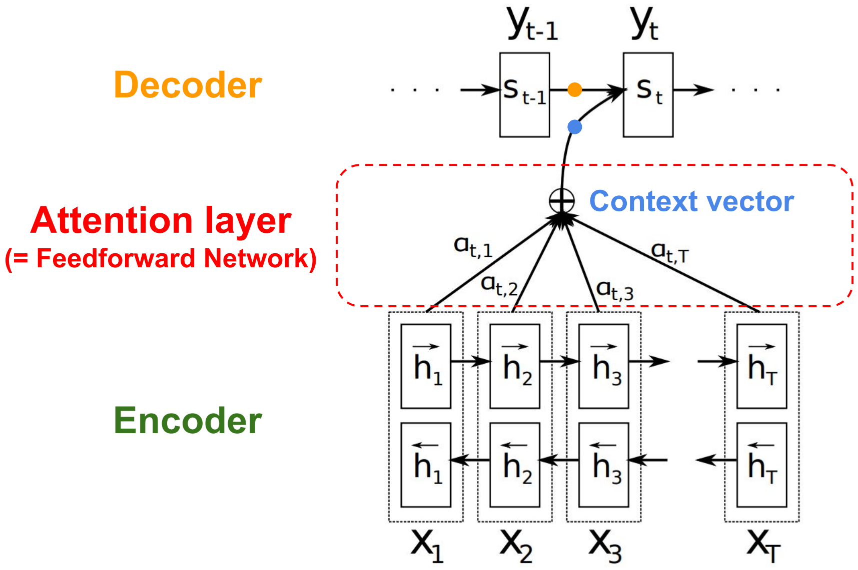


Attention Mechanism: Intuition

- "Direct" connections between any steps in the encoder/decoder model



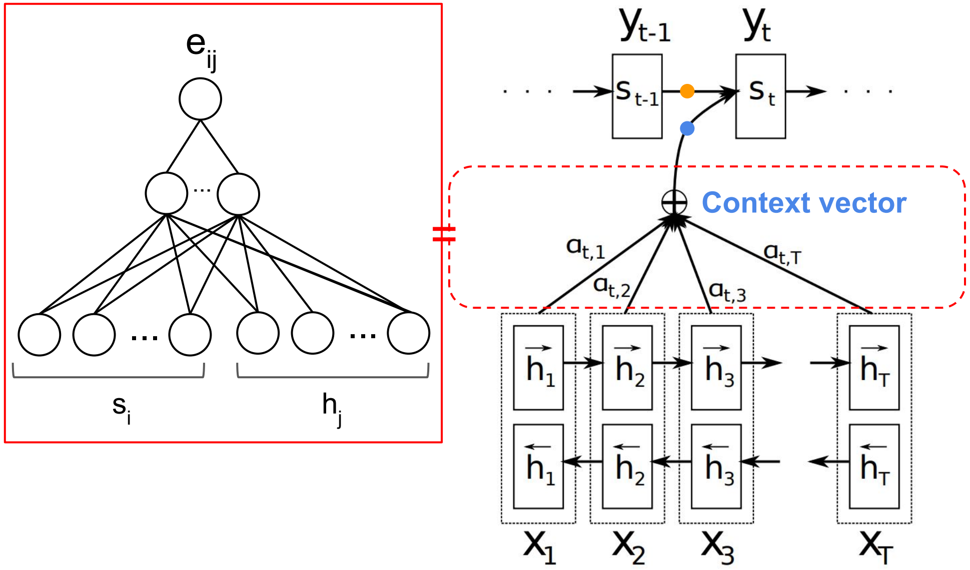
Attention Mechanism: Attention Layer



Previous hidden state
+ context vector

$$\underline{c_i} = \sum_{j=1}^{T_x} \underbrace{\alpha_{ij}}_{\text{alignment score}} \underline{h_j}$$

Attention Layer in Depth



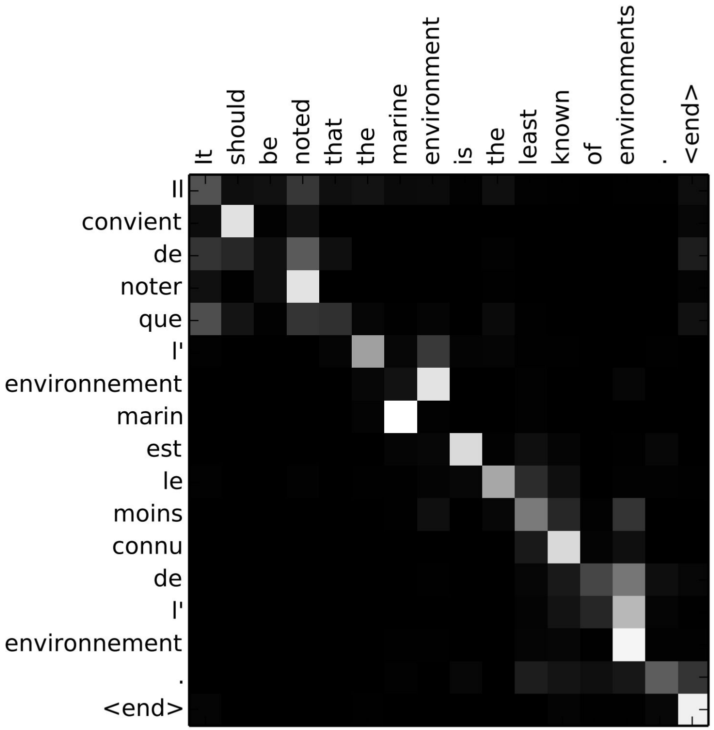
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

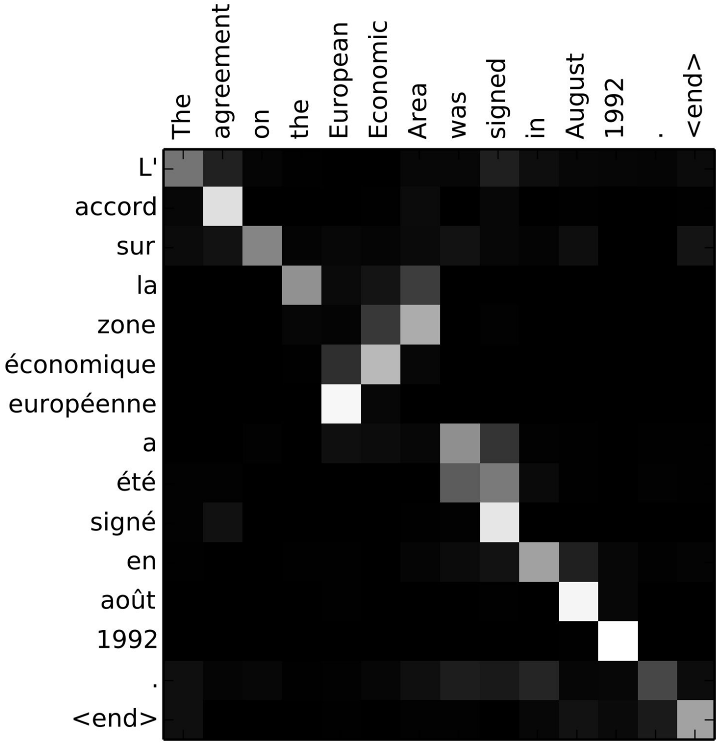
$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)}$$

$$e_{ij} = \text{score}\left(s_i, h_j\right)$$

$$\boxed{\text{score}\left(s_i, h_j\right) = \mathbf{v}_\mathbf{a}^\mathbf{T} \tanh\left(\mathbf{W}_a [s_i; h_j]\right)}$$

Attention Weight Analysis

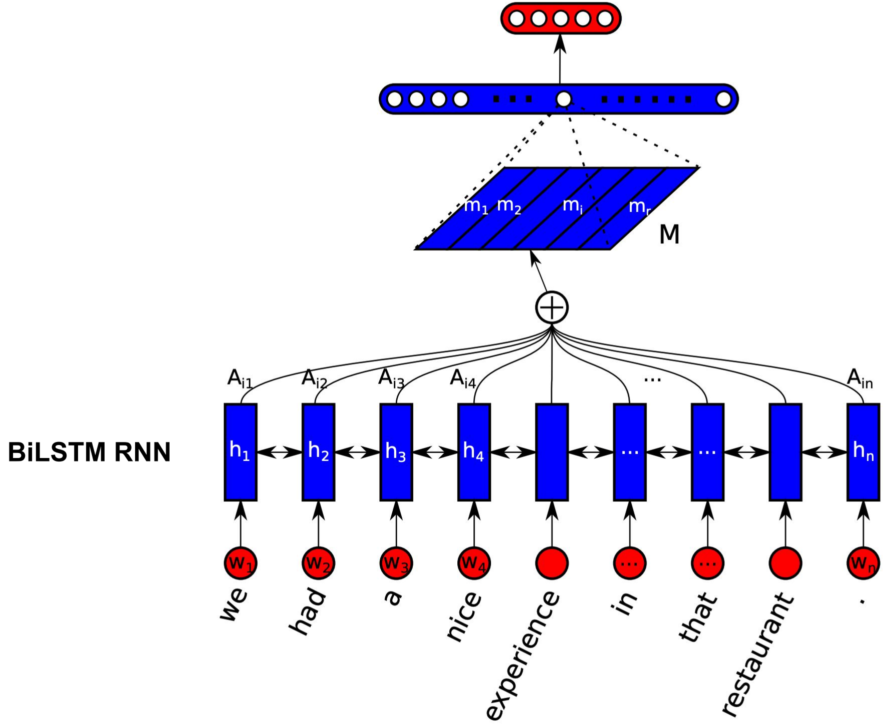




Alignment Score Options

	Name	Alignment score function	Citation
	Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Original →	Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
	Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
	General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
	Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Transformer →	Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

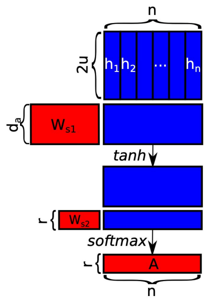
Self-Attention Mechanism for Text Classification



$$M = AH$$

$$A = softmax\left(W_{s2} tanh\left(W_{s1} H^T\right)\right)$$

$$H = (\mathbf{h_1}, \mathbf{h_2}, \cdots \mathbf{h_n})$$



Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, Yoshua Bengio, A Structured Self-attentive Sentence Embedding, ICLR 2017.

Takeaway

- Encoder-Decoder Models (aka seq2seq Models)
  - Many applications: Response Generation, Machine Translation, Text Summarization, etc.
- Decoding Algorithms
  - Greedy Search
  - Beam Search
- Attention mechanism
  - Alignment scores
  - Self-attention mechanism

Next Week

- Tue 10/5 Review session & initial discussion about term projects
  - Quick survey: Do you have any ideas in your mind? (Text? Image? or else?)
- Thu 10/7 Midterm exam (written exam on campus)
  - Time: 2pm-3:20pm
  - Location: TBA
- Fall Break!