Recurrent Neural Language Models

CSCI 601-471/671 (NLP: Self-Supervised Models)

JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

https://self-supervised.cs.jhu.edu/sp2024/

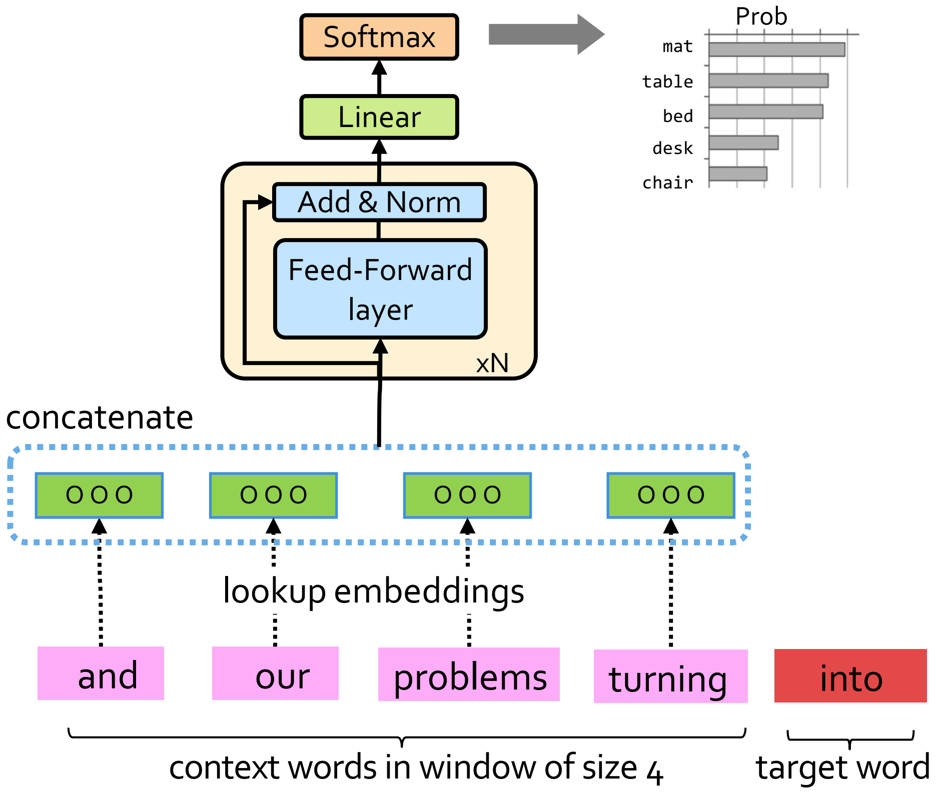
Logistics review

- How was the quiz 1?
  - Sweeter than a piece of cake.
  - Good, but somewhat challenging. About what I expected.
  - Feels like walking through the jungle barefoot... while juggling chainsaws.

- Next few homework assignments:
  - Will likely involve GPUs
  - Will likely be in groups
  - Think about who you may want to be teammates with.

Recap

- **Neural Language Models:** neural networks trained with LM objective.
- **Fixed-window Neural LM:** first of many neural LMs we will see in this class.



What Changed from N-Gram LMs to Neural LMs?

- What is the source of Neural LM's strength?
- Why sparsity is less of an issue for Neural LMs?

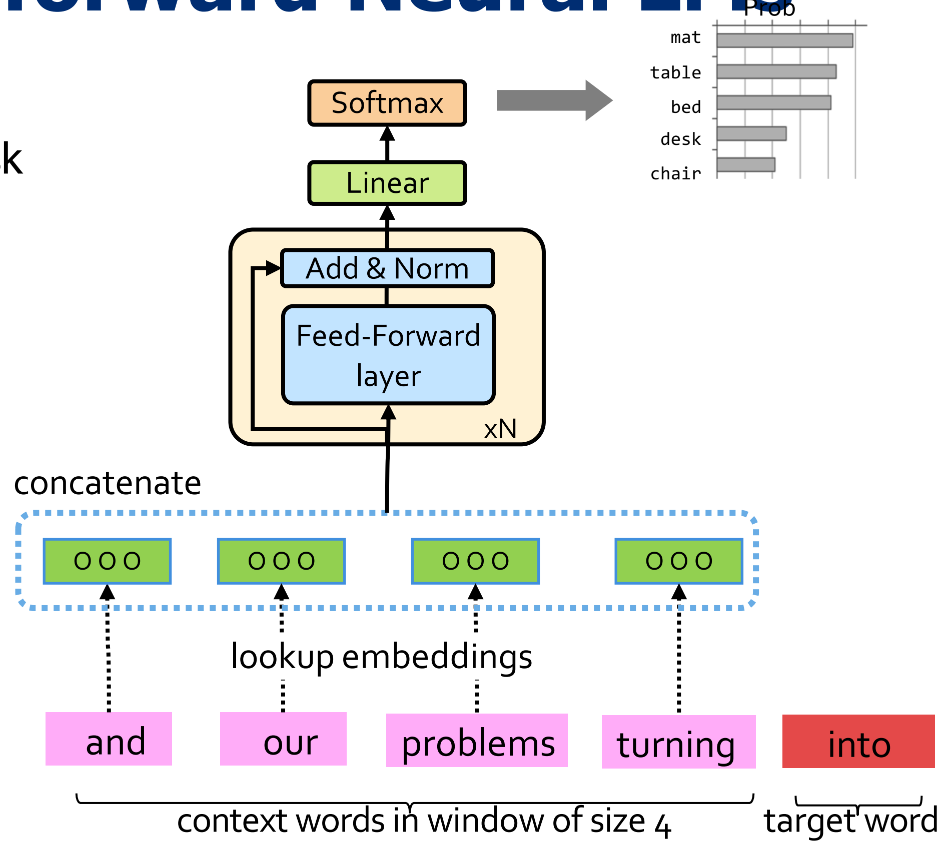
- **Answer:** In n-grams, we treat all prefixes independently of each other! (even those that are semantically similar)

```
students opened their ___
pupils opened their ___
scholars opened their ___
undergraduates opened their ___
students turned the pages of their ___
students attentively perused their ___
...
```

Neural LMs are able to share information across these semantically-similar prefixes and overcome the sparsity issue.

Moving Beyond Feedforward Neural LMs

- Are competitive at language modeling task
- However, they
  - have difficulty in remembering long range dependencies
  - have a fixed window size
- Key question: how to better capture long-range dependencies?
- Alternative here: a new family of neural networks: recurrent nets



Chapter Goals

1. Introducing Recurrent Neural Networks (RNNs)
2. Training RNNs
3. RNNs for natural language, particularly for language modeling
4. RNNs: Pros and Cons
5. Algorithms for sampling from LMs
6. Use [Pre-trained] LMs for downstream tasks

Chapter goals — Getting comfortable with RNNs for language modeling and the use of LMs for solving down-stream tasks.

Recurrent Neural Nets

Infinite Use of Finite Model

- Main question: how can a **finite** model a **long** (infinite) context?

- Solution: recursion! (recursive use of a model)

- RNNs are a family of neural networks introduced to **learn sequential data** via **recursive** dynamics.
- Inspired by the temporality of human thoughts

[Jeff Elman, "Finding structure in time," 1990]

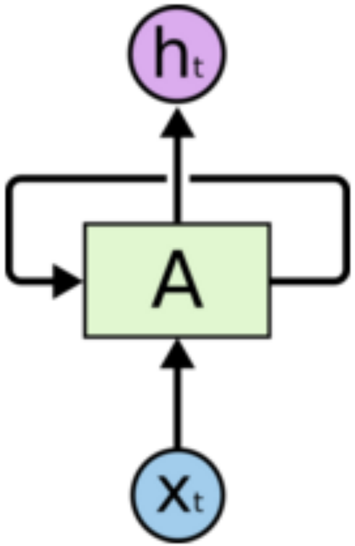
Recurrent Neural Networks (RNNs)

new state

old state

Input vector at t

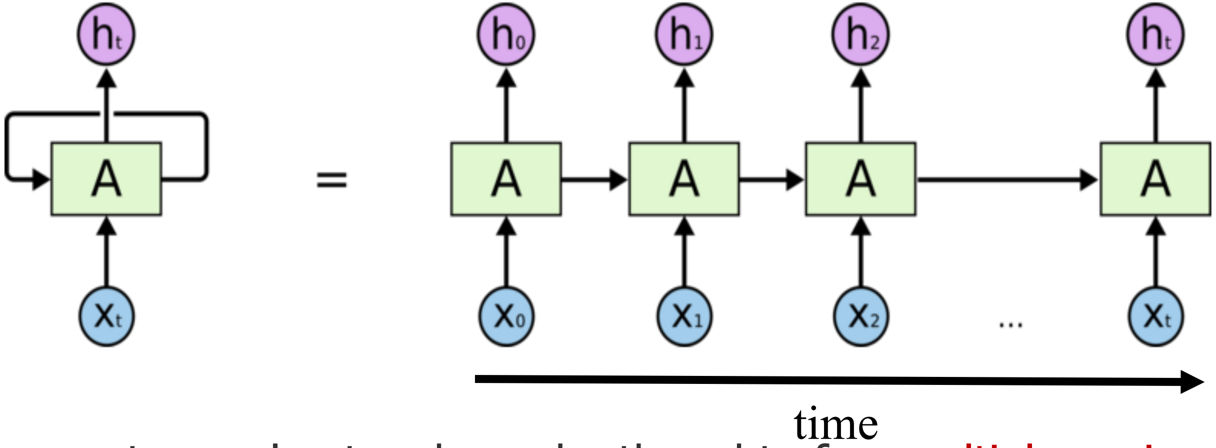
$$h_t = f(h_{t-1}, x_t)$$



- In the diagram, $f(.)$ looks at some **input** x_t and its **previous hidden state** h_{t-1} and outputs **a revised state** h_t.
- A loop allows information to be passed from one step of the network to the next.

Unrolling RNN

- The diagram above shows what happens if we **unroll the loop**.

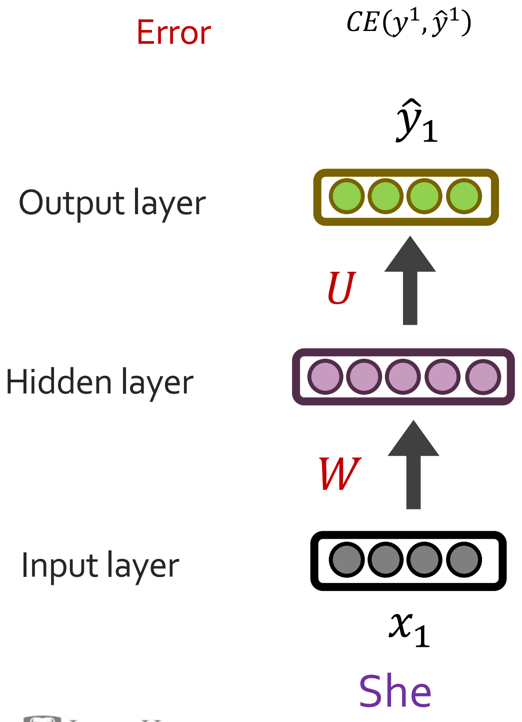


- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

[Jeff Elman, "Finding structure in time," 1990]

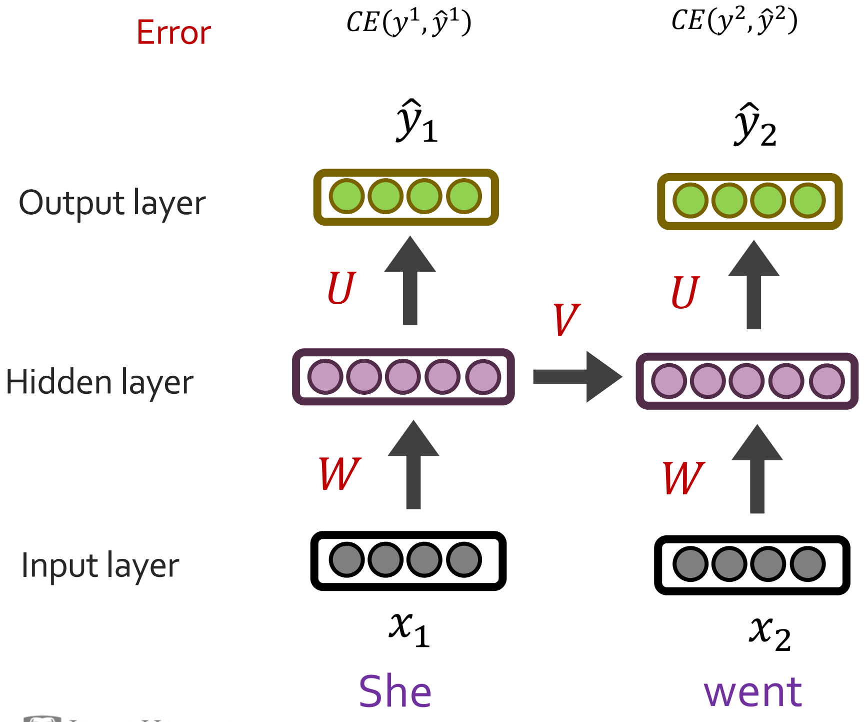
RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN: Forward Propagation

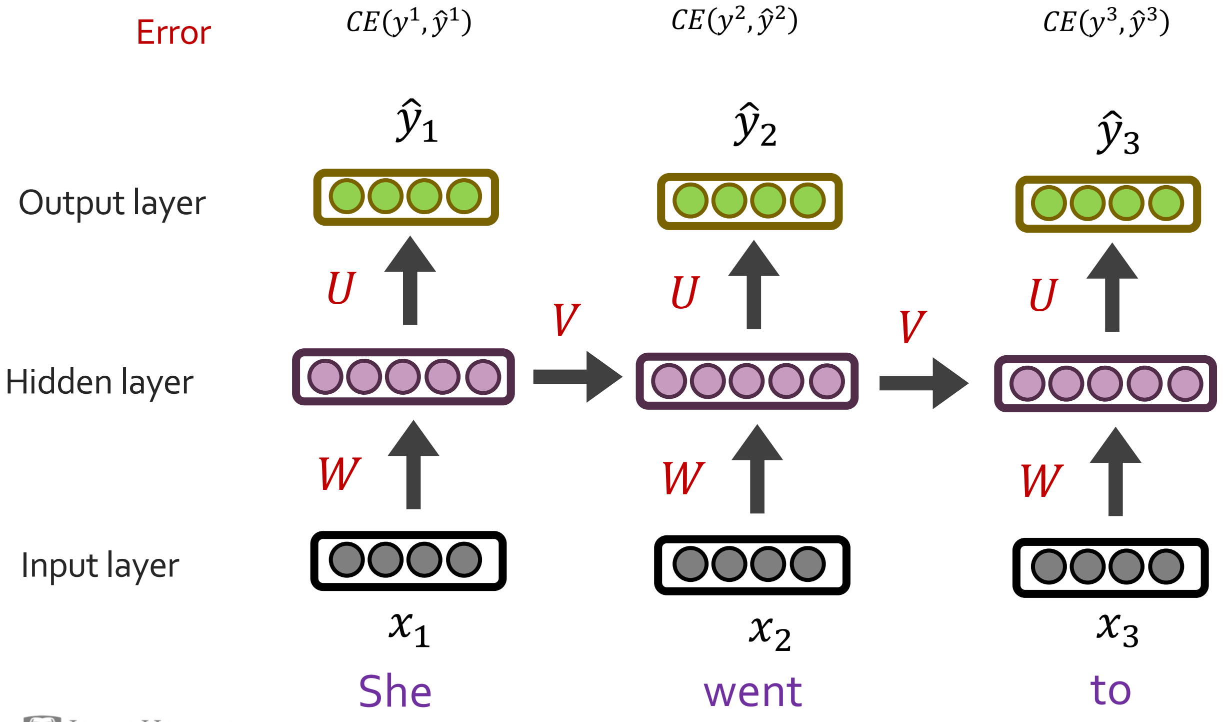
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



[Slide credit: Chris Tanner]

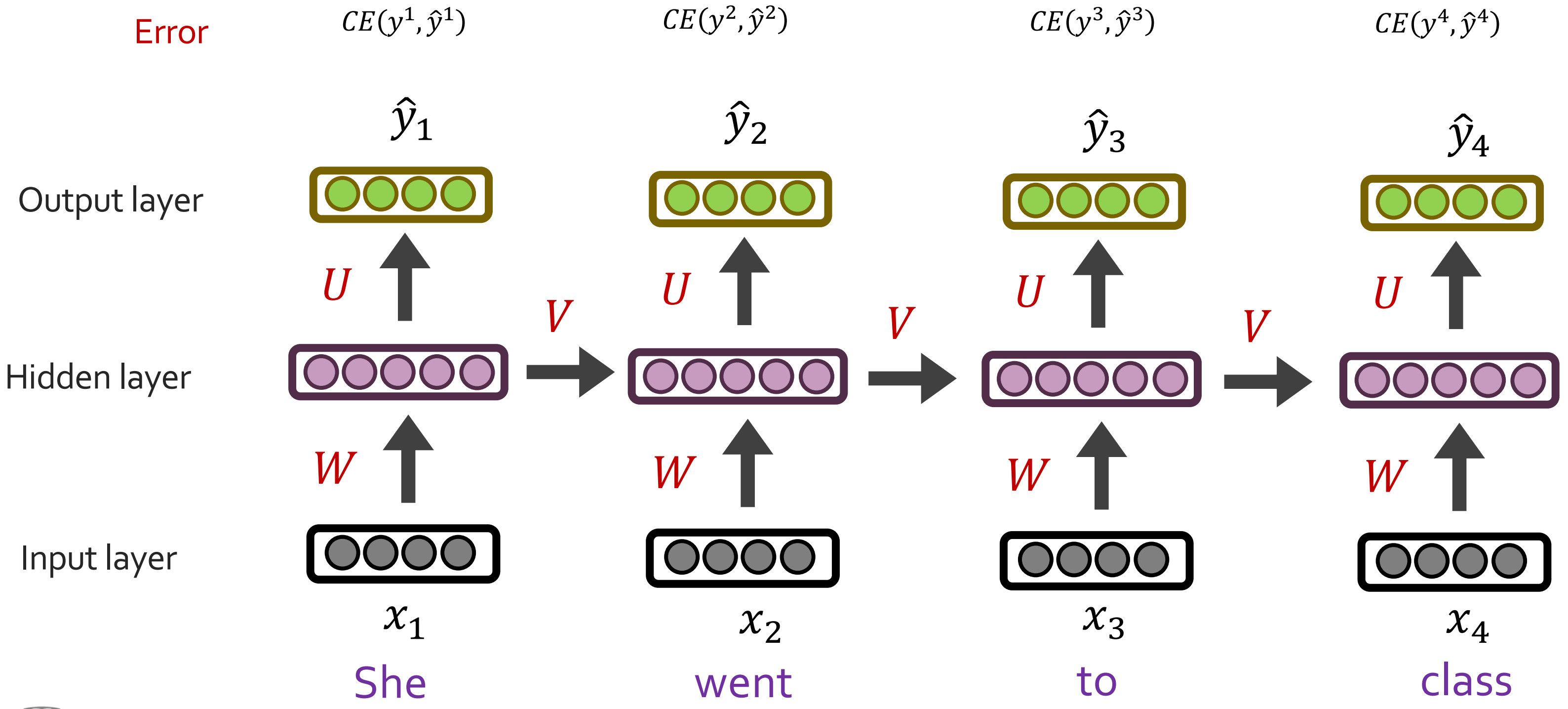
RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$

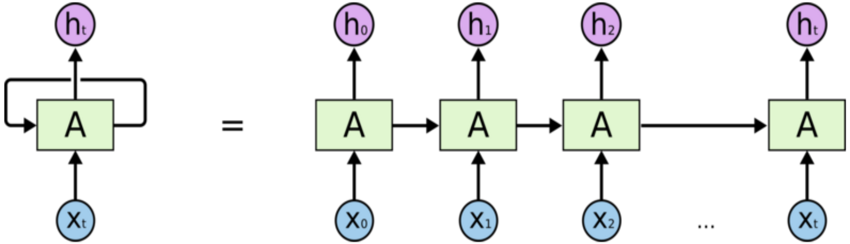


[Slide credit: Chris Tanner]

Summary

- Main idea behind RNNs: Infinite use of finite structure.

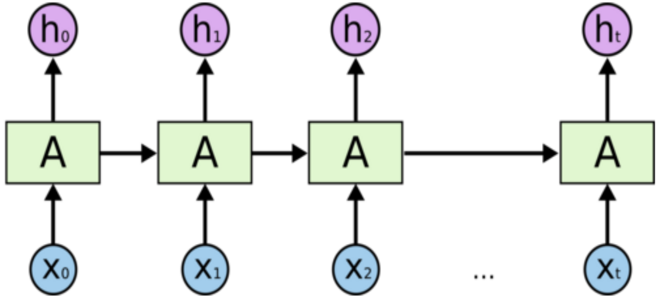
- Inference is a repeated use of a same model over time.



- Next: how do you train RNNs?

Recurrent Neural Networks and Natural Language

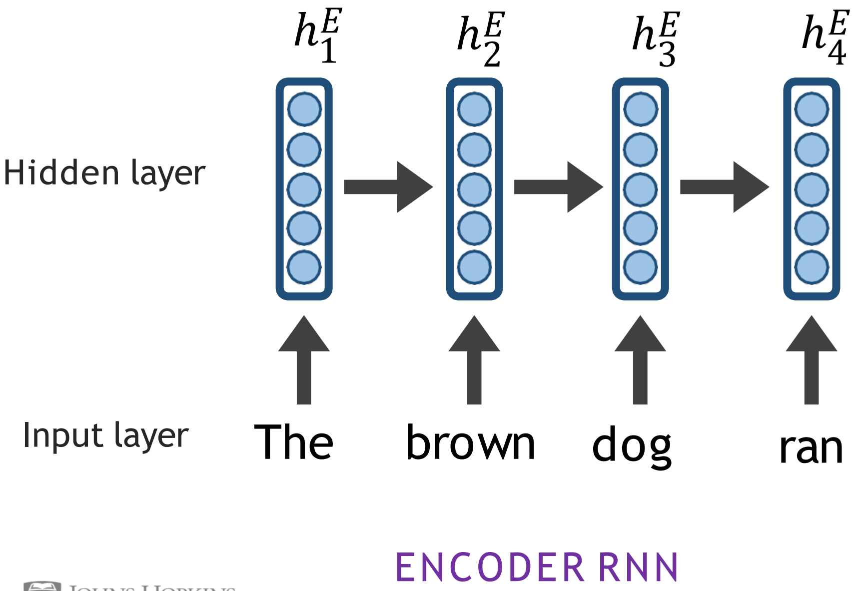
LMs w/ Recurrent Neural Nets



$$\underbrace{\mathrm{P}(X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1})}_{\text{context}}$$

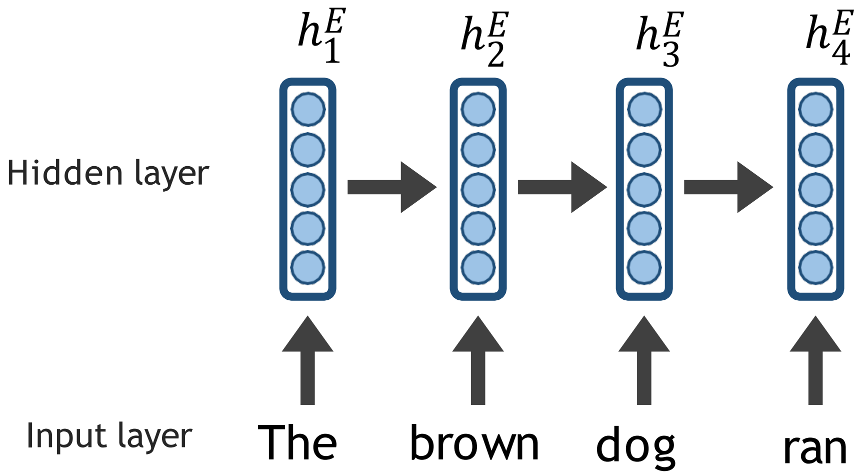
- We feed the words one at a time to the RNN.
- A predictive head uses the latest embedding vector to produce a probability over the vocabulary.

RNN to Map Sequence to Another Sequence (aka Seq2Seq)



RNN to Map Sequence to Another Sequence (aka Seq2Seq)

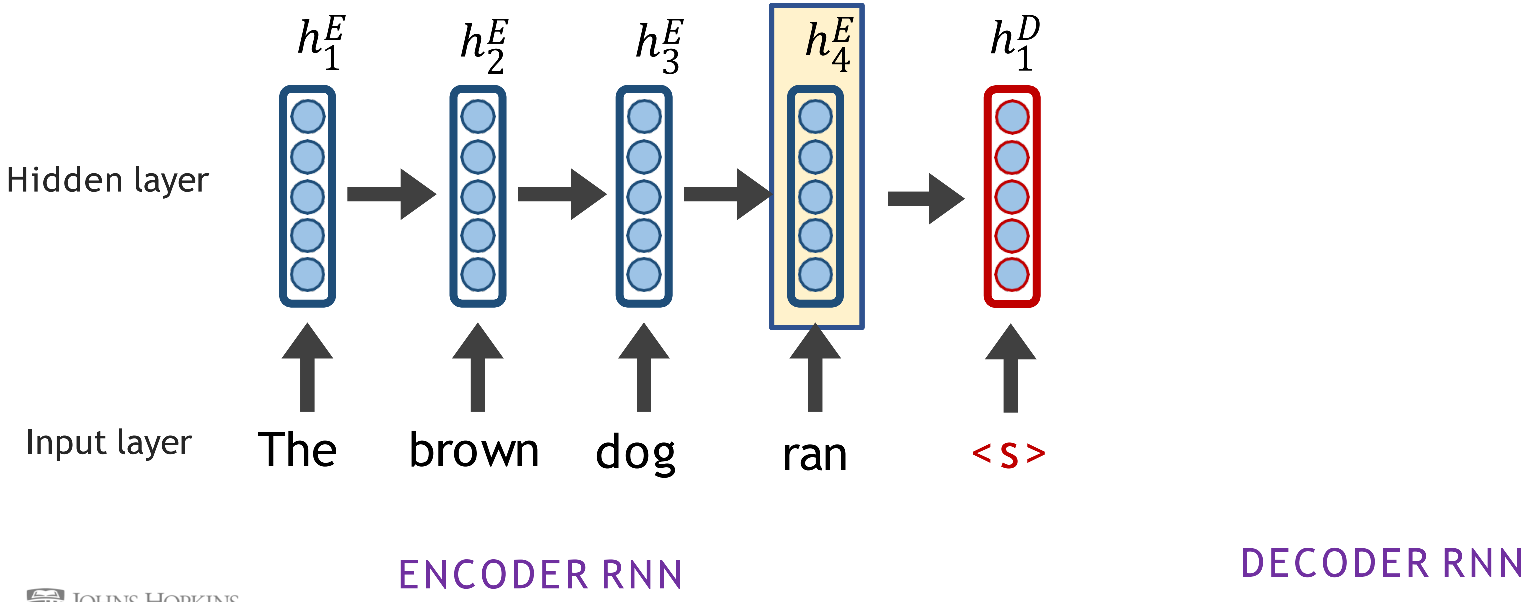
The final hidden state of the encoder RNN
is the initial state of the decoder RNN



ENCODER RNN

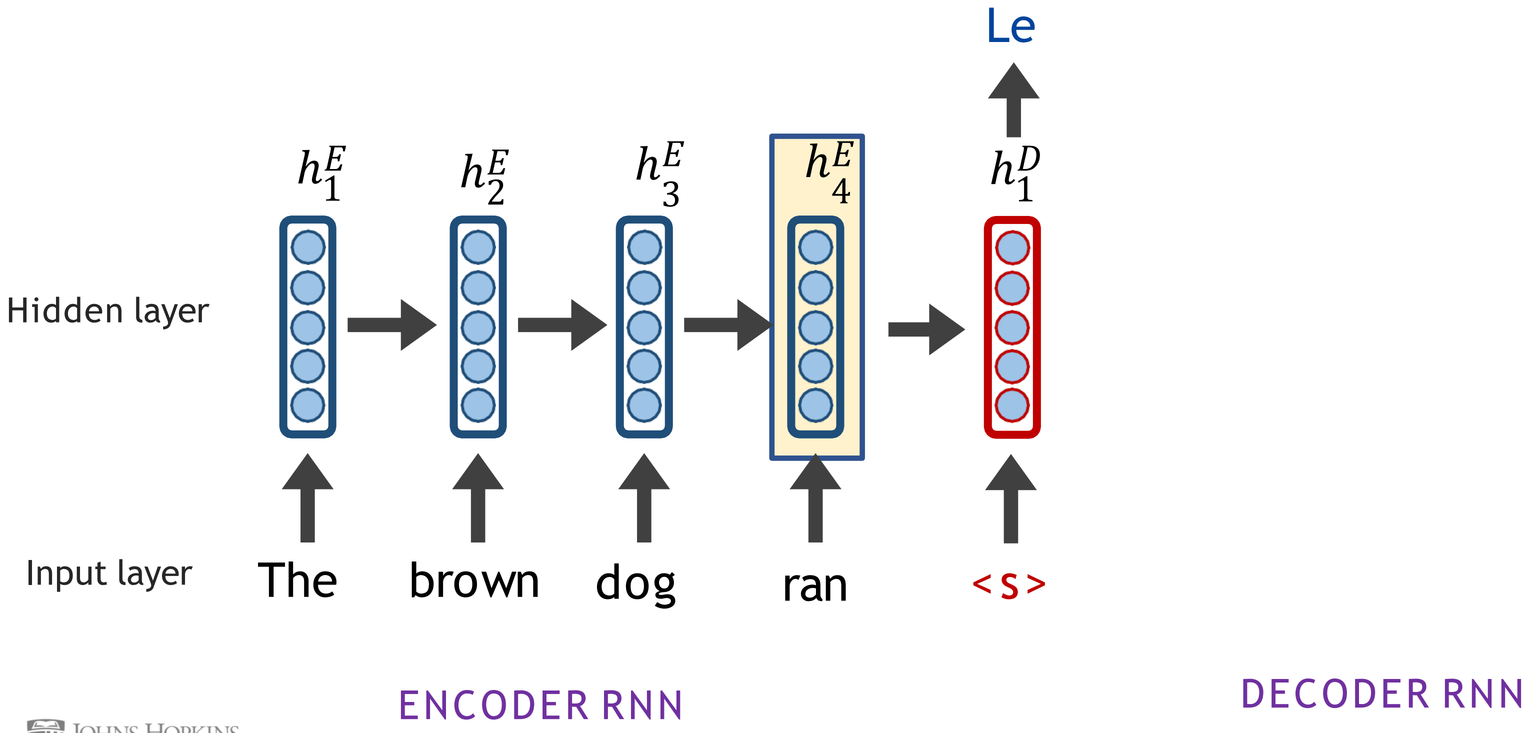
RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



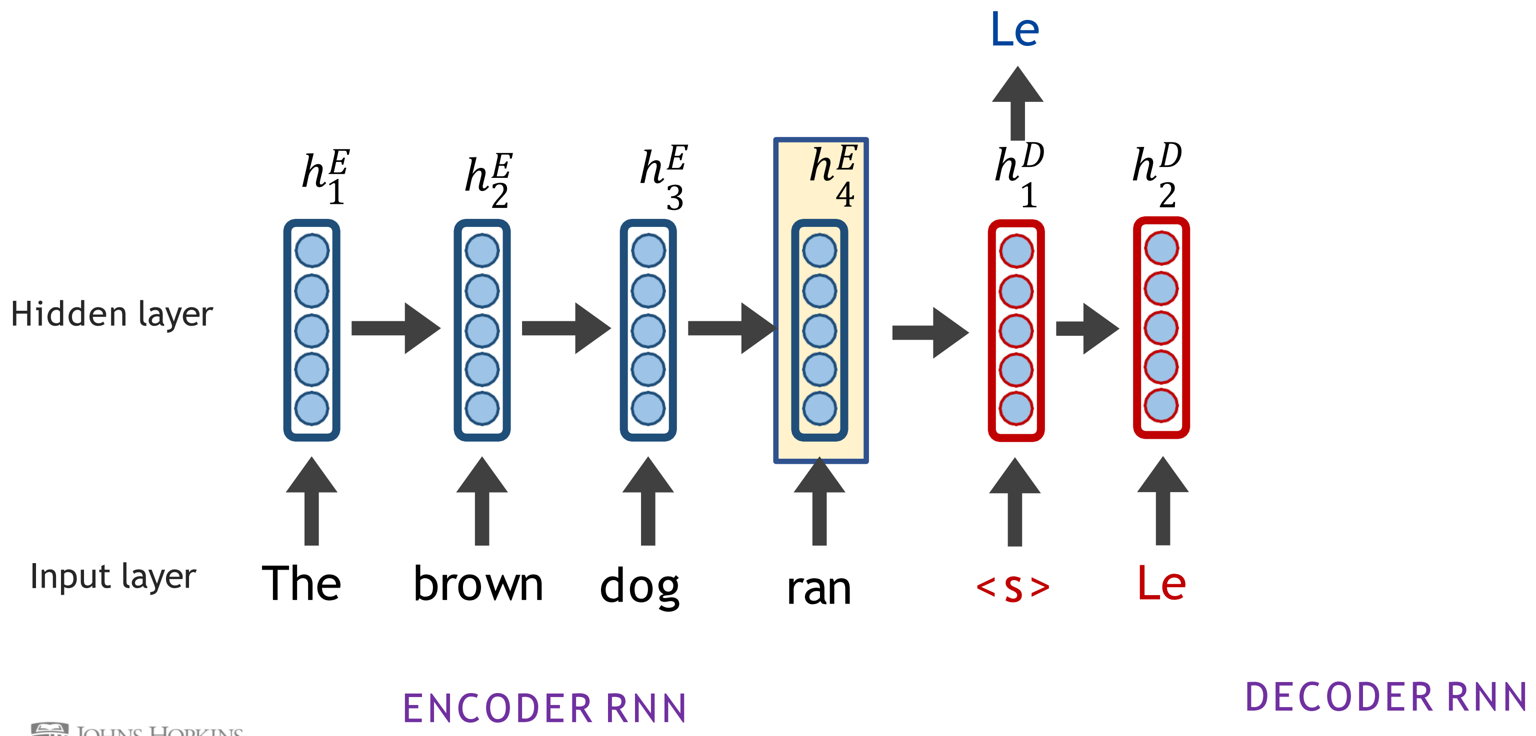
RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



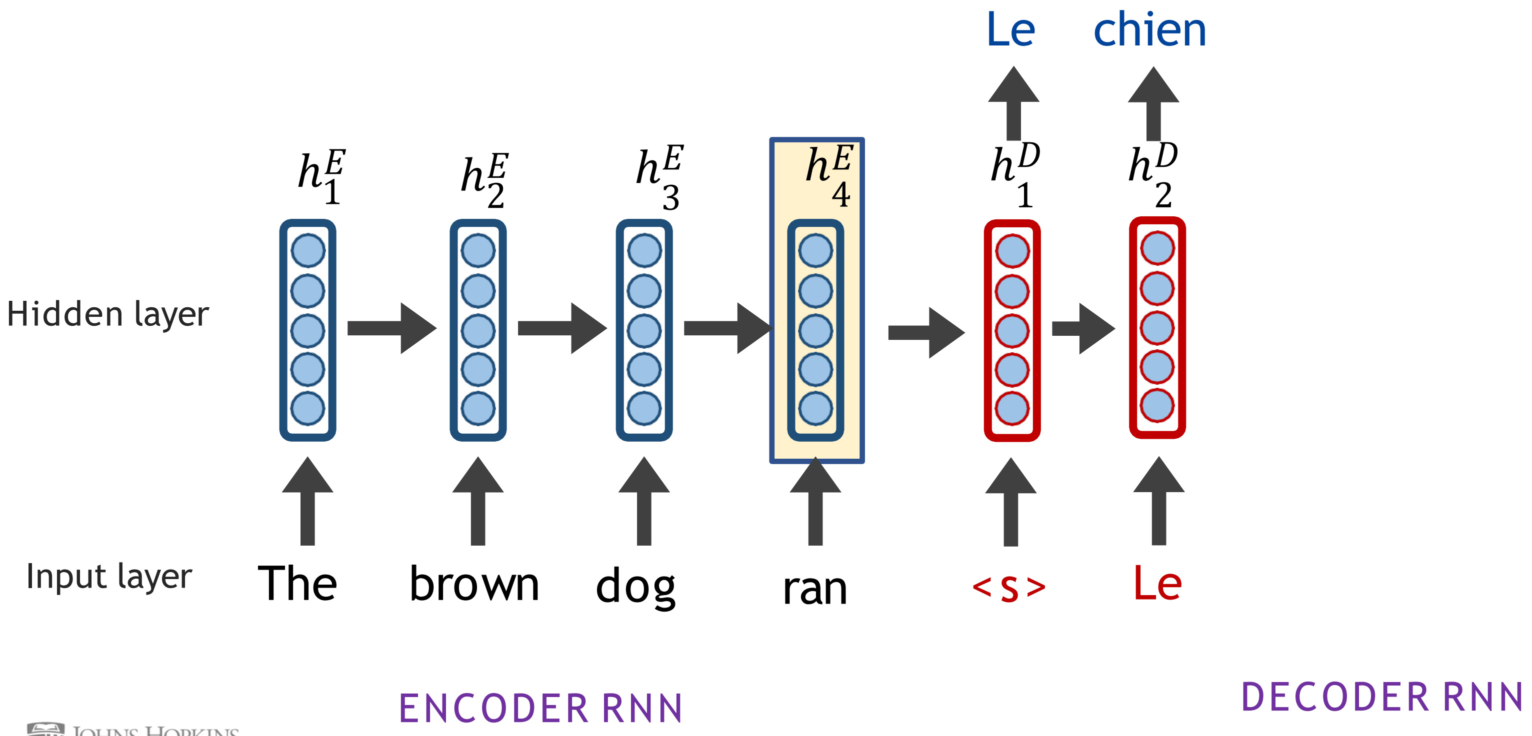
RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



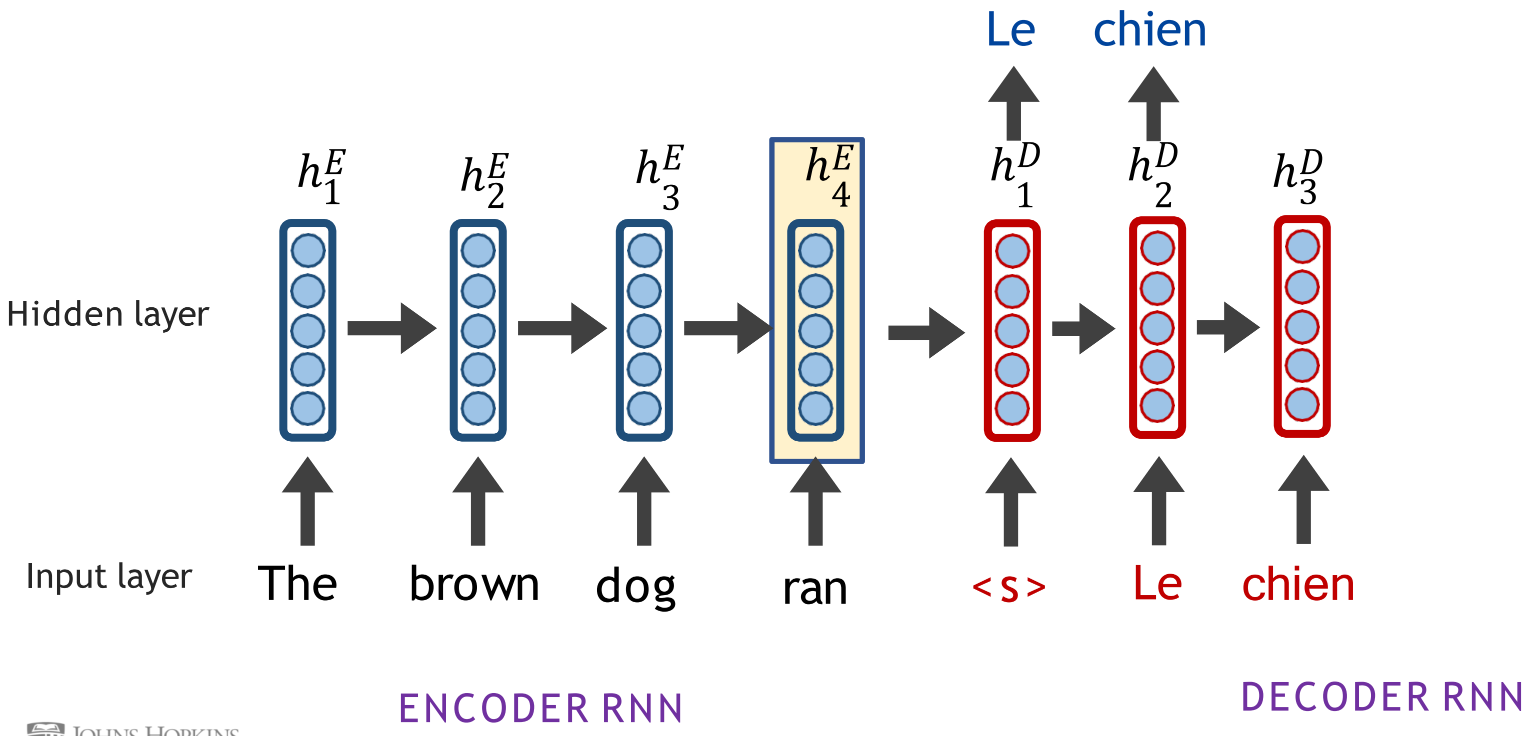
RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN is the initial state of the decoder RNN

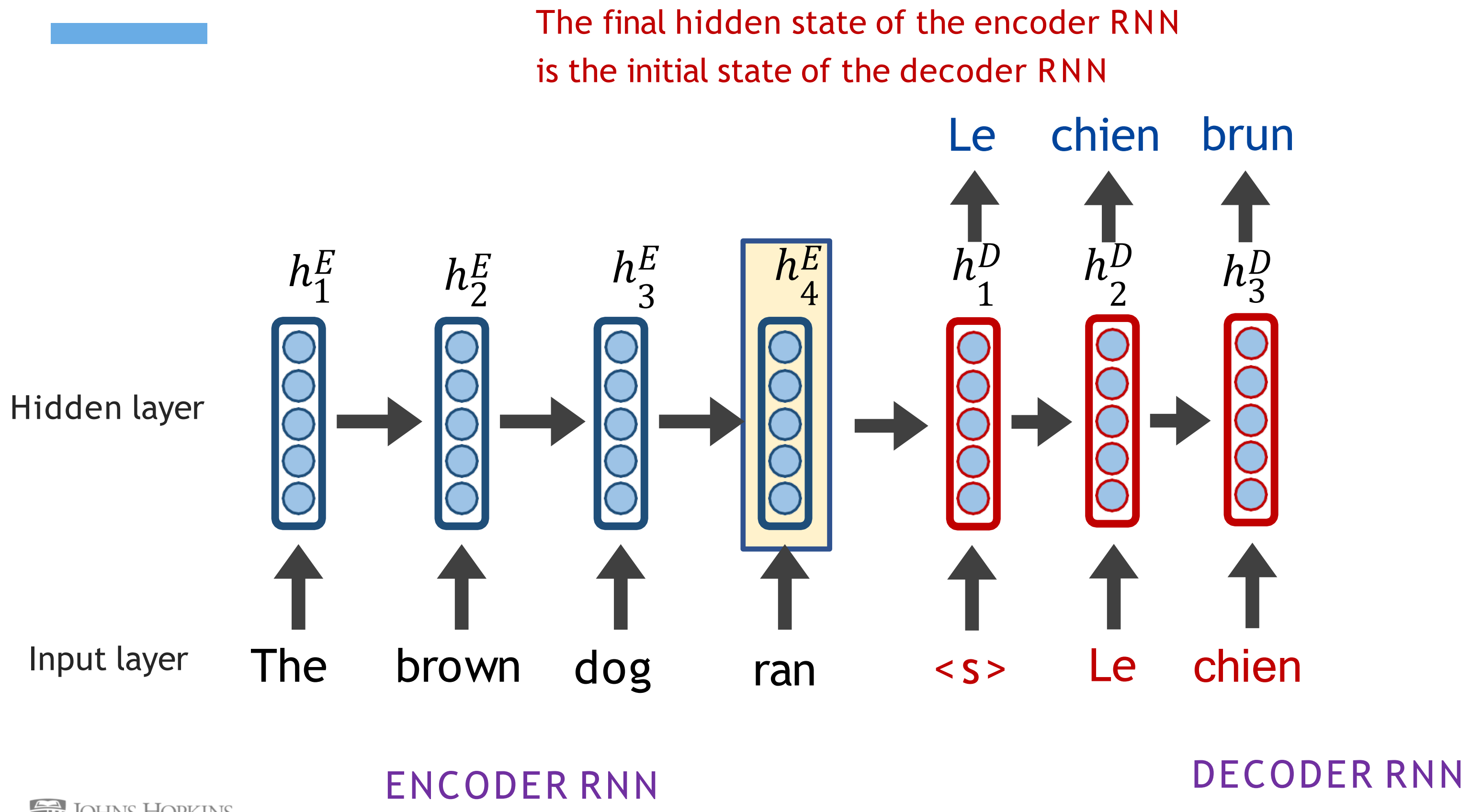


RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN

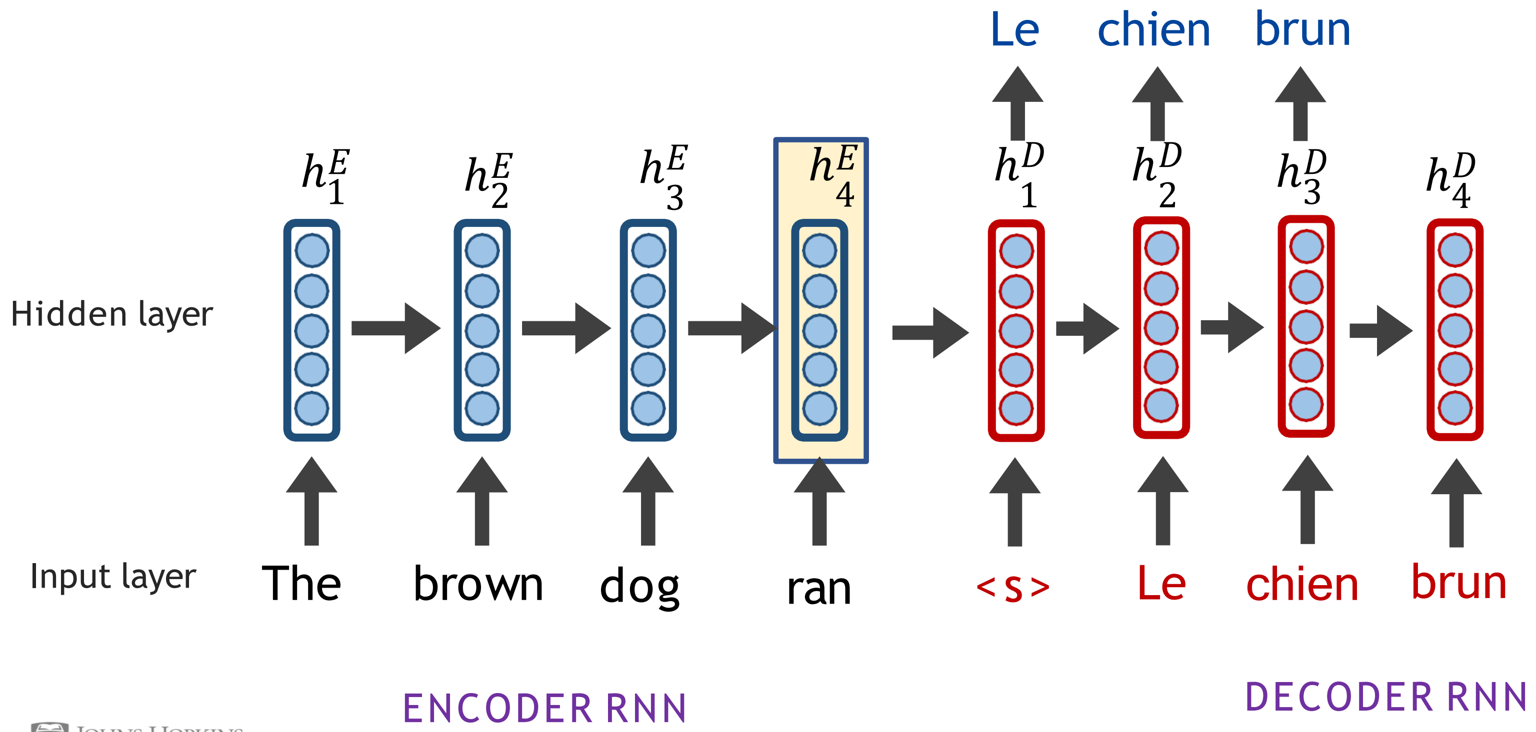


RNN to Map Sequence to Another Sequence (aka Seq2Seq)



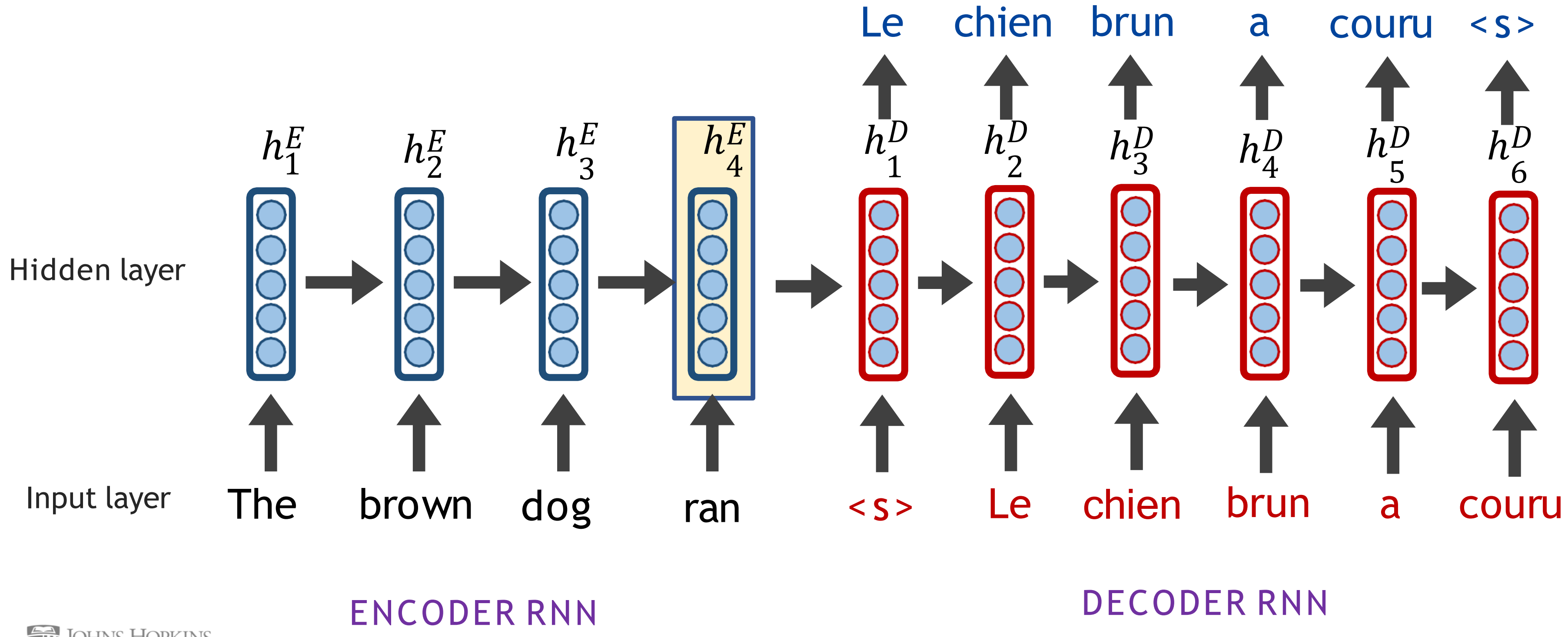
RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



RNN to Map Sequence to Another Sequence (aka Seq2Seq)

The final hidden state of the encoder RNN
is the initial state of the decoder RNN



RNN: Generation

- When trained on Harry Potter text, it generates:

"Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"

"No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

RNNs: Generation

- RNN-LM trained on Obama speeches:

```
The United States will step up to the cost of a new
challenges of the American people that will share the fact
that we created the problem. They were attacked and so that
they have to say that all the task of the final days of war
that I will not be able to get this done.
```

https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

35

RNNs in Practice

- RNN-LM trained on food recipes:

```
Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
      Yield: 6 Servings

      2 tb Parmesan cheese -- chopped
      1 c  Coconut milk
      3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and
simmer until firm. Serve hot in bodied fresh, mustard, orange and cheese. Combine the
cheese and salt together the dough in a large skillet; add the ingredients and stir
in the chocolate and pepper.
```

https://gist.github.com/nylki/1efbaa36635956d35bcc

Evaluation LMs with Perplexity (2016)

n-gram model →

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Source: https://engineering.fb.com/2016/10/25/ml-applications/building-an-efficient-neural-language-model-over-a-billion-words/

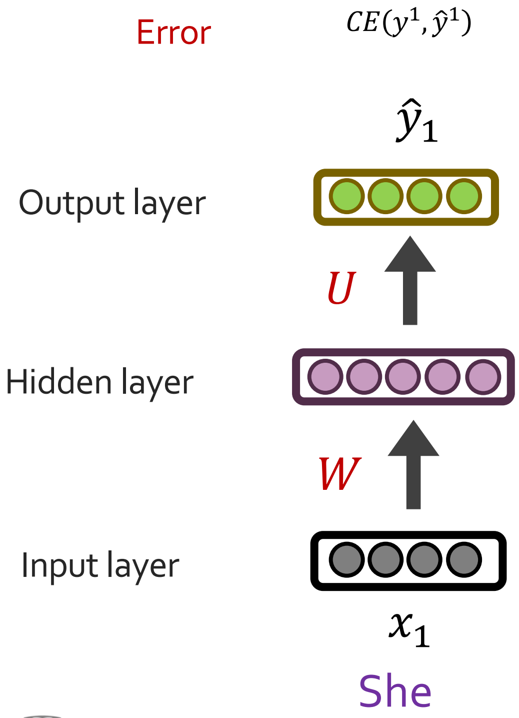
Summary

- RNNs: Repeated use of finite structure.
- A natural fit for language modeling.
- Next: how to train then.

Training RNNs

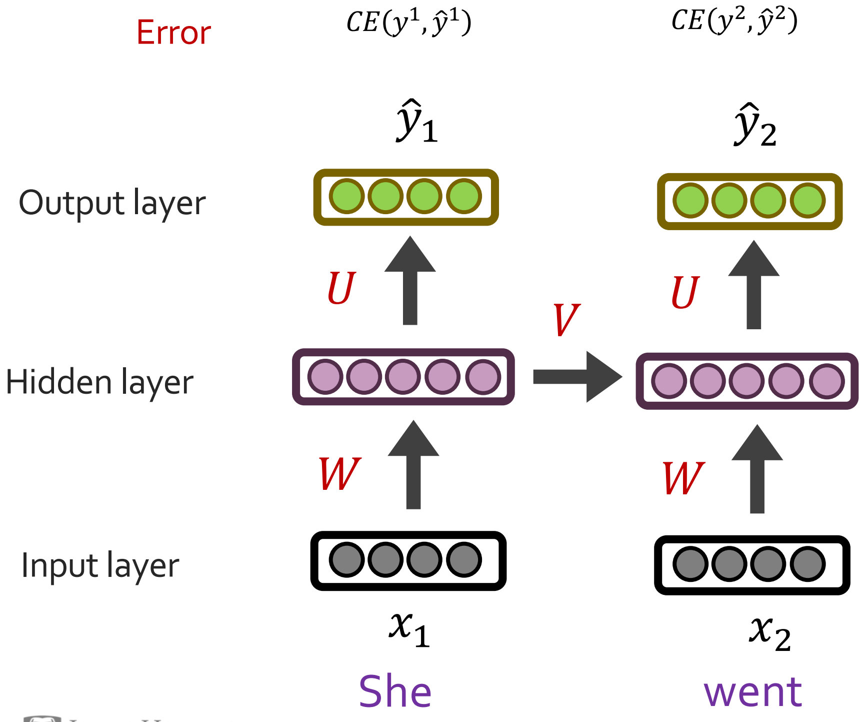
RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN: Forward Propagation

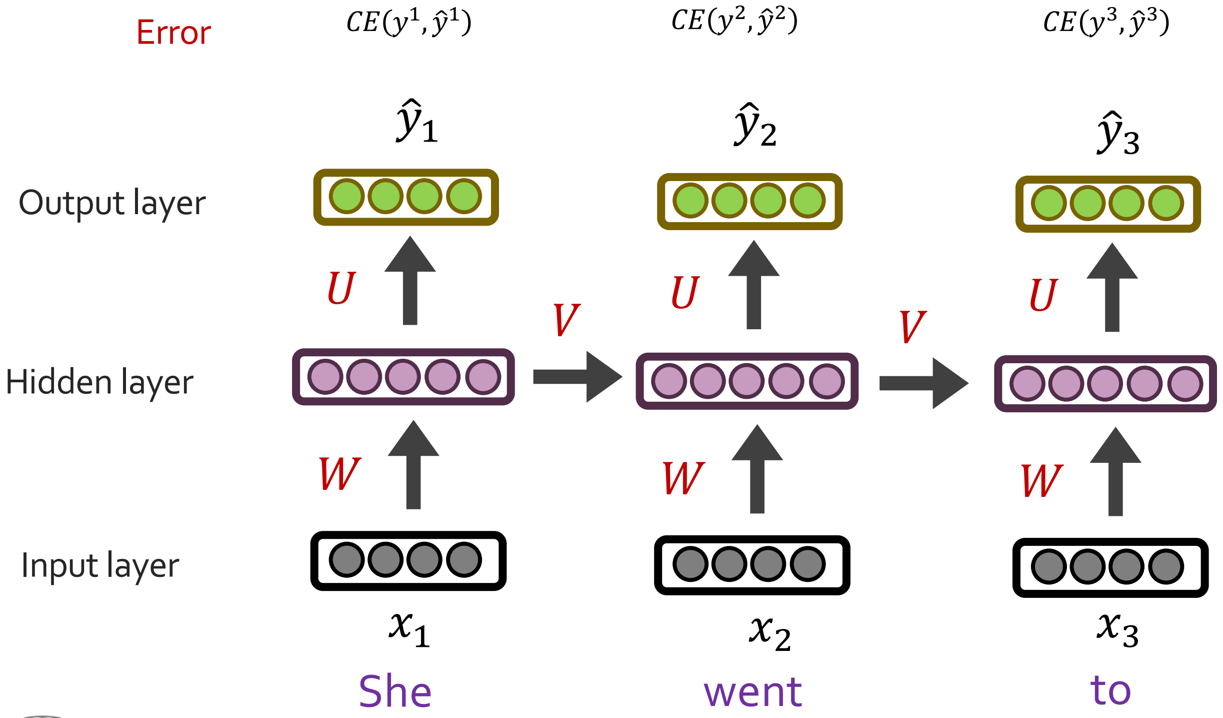
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



[Slide credit: Chris Tanner]

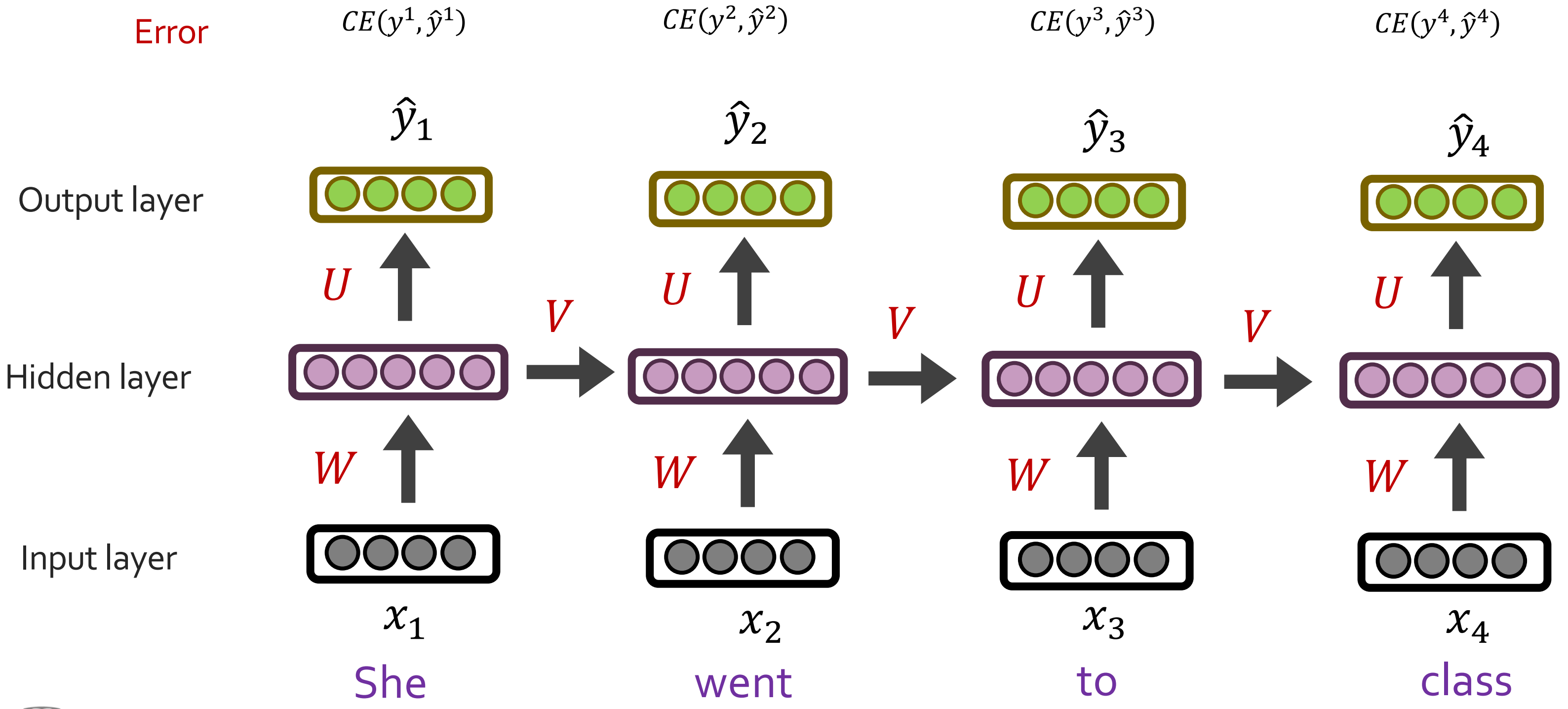
RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y^i_w \log(\hat{y}^i_w)$$



[Slide credit: Chris Tanner]

RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$

Error $CE(y^1, \hat{y}^1)$ $CE(y^2, \hat{y}^2)$ $CE(y^3, \hat{y}^3)$ $CE(y^4, \hat{y}^4)$

Output layer

During training, regardless of our output predictions, we feed in the correct inputs

Hidden layer

W W W W

Input layer

x_1 She

x_2 went

x_3 to

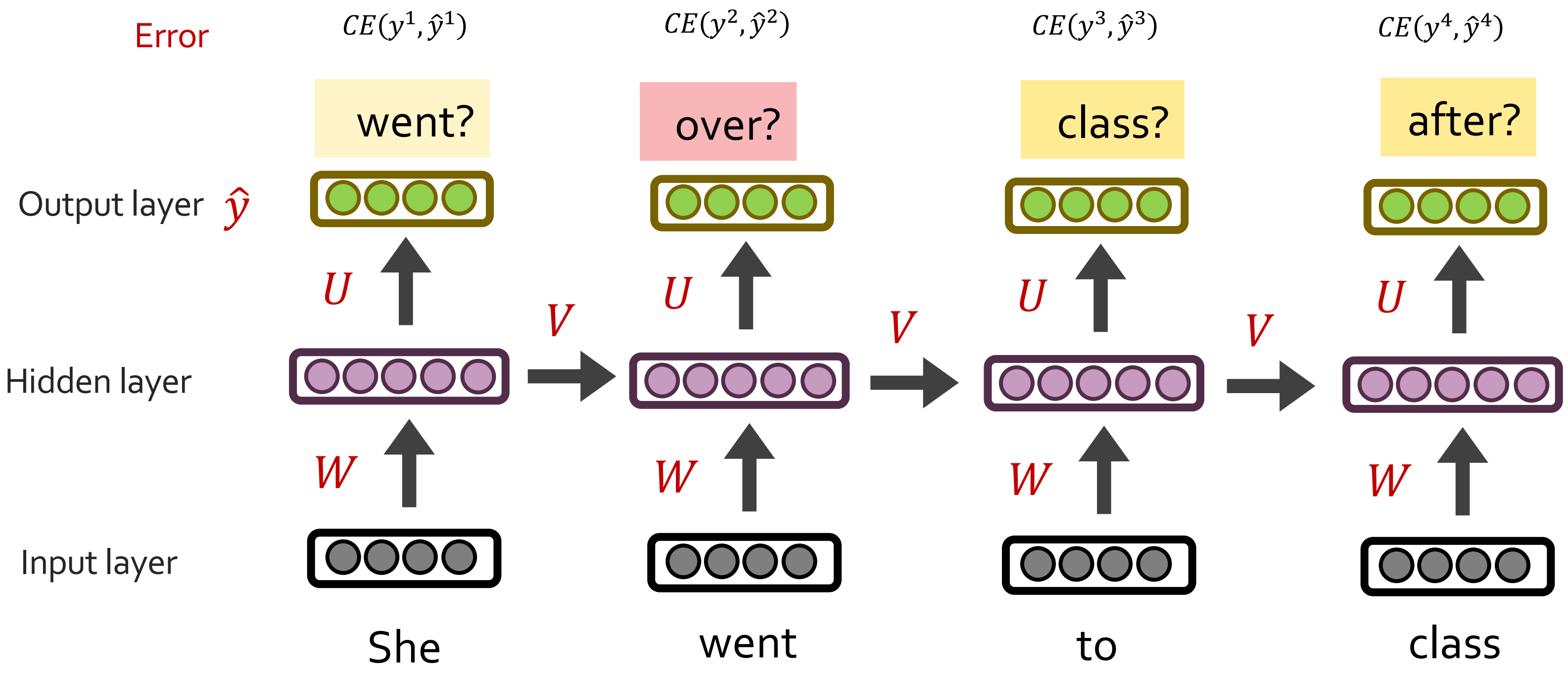
x_4 class

Johns Hopkins Whiting School of Engineering

[Slide credit: Chris Tanner]

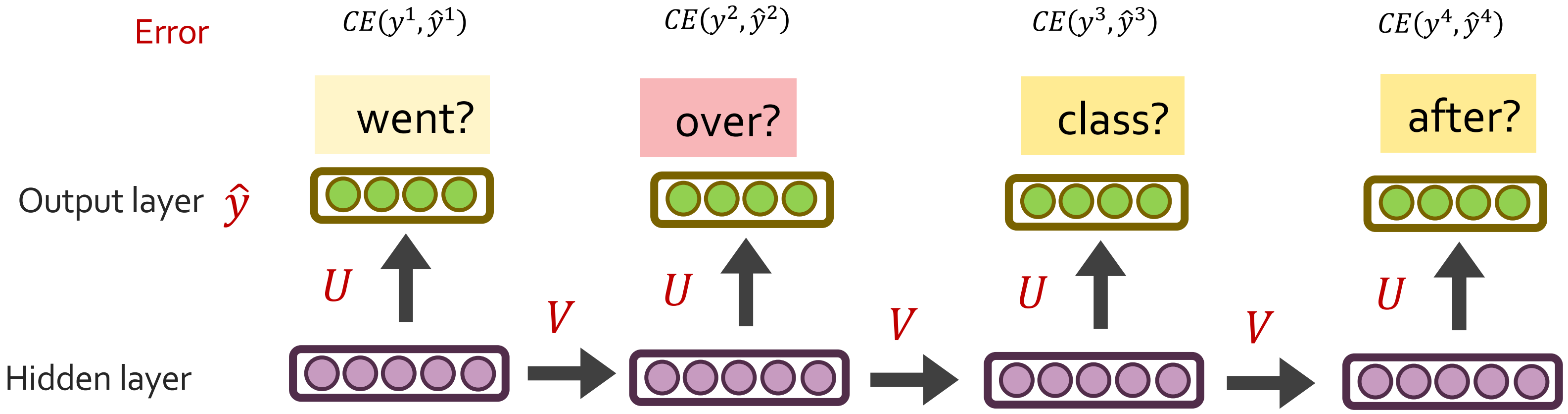
RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN: Forward Propagation

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



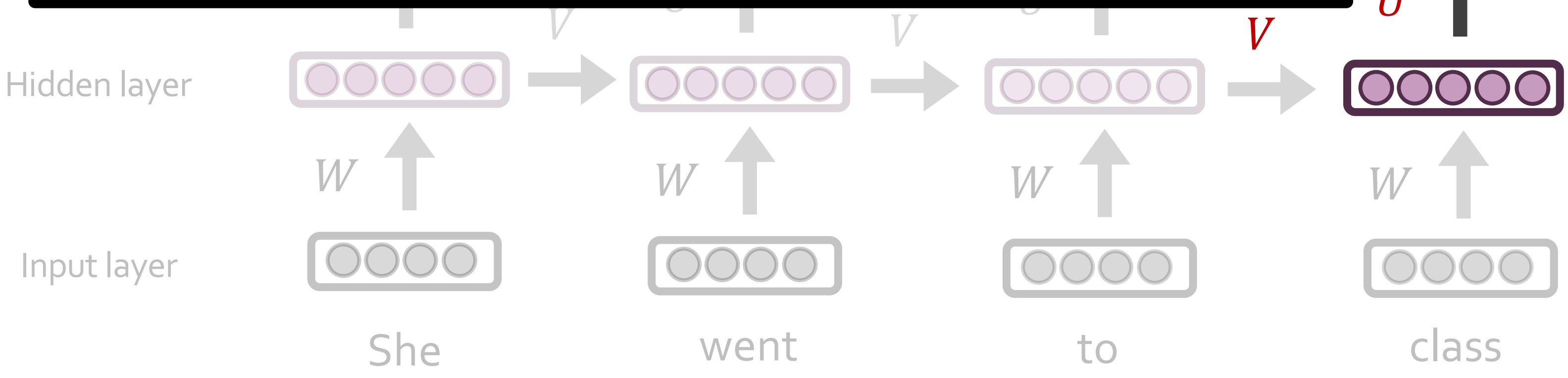
Our total loss is simply the average loss across all T time steps

Backward Step

$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$

To update our weights (e.g. Θ), we calculate the gradient of our loss w.r.t. the repeated weight matrix (e.g., $\frac{\partial L}{\partial \Theta}$).

Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.

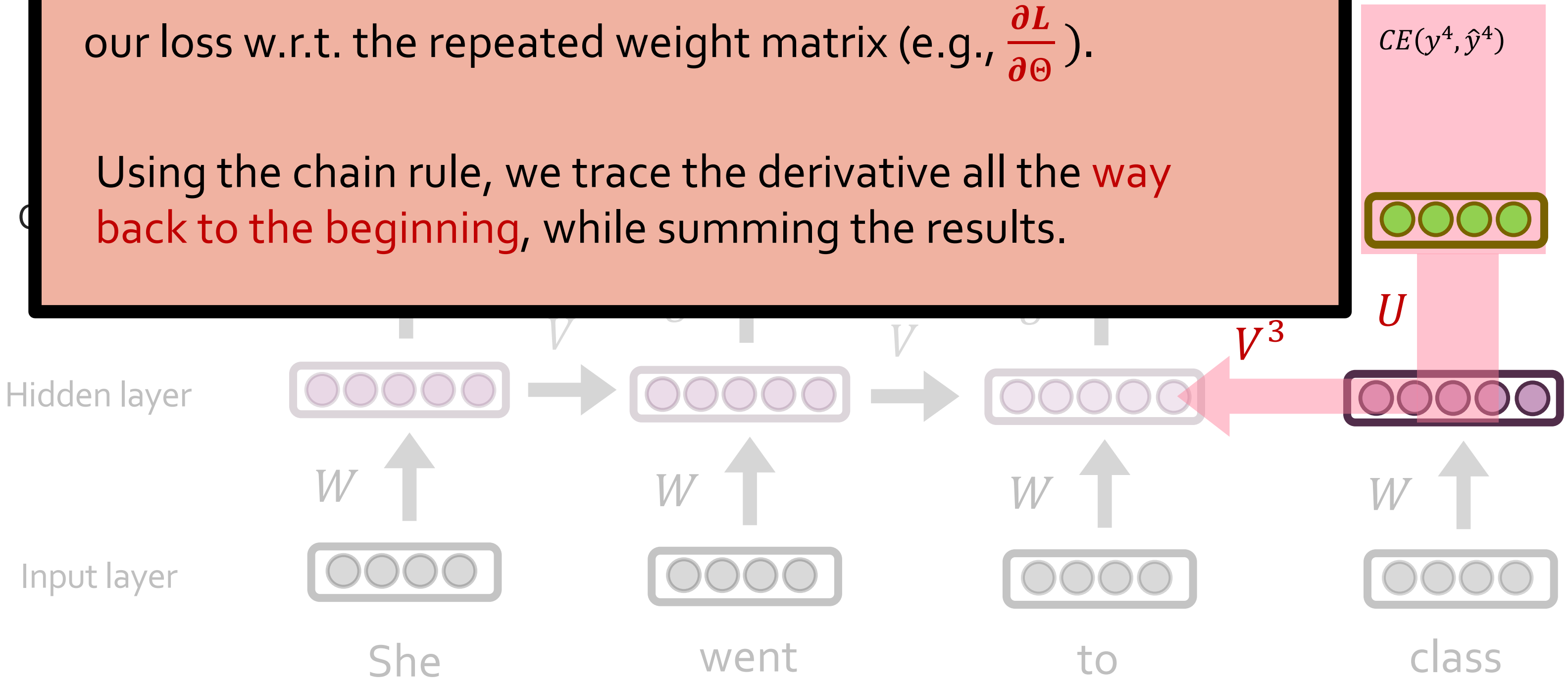


Backward Step

$$\frac{\partial L}{\partial V}$$

To update our weights (e.g. Θ), we calculate the gradient of our loss w.r.t. the repeated weight matrix (e.g., $\frac{\partial L}{\partial \Theta}$).

Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.

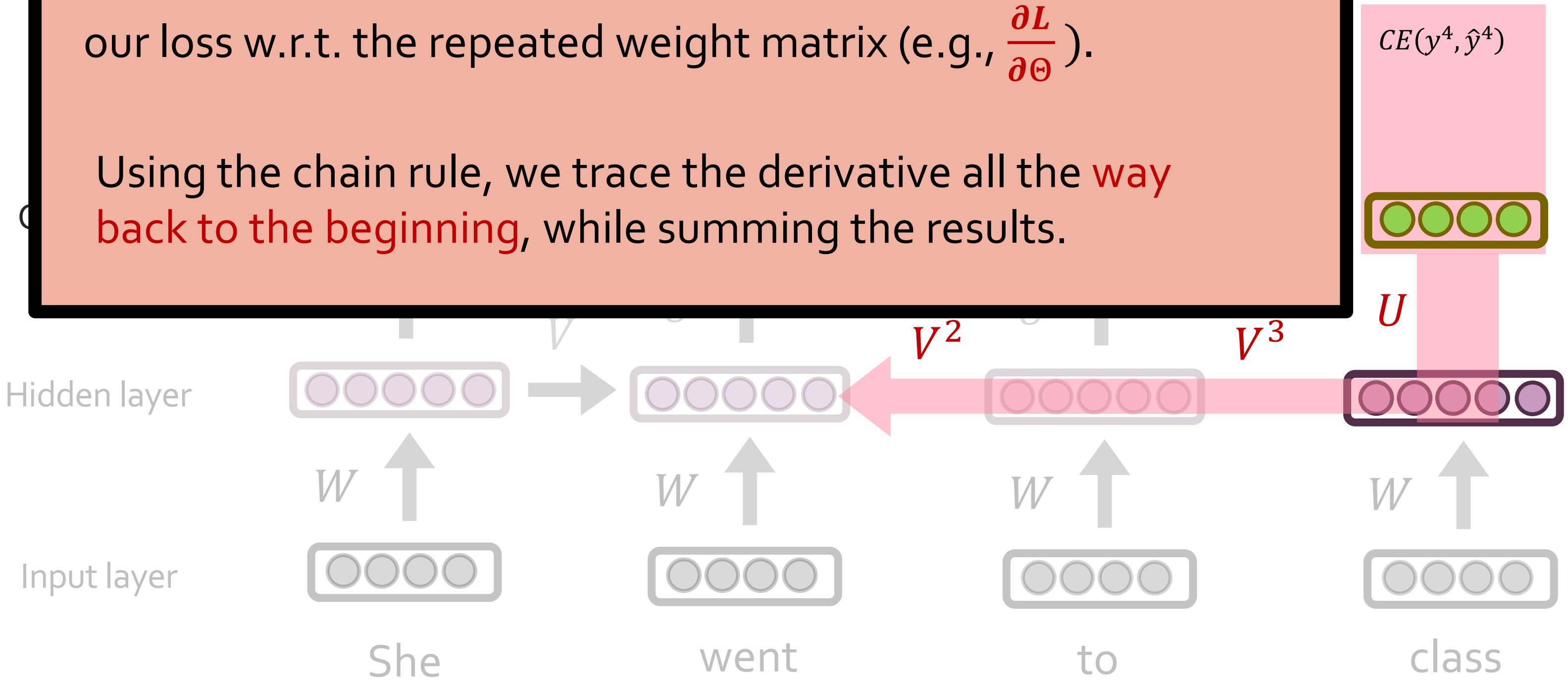


Backward Step

$$\frac{\partial L}{\partial V}$$

To update our weights (e.g. Θ), we calculate the gradient of our loss w.r.t. the repeated weight matrix (e.g., $\frac{\partial L}{\partial \Theta}$).

Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.

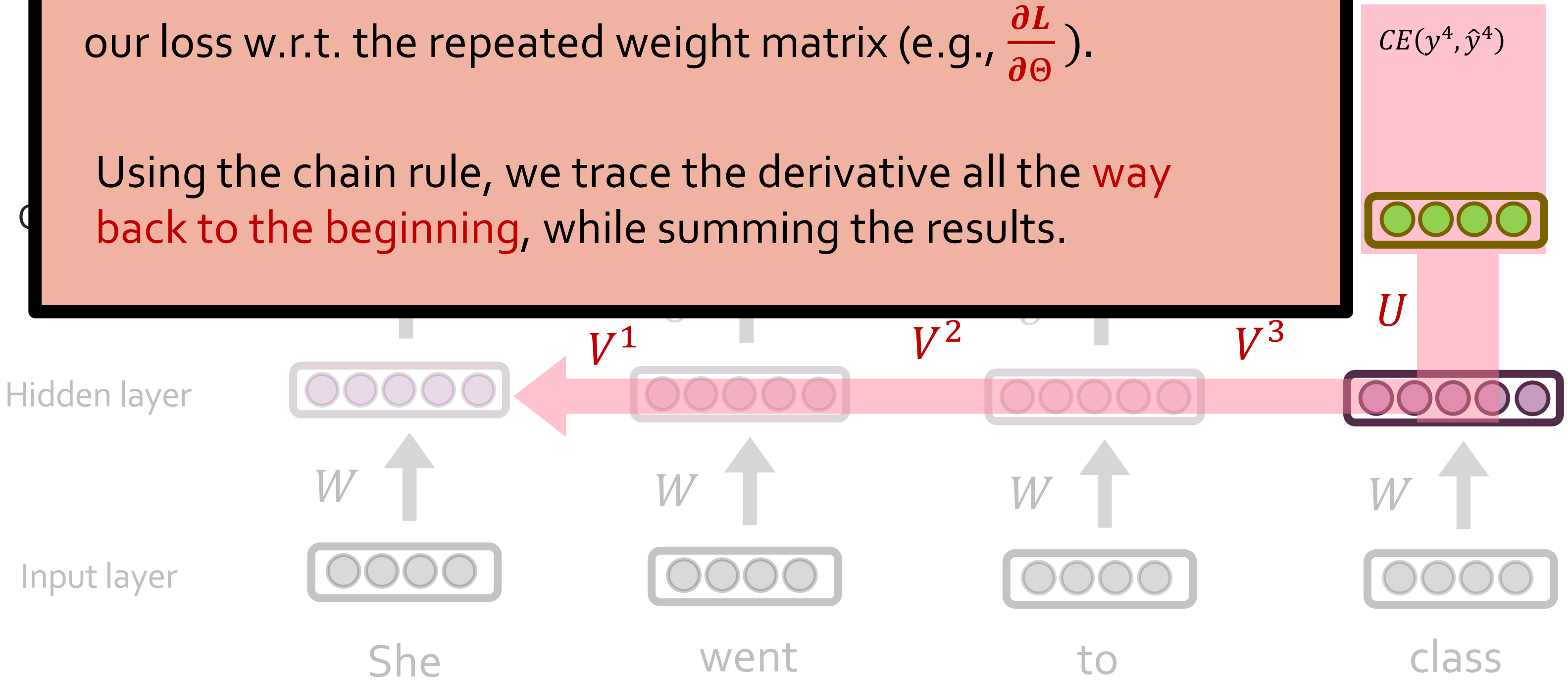


Backward Step

To update our weights (e.g. Θ), we calculate the gradient of our loss w.r.t. the repeated weight matrix (e.g., $\frac{\partial L}{\partial \Theta}$).

Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.

$$\frac{\partial L}{\partial V}$$



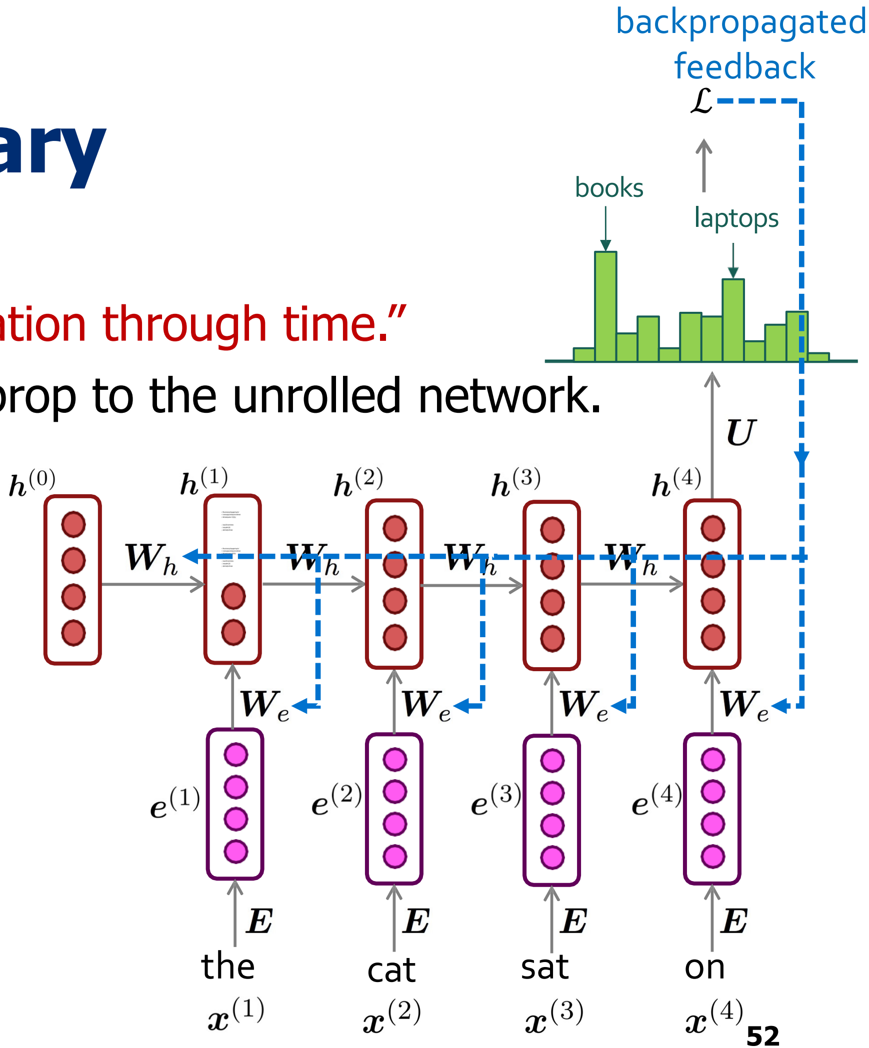
[Slide credit: Chris Tanner]

Training RNNs: Summary

- RNNs can be trained using "backpropagation through time."
- Can be viewed as applying normal backprop to the unrolled network.

- Model's learnable parameters Θ

1. Compute $\mathcal{L}(\Theta)$ for a batch of sentences
2. Compute gradients $\nabla_\Theta \mathcal{L}(\Theta)$
3. Update the weights and then repeat

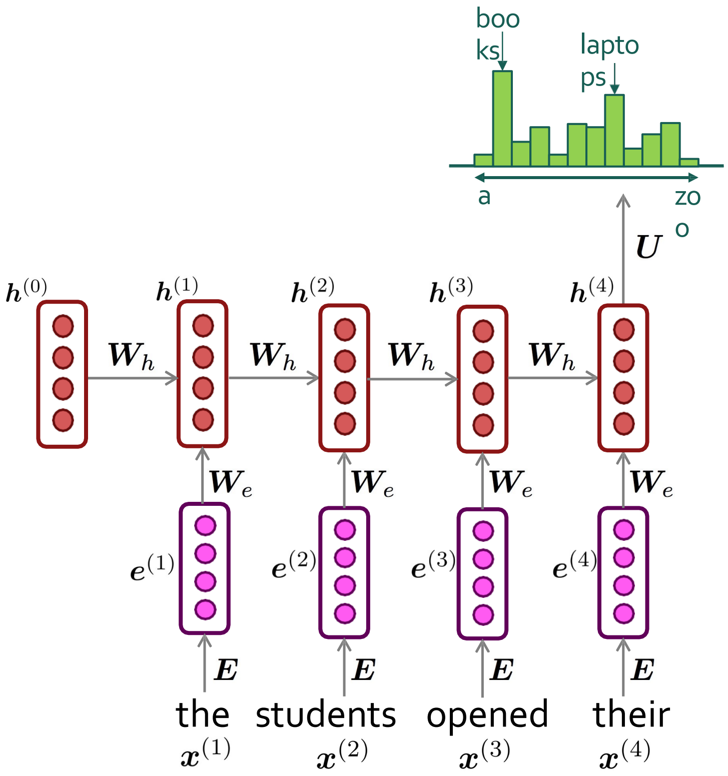


RNN-LMs:
Pros and Cons

RNNs: Advantages

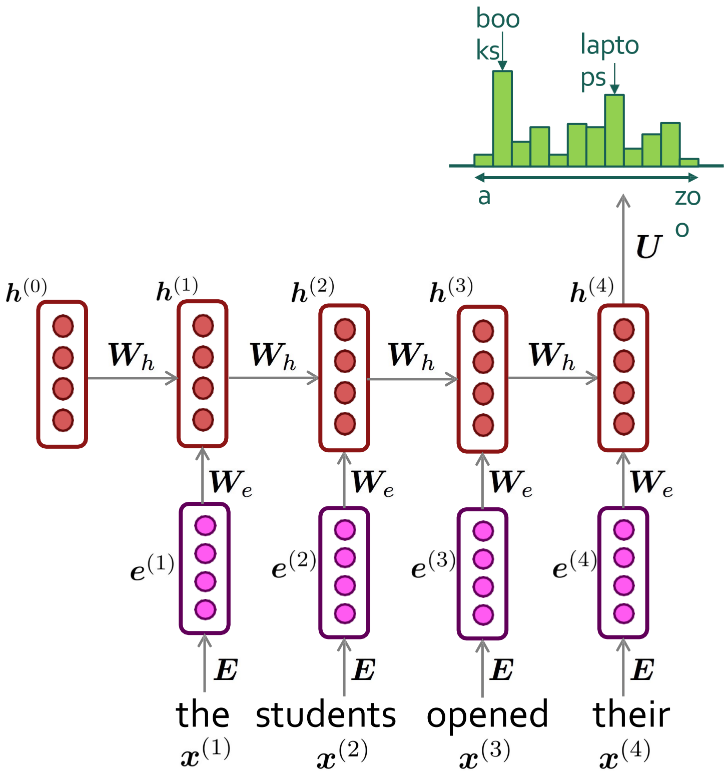
- Model size doesn't increase for longer inputs — reusing a compact set of model parameters.

- Computation for step t can (in theory) use information from many steps back



RNNs: Weaknesses

- Recurrent computation is slow and difficult to parallelize.
  - Next week: self-attention mechanism, better at representing long sequences and also parallelizable.
- While RNNs in theory can represent long sequences, they quickly forget portions of the input.
- Vanishing/exploding gradients.



Vanishing/Exploding Gradient Problem: Intuition

- Backpropagated errors multiply at each layer, resulting in

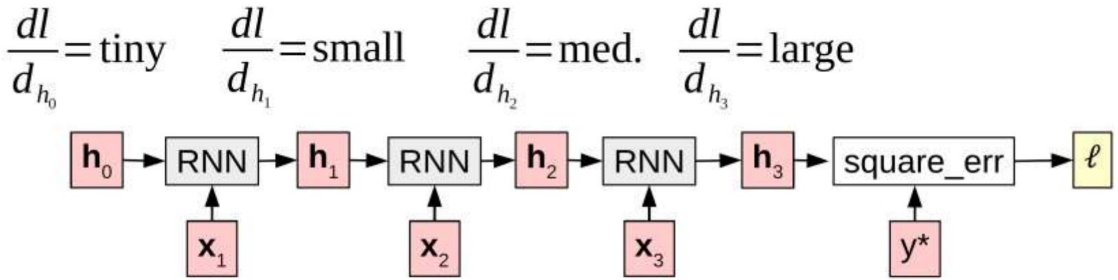


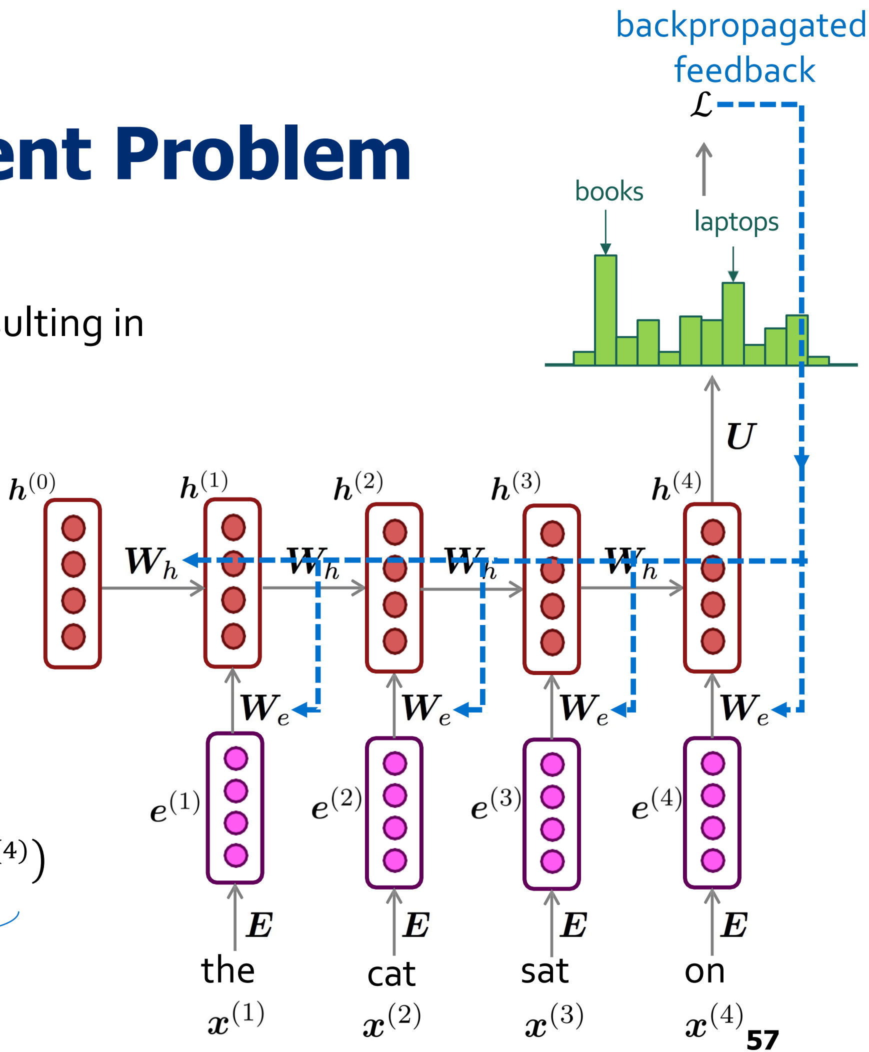
Figure from Graham Neubig

Vanishing/Exploding Gradient Problem

- Backpropagated errors multiply at each layer, resulting in exponential decay (if derivative is small) or growth (if derivative is large).
- Makes it very difficult train deep networks, or simple recurrent networks over many time steps.

$$\nabla_{\mathcal{L}}(\mathbf{W}_h) = \left(\mathbf{J}_{\mathcal{L}}(\mathbf{W}_{L-1})\right)^{\mathrm{T}} = \sum_{t=0} \left(\mathbf{J}_{\mathcal{L}}(\boldsymbol{h}^{(t)})\, \mathbf{J}_{\boldsymbol{h}^{(t)}}(\mathbf{W}_h)\right)^{\mathrm{T}}$$

$$\mathbf{J}_{\mathcal{L}}(\boldsymbol{h}^{(0)}) = \underbrace{\mathbf{J}_{\boldsymbol{h}^{(1)}}(\boldsymbol{h}^{(0)})\mathbf{J}_{\boldsymbol{h}^{(2)}}(\boldsymbol{h}^{(1)}) \times \ldots \times \mathbf{J}_{\boldsymbol{h}^{(4)}}(\boldsymbol{h}^{(3)})\, \mathbf{J}_{\mathcal{L}}(\boldsymbol{h}^{(4)})}_{\text{chain rule}}$$

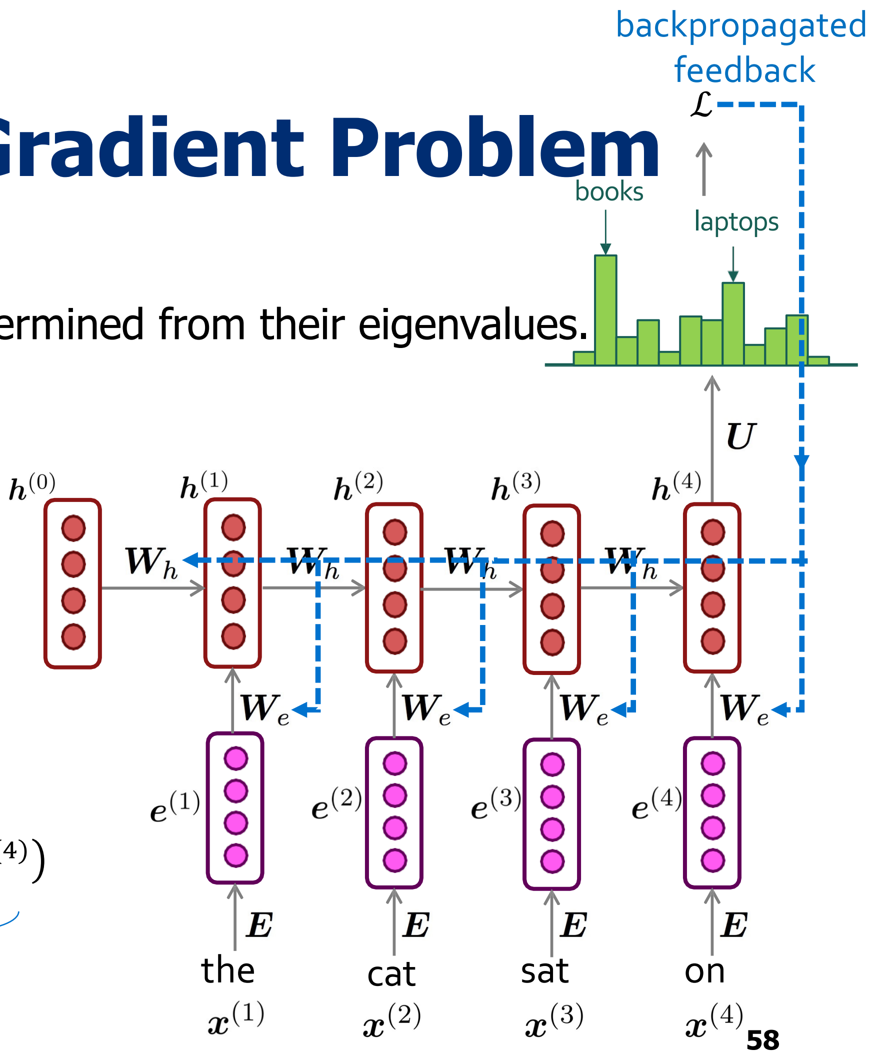


Vanishing/Exploding Gradient Problem

- **Note:** instability of matrix powers can be determined from their eigenvalues.

Gradient signal from far away is lost. So, model weights are updated only with respect to near effects, not long-term effects.

$$\mathbf{J}_{\mathcal{L}}\left(\boldsymbol{h}^{(0)}\right) = \underbrace{\mathbf{J}_{\boldsymbol{h}^{(1)}}\left(\boldsymbol{h}^{(0)}\right)\mathbf{J}_{\boldsymbol{h}^{(2)}}\left(\boldsymbol{h}^{(1)}\right) \times \ldots \times \mathbf{J}_{\boldsymbol{h}^{(4)}}\left(\boldsymbol{h}^{(3)}\right)}_{\text{chain rule}} \mathbf{J}_{\mathcal{L}}\left(\boldsymbol{h}^{(4)}\right)$$



RNNs: Difficulty in Learning Long-Range Dependencies

- While RNNs in theory can represent long sequences, in practice teaching them about long-range dependencies is non-trivial.
- **Gradient clipping:**
  - If the norm of the gradient is greater than some threshold, scale it down before applying SGD update.
  - **Intuition:** take a step in the same direction, but a smaller step

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Johns Hopkins Whiting School of Engineering

[“On the difficulty of training recurrent neural networks”, Pascanu et al, 2013]

RNNs: Difficulty in Learning Long-Range Dependencies (2)

- While RNNs in theory can represent long sequences, in practice teaching them about long-range dependencies is non-trivial.
- **Using residual layers:**
  - lots of new deep architectures (RNN or otherwise) add direct connections, thus allowing the gradient to flow)

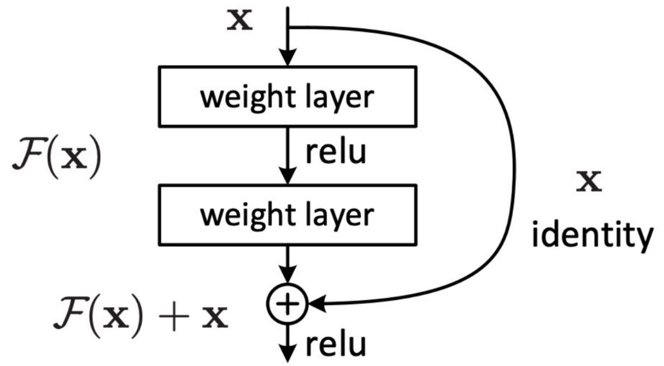


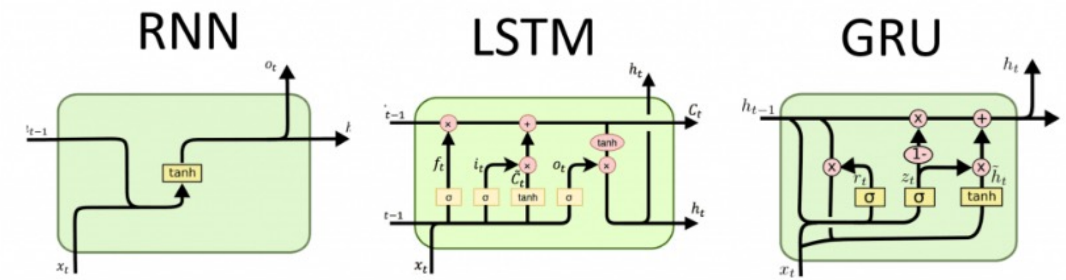
Figure 2. Residual learning: a building block.

"Deep Residual Learning for Image Recognition", He et al, 2015. https://arxiv.org/pdf/1512.03385.pdf

60

RNNs: Difficulty in Learning Long-Range Dependencies (3)

- While RNNs in theory can represent long sequences, in practice teaching them about long-range dependencies is non-trivial.
- Changes to the architecture makes it easier for the RNN to preserve information over many timesteps
  - Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997, Gers+ 2000]
  - Gated Recurrent Units (GRU) [Cho+ 2014]

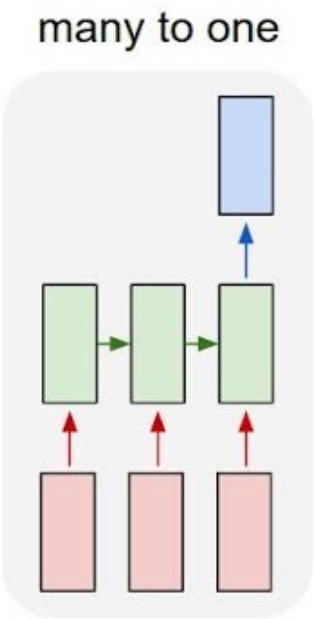


61

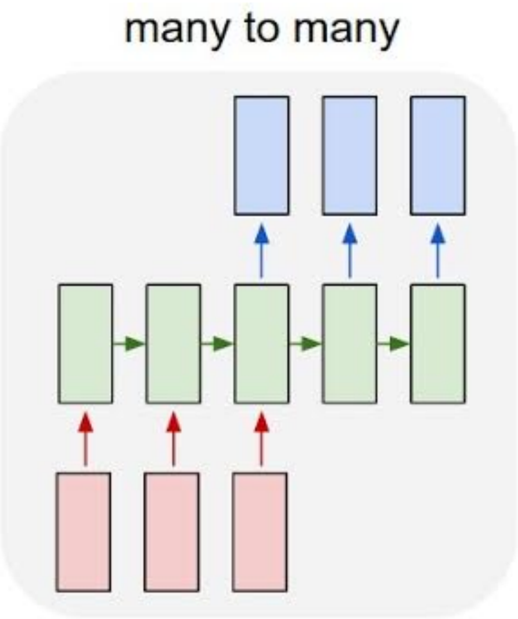
RNNs: Difficulty in Learning Long-Range Dependencies (3)

- While RNNs in theory can represent long sequences, in practice teaching them about long-range dependencies is non-trivial.
- Changes to the architecture makes it easier for the RNN to preserve information over many timesteps
  - Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997, Gers+ 2000]
  - Gated Recurrent Units (GRU) [Cho+ 2014]
- Many of these variants were the dominant architecture of In 2013–2015.
- We will not cover these alternative architecture in favor or spending more time on more modern developments.

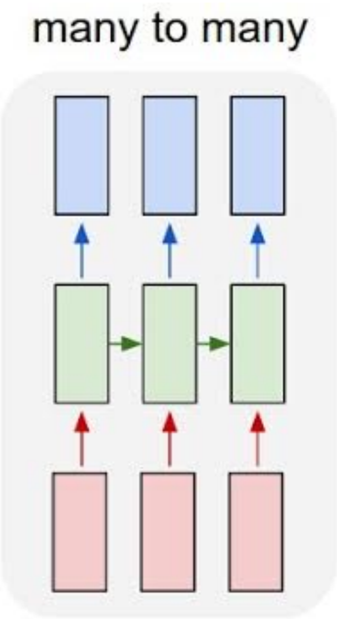
Adapting RNNs to Application



Text Classification

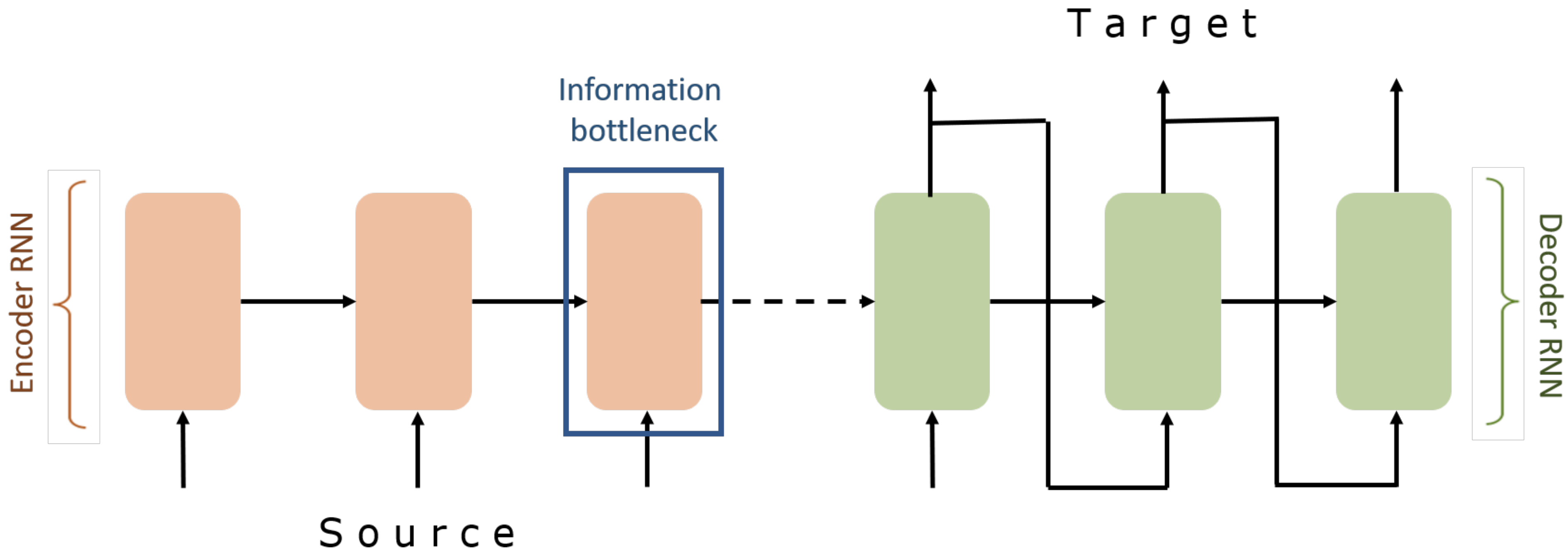


Language Modeling



POS Tags

Encoder-Decoder Architectures

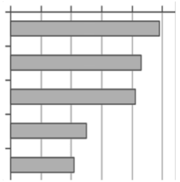


Encoder-Decoder Architectures

- It is useful to think of generative models as two sub-models.

“The cat sat on the [MASK]”

Some model



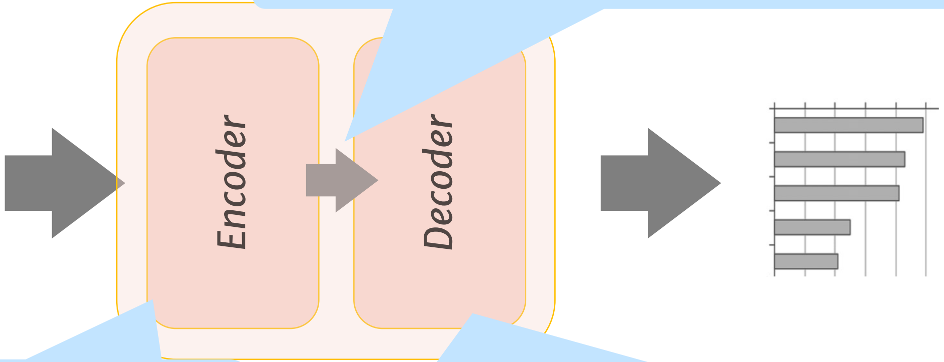
65

Encoder-Decoder Architectures

- It is useful to think of generative models as two sub-models

Representation (compression) of the context

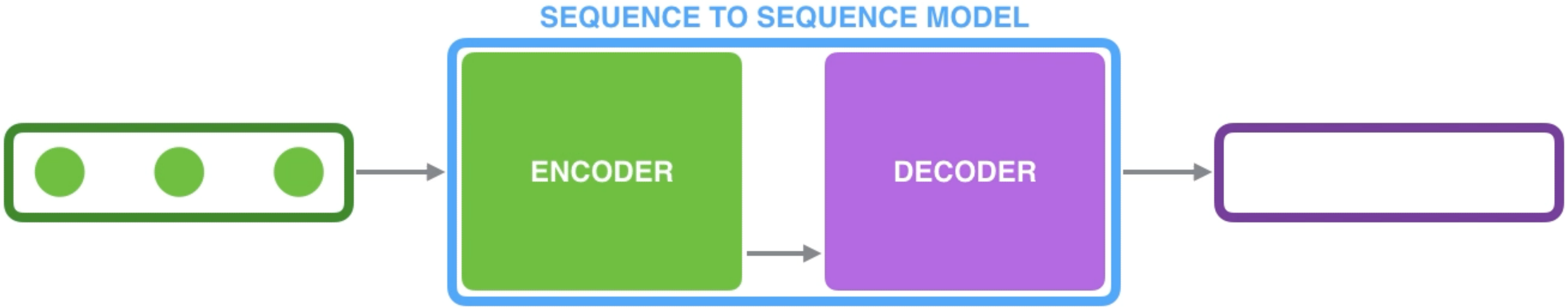
“The cat sat on the [MASK]”



Processes the context and compiles it into a vector.

Produces the output sequence item by item using the representation of the context.

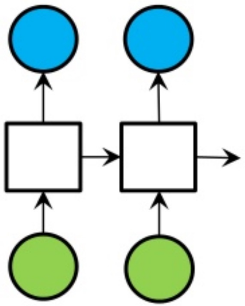
Encoder-Decoder Architectures



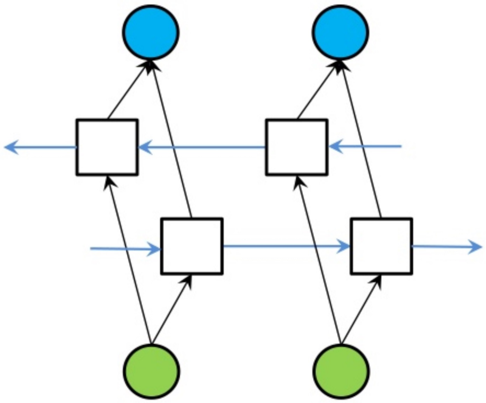
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Extending RNNs to Both Directions

- An RNN limitation: Hidden variables capture only one side of the context.
- Solution: Bi-Directional RNNs



RNN



Bi-directional RNN

Summary

- RNNs provide a compact model, regardless of sequence size. In theory, this is great!

- In practice, however:
  - They still struggle with remembering long-range dependencies.
  - Training them is not difficult because of vanishing/exploding gradients.

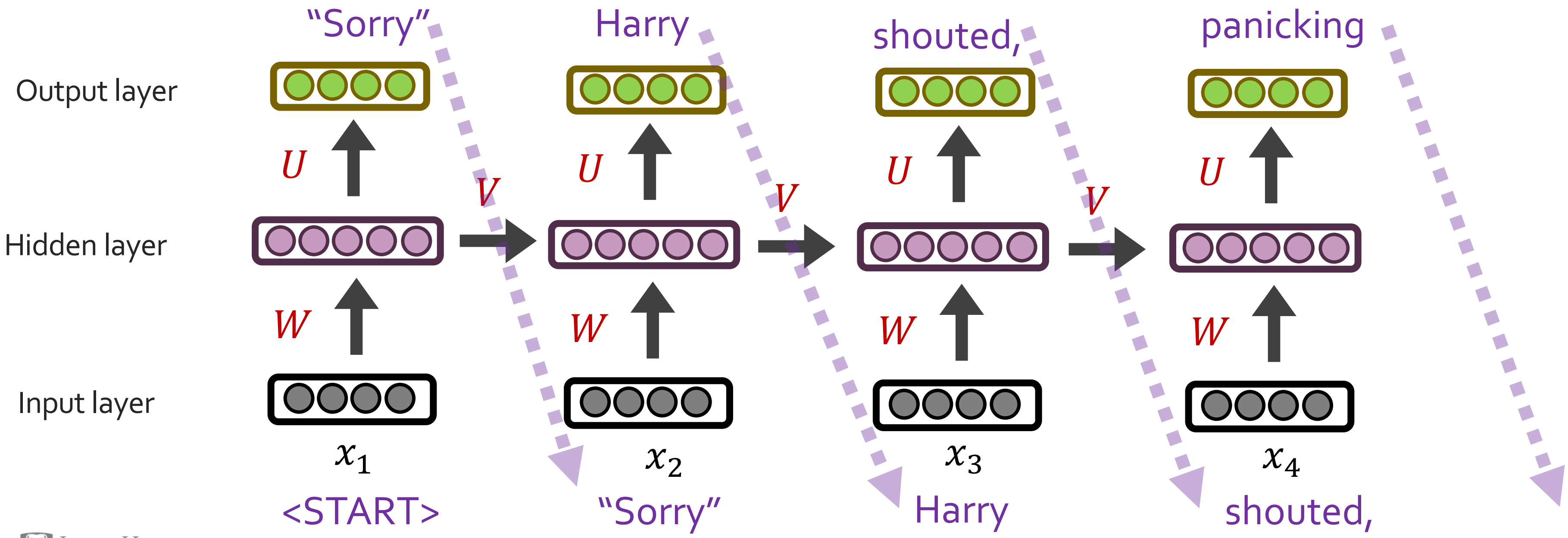
- Despite these limits, RNNs provided improvements at the time that they were introduced and laid the foundation for the future progress.

- **Next:** Pre-training RNNs

Bonus:
Pre-training
Representations of RNNs

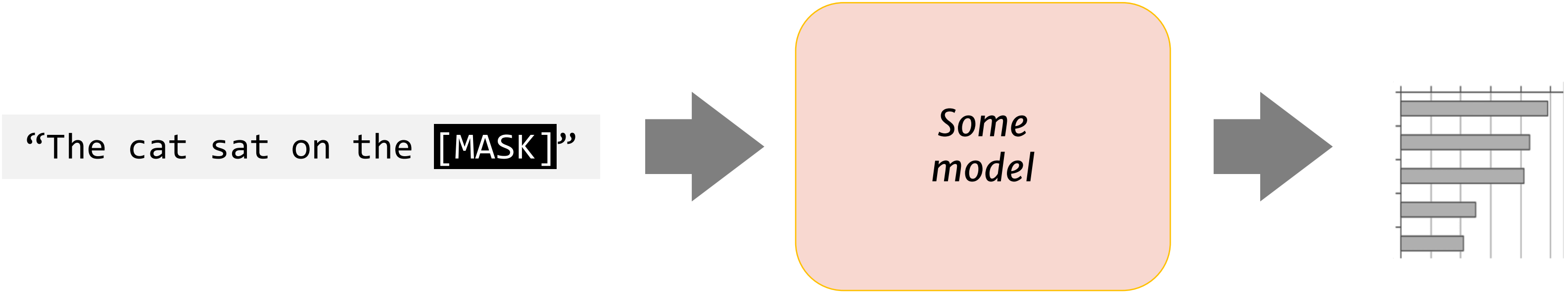
Recap: Recurrent Neural Networks

- Repeated use of a **finite** model

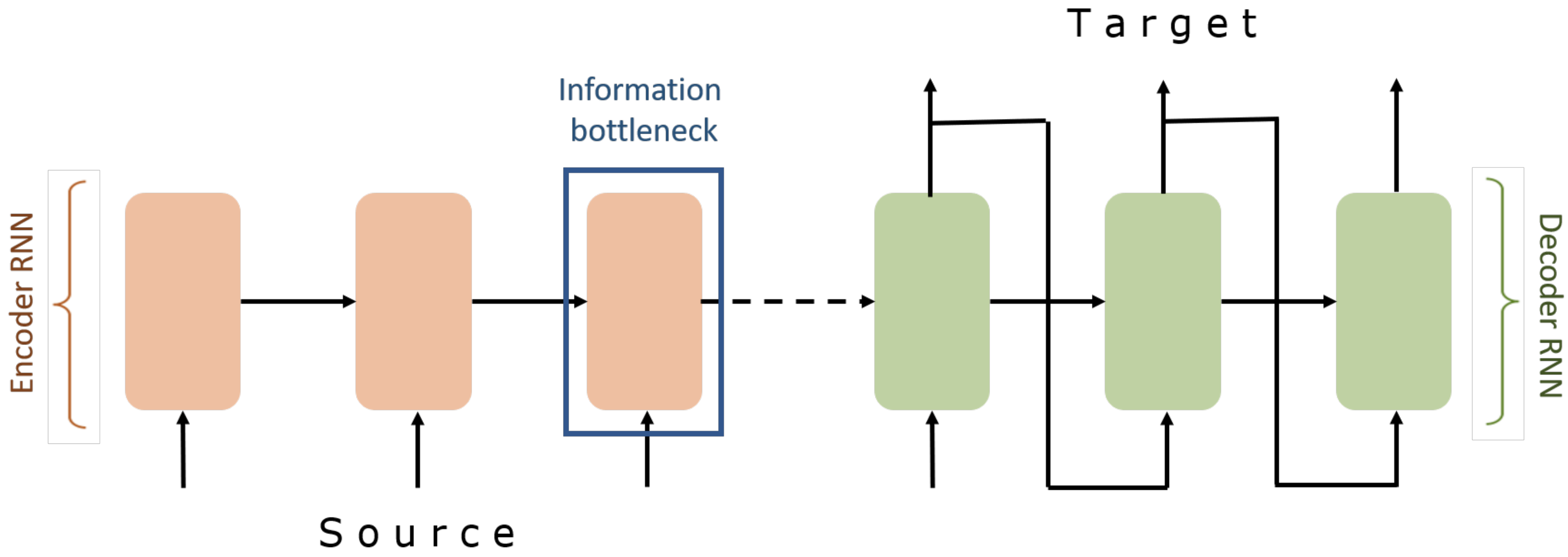


Recap: Encoder-Decoder Architectures

- It is useful to think of generative models as two sub-models.



Recap: Encoder-Decoder Architectures



Contextual Meaning of Words

- Earlier word embedding methods (e.g., Word2Vec, GloVe) learn a single “static” vector for each word.
  - Static embeddings are not flexible and expressive enough.

- The children love to play outside in the park.
- She went to see a play at the local theater.
- They play the piano beautifully.

Information from context is necessary to capture the correct meaning of the word.

[Deep contextualized word representations, Peters et al. 2018]

ELMo: First Major Self-Supervised LM

- **Goal:** get highly rich, contextualized embeddings (word tokens) that depend on the entire sentence in which a word is used.

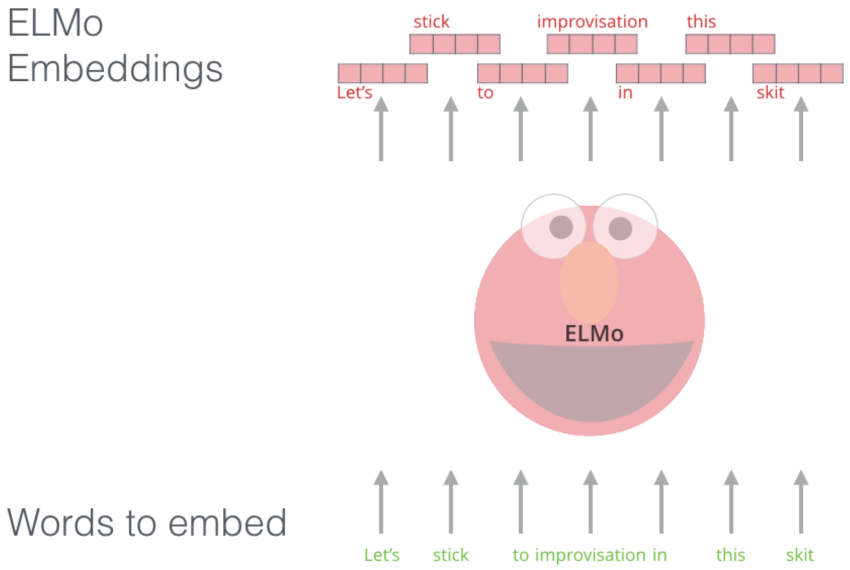
  - The children love to play outside in the park.
  - She went to see a play at the local theater.
  - They play the piano beautifully.

[-0.37, 0.17, -0.36, 0.12, 0.18]   [0.45, -0.26, 0.49, 2.37, -1.2]   [2.05, -1.57, 1.07, 1.37, 0.32]

[Deep contextualized word representations, Peters et al. 2018]

ELMo: First Major Self-Supervised LM

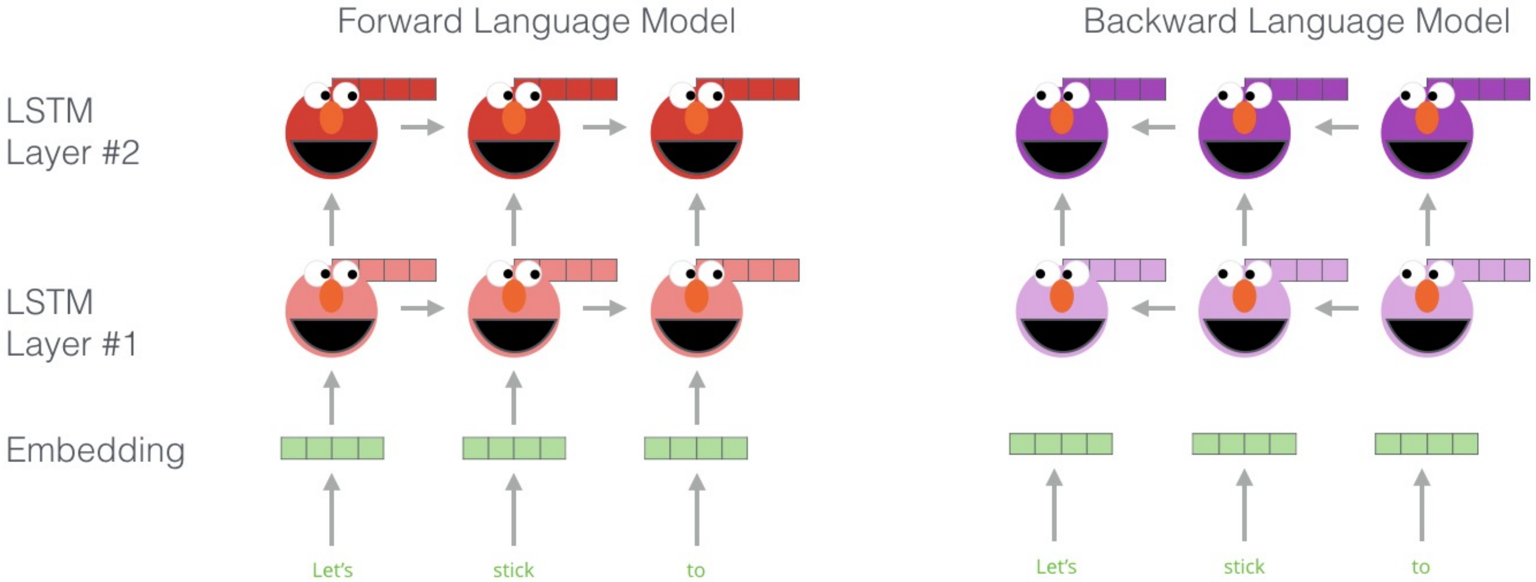
- **Goal:** get highly rich, contextualized embeddings (word tokens) that depend on the entire sentence in which a word is used.



[Deep contextualized word representations, Peters et al. 2018]

ELMo: First Major Self-Supervised LM

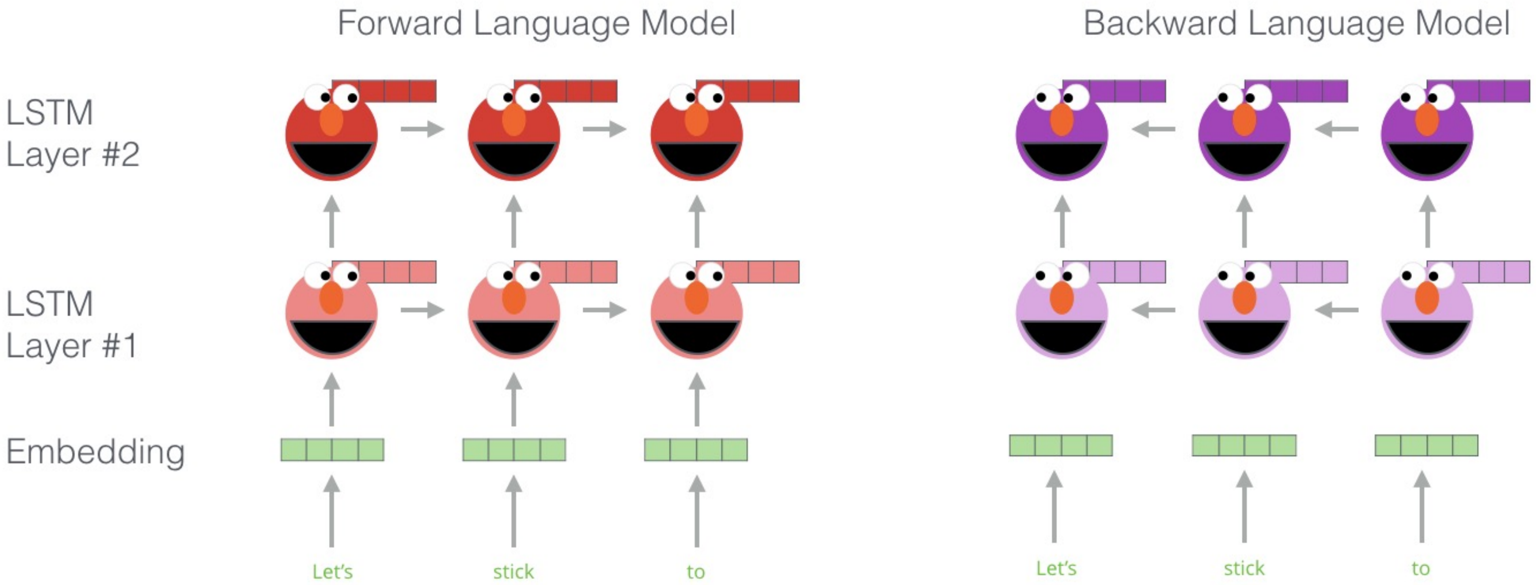
- Use both directions of context (bi-directional), with increasing abstractions (stacked)
  - Two LSTMs in different directions — capture both directions



[Deep contextualized word representations, Peters et al. 2018]

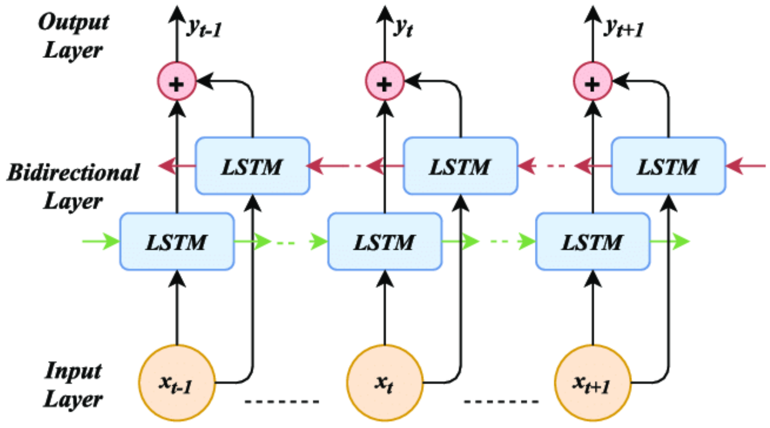
ELMo: First Major Self-Supervised LM

- Linearly combine all abstract representations (hidden layers) and optimize w.r.t. a particular task (e.g., sentiment classification)



[Deep contextualized word representations, Peters et al. 2018]

ELMo: Some Details



- Train a forward language model by modeling prob of each word, given its left context.

$$p(t_1, \ldots, t_k) = \prod_{k=1}^{N} p(t_k \mid t_1, \ldots, t_{k-1})$$

- Similarly, train a backward language model, conditioned on the right context.

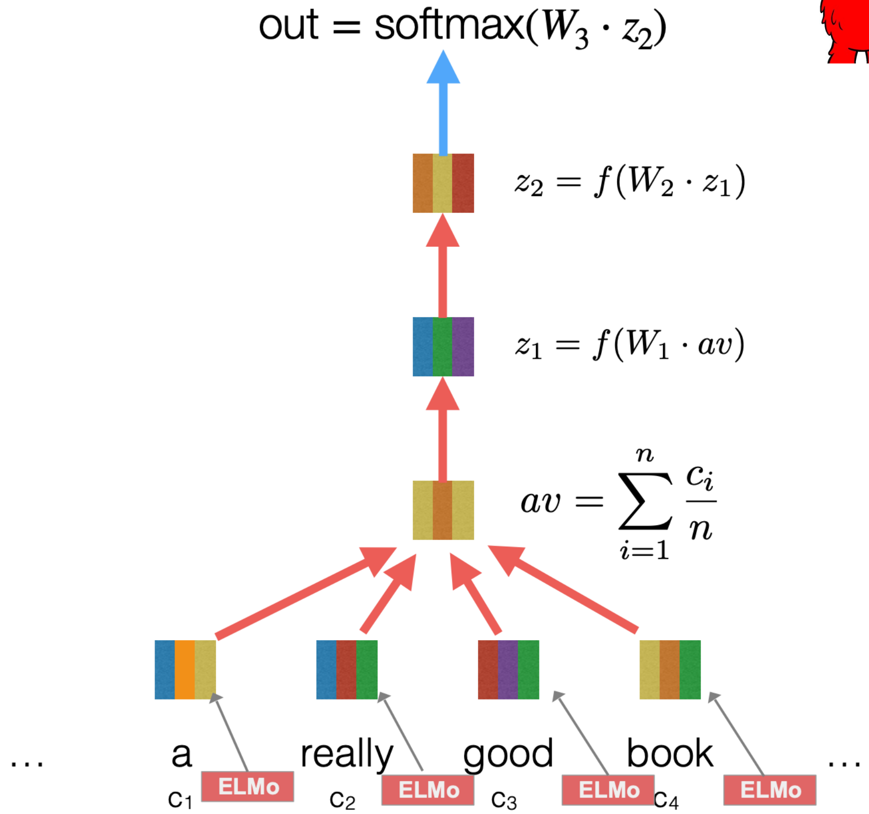
$$p(t_1, \ldots, t_k) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, \ldots, t_N)$$

- Some training details:
  - Use 4096 dim hidden states
  - Residual connections from the first to second layer
  - Trained 10 epochs on 1B Word Benchmark
  - Results in perplexity of ~39

[Deep contextualized word representations, Peters et al. 2018]

Adapting ELMo Representations for Tasks

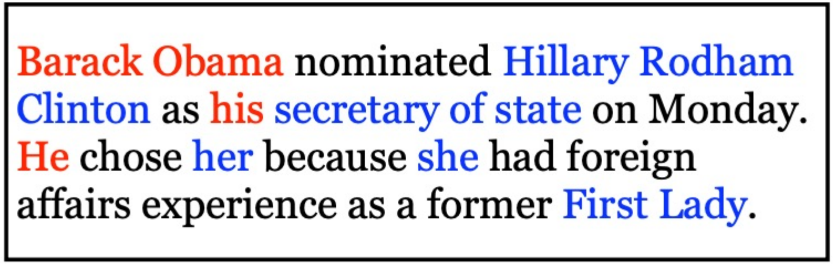
- Fine-tune classifiers using contextualized word representations extracted from ELMo.



[Deep contextualized word representations, Peters et al. 2018]

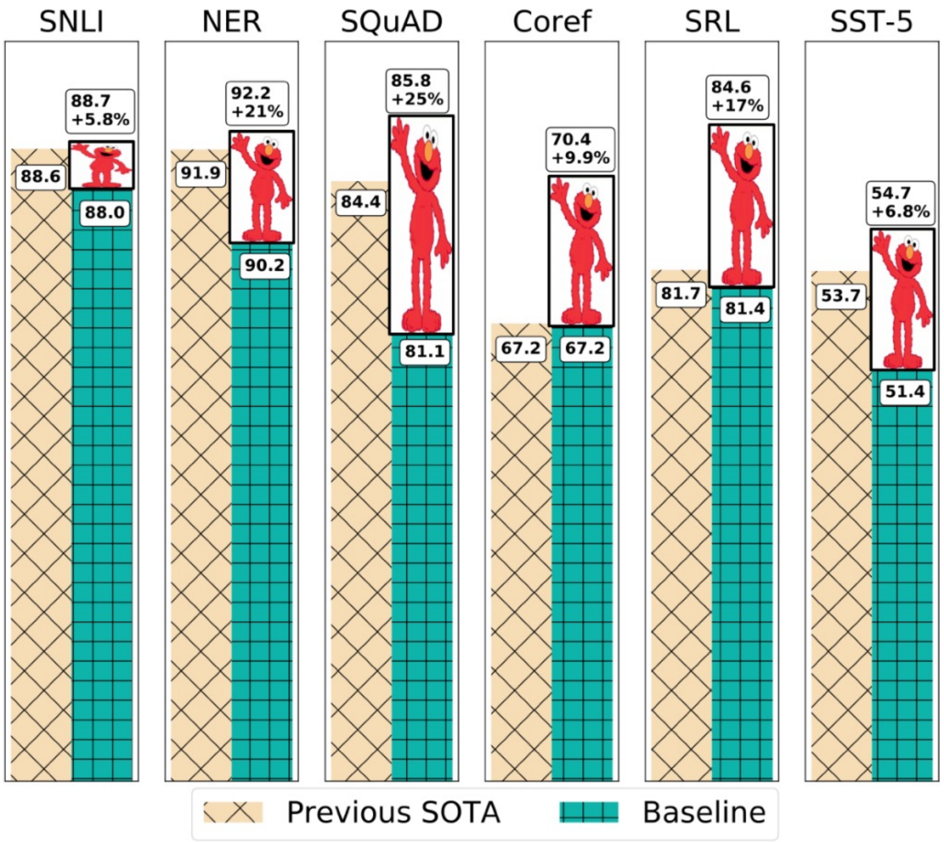
ELMo: Evaluation

- SQuAD: question answering
- SNLI: textual entailment
- SRL: semantic role labeling
- Coref: coreference resolution
- NER: named entity recognition
- SST-5: sentiment analysis

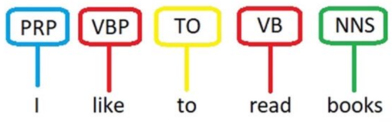


[Deep contextualized word representations, Peters et al. 2018]

Experimental Results

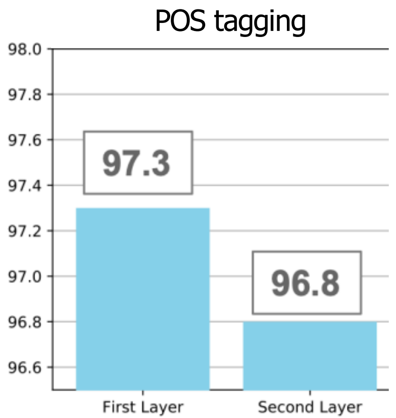


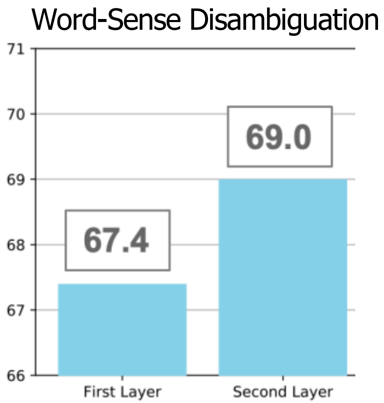
[Deep contextualized word representations, Peters et al. 2018]



The **bank** can guarantee deposits will eventually cover future tuition costs because it invests in adjustable-rate mortgage securities.

bank[1]	Gloss:	a financial institution that accepts deposits and channels the money into lending activities
	Examples:	"he cashed a check at the bank", "that bank holds the mortgage on my home"
bank[2]	Gloss:	sloping land (especially the slope beside a body of water)
	Examples:	"they pulled the canoe up on the bank", "he sat on the bank of the river and watched the currents"





First Layer > Second Layer

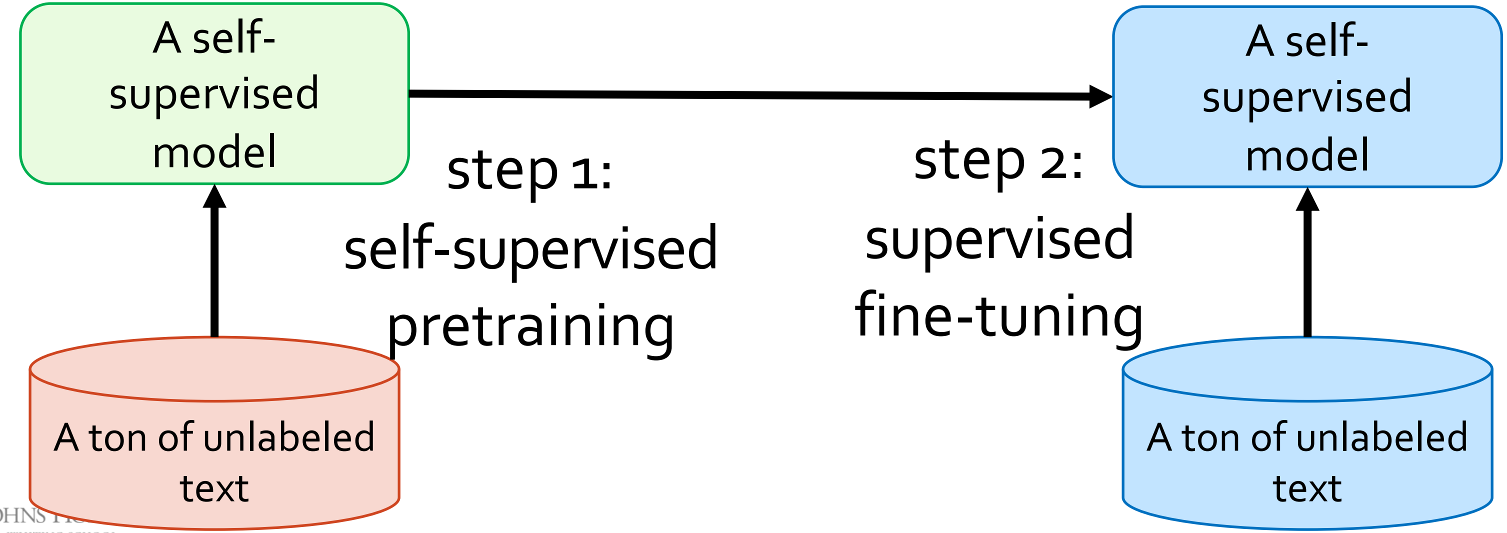
Second Layer > First Layer

Syntactic information is better represented at lower layers while semantic information is captured a higher layers

[Deep contextualized word representations, Peters et al. 2018]

A broader lesson: Pre-training + Fine-tuning

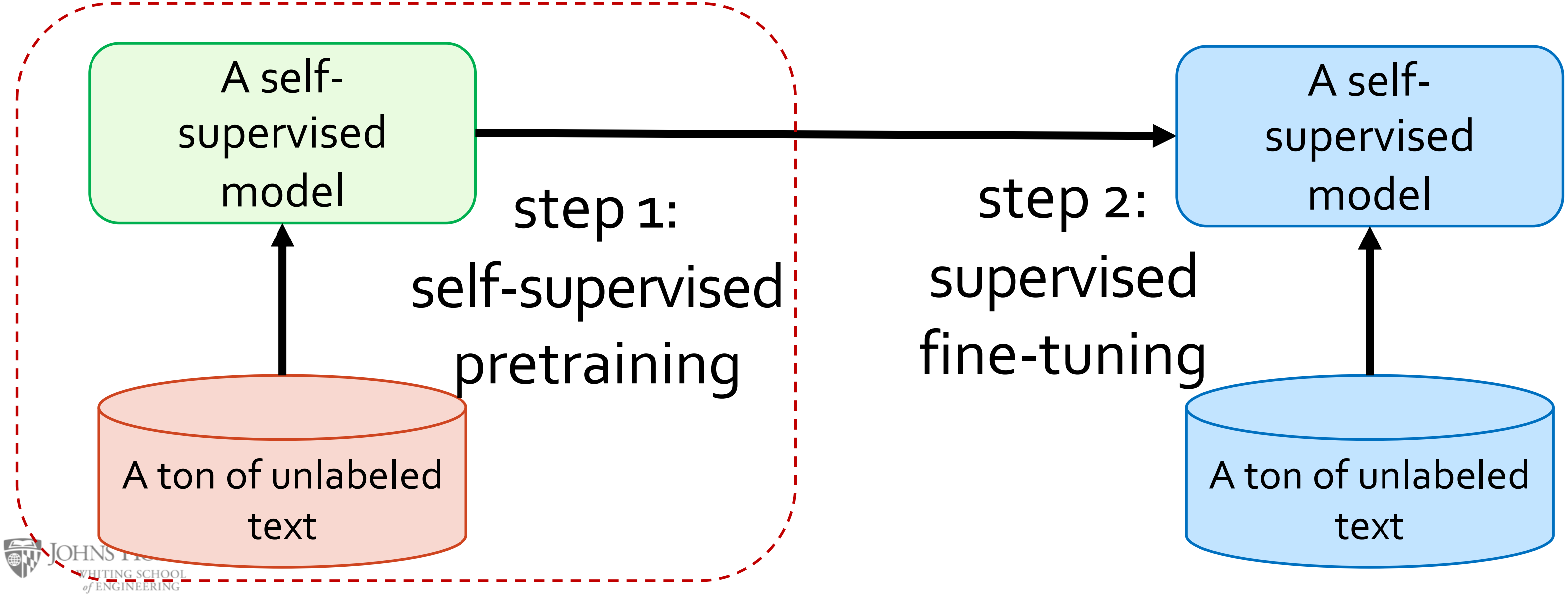
- Let’s say I want to train a model for sentiment analysis
- In the past, I would simply train a supervised model on Word2Vec representation of review sentences (e.g., HW2).



A broader lesson: Pre-training + Fine-tuning

Contextual representation of LMs is much **stronger** than word-level ones.

Now that we have Fixed-Window LM, we can use it to build better classifiers!



Summary

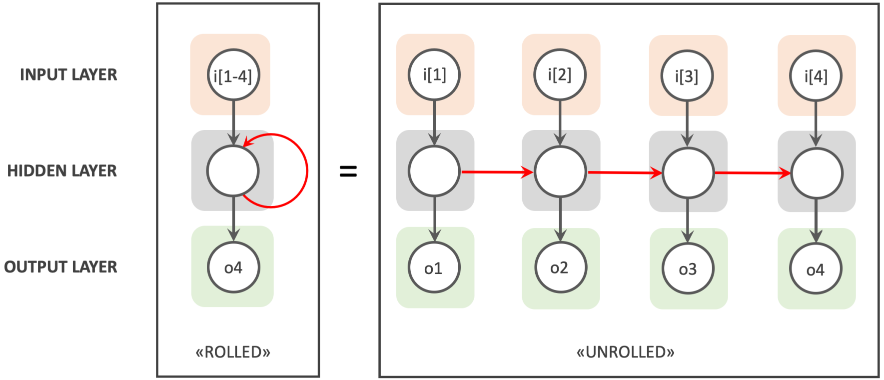
- ELMo: Stacked Bi-directional LSTMs

- ELMo yielded incredibly good contextualized embeddings, which yielded SOTA results when applied to many NLP tasks.

- Main ELMo takeaway: given enough [unlabeled] training data, having tons of parameters model is useful — the system can determine how to best use context.

Summary

- Recurrent Neural Networks
  - A family of neural networks that allow architecture for inputs of variable length



- RNN-LM: LM based on RNNs
- A notable example: **ELMo**

- Cons:
  - Sequential processing
  - While in theory it maintain infinite history, in practice it suffers from long-range dependencies.

Algorithms for Sampling from Language Models

CSCI 601-471/671 (NLP: Self-Supervised Models)

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

https://self-supervised.cs.jhu.edu/sp2024/

The Sampling Question

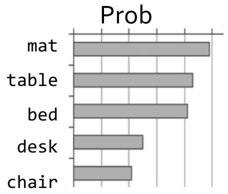
How do we generate language from LMs?

Given:

$$\mathbf{P}(\overbrace{X_t}^{\text{next word}} \mid \overbrace{X_1, \ldots, X_{t-1}}^{\text{context}})$$

"The cat sat on the [MASK]"

Some model

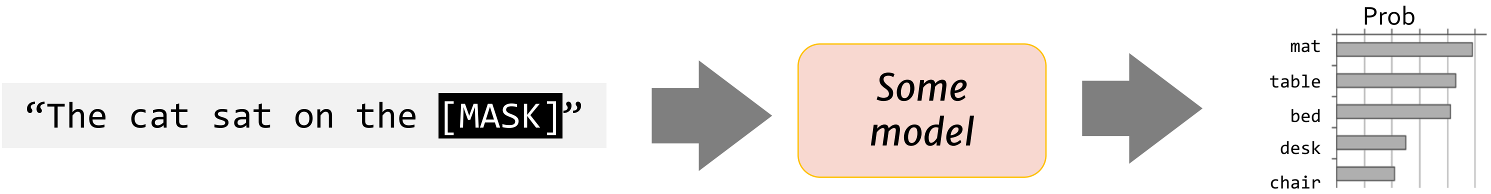


Approach 1: Greedy (Argmax)

- Challenge:
  - Generates boring results — not creative.
  - May repeat itself .

“I went to the place that the place that the place that the place ...”

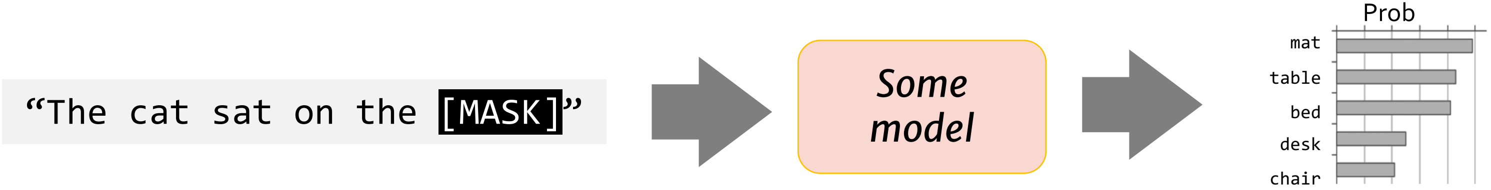
$$x_t = \text{argmax}\, \mathbf{P}(\overbrace{X_t}^{\text{next word}} | \overbrace{X_1, \ldots, X_{t-1}}^{\text{context}})$$



Approach 2: Sampling from the whole distribution

- Challenge: Likely to result in lots of nonsensical generations.
- Reason: LMs distribution is more meaningful about high-prob items, but as we get further away from high-prob items, the probs are less meaningful.

$$x_t \sim \mathbf{P}(\overbrace{X_t}^{\text{next word}} | \overbrace{X_1, \ldots, X_{t-1}}^{\text{context}})$$



[The Curious Case of Neural Text Degeneration, Holtzman et al., 2020]

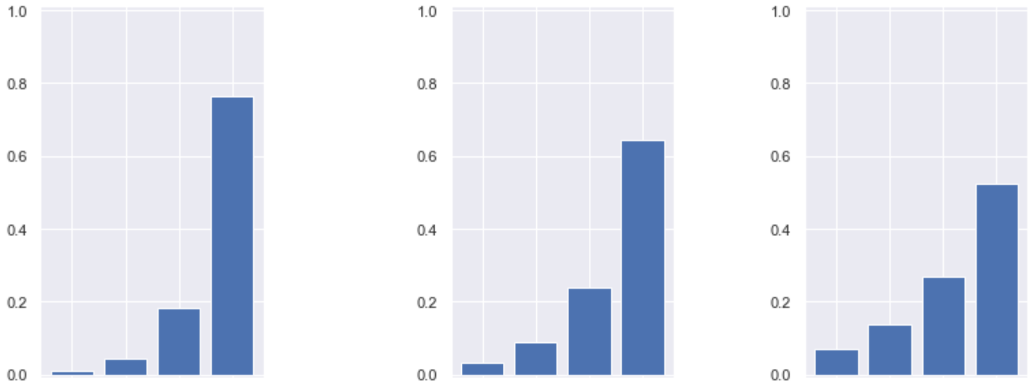
Quiz: Softmax Temperature Parameter

- Let's add parameter T to our Softmax definition:

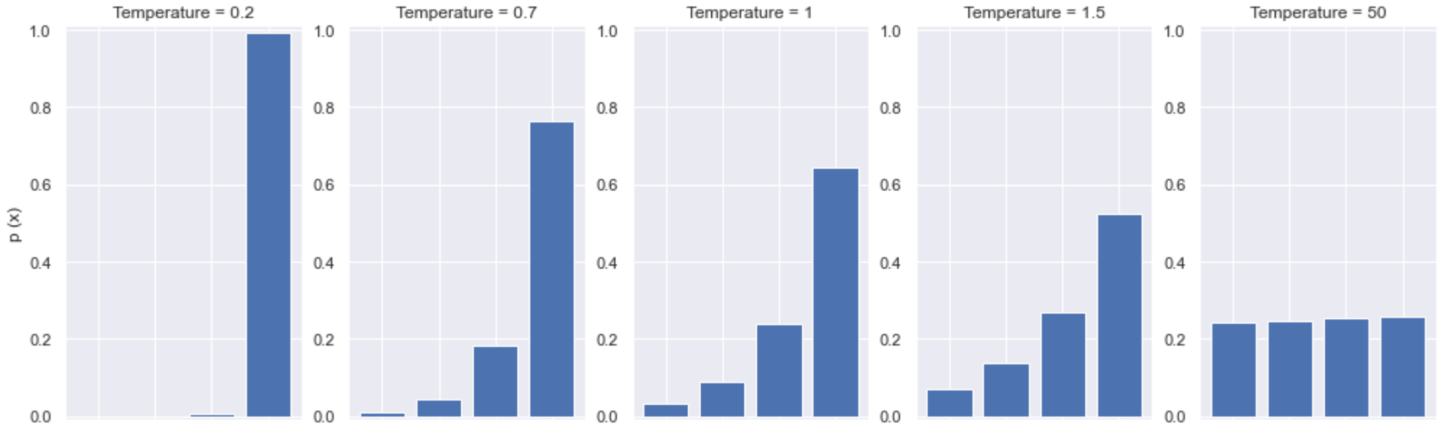
$$\frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

- Suppose if T=1, the output of Softmax looks like this:

- How will increasing T it change this distribution?

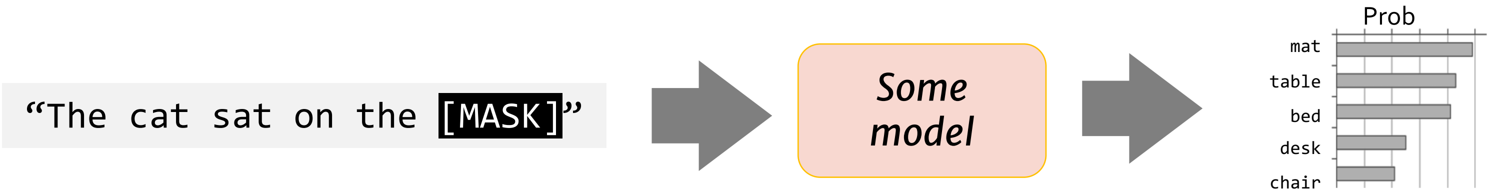


Approach 3: Sampling + Temperature



$$\frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Small-ish T would assign more prob to the top of the distribution, while not losing diversity.

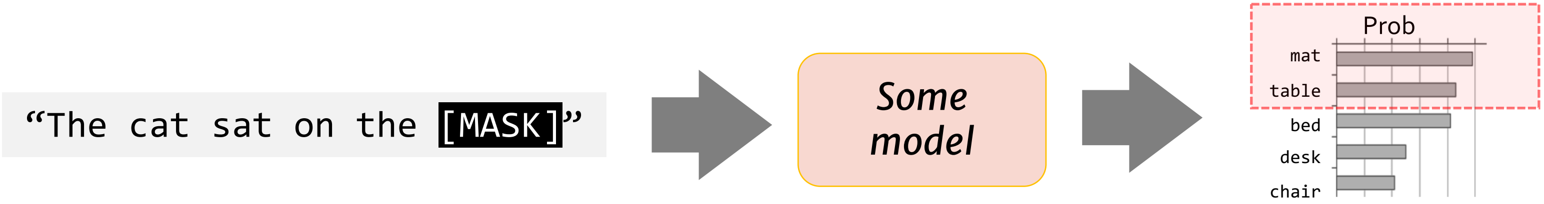


[The Curious Case of Neural Text Degeneration, Holtzman et al., 2020]

Approach 4: Top-p Sampling (Nucleus sampling)

- On each step, randomly sample from the distribution, but restricted to just the top-p most probable words
  - Like pure sampling, but truncate the distribution to high-prob content

- p=1 is basically sampling from the whole distribution

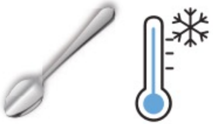


[The Curious Case of Neural Text Degeneration, Holtzman et al., 2020]

Local Incoherence

Pure Sampling

The Australian Food Safety Authority has warned Australia's beaches may be revitalised this year because healthy seabirds and seals have been on the move. More than 50,000 seabirds, sea mammals and seahorses have been swept into the sea by the Holden CS118 and Adelaide Airport CS300 from 2013. A major white-bat and umidauda migration across Australia is under way in Australia for the first time, with numbers reaching an estimated 50,000.



Sampling, t=0.9

Last week's intense storms and a series of powerful cyclones have been officially blamed for the deaths of at least nine large fin whales near Whitsundays - the largest loss of any species globally. The fin whales: packed in the belly of one killer whale thrashing madly in fear as another tries to bring it to safety. When the colossal animal breached the waters of Whitsundays, he'd been seen tagged for a decade.

Nucleus, p=0.95

There has been an unprecedented number of calves caught in the nets of whaling stations that operate in WA. Pilot whales continue to migrate to feeding grounds to feed their calves. They are now vulnerable due to the decline of wild populations; they are restricted to one breeding site each year. Image copyright Yoon Bo Kim But, with sharp decline in wild populations the size of the Petrels are shrinking and dwindling population means there will only be room for a few new fowl.

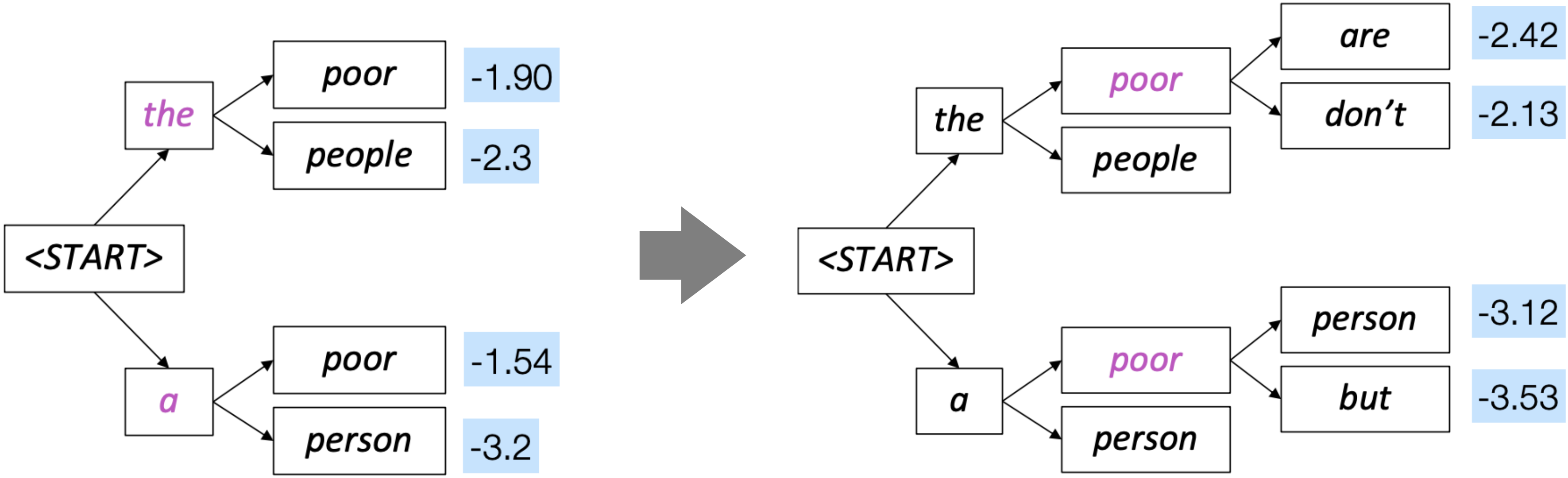
WebText

Poor nutrition has led to a rise in the number of stranded humpback whales on the West Australian coast, veterinary researchers have said. Carly Holyoake, from Murdoch University, at the Australian Veterinary Association's annual conference in Perth on Wednesday, said an unprecedented number of mostly young whales had become stranded on the coast since 2008.

[The Curious Case of Neural Text Degeneration, Holtzman et al., 2020]

Fancier Approaches: Beam Search

- A heuristic search that allows maximizing words probabilities for a window of words
- Out of scope for us. Feel free to check it in your own time.



Many others that we do not cover …

- There are many other algorithms for sampling sentences from LMs that we will not see in this course.

HuggingFace🤗 Generation Function

- **min_length** (`int`, *optional*, defaults to 10) — The minimum length of the sequence to be generated.
- **do_sample** (`bool`, *optional*, defaults to `False`) — Whether or not to use sampling ; use greedy decoding otherwise.
- **early_stopping** (`bool`, *optional*, defaults to `False`) — Whether to stop the beam search when at least `num_beams` sentences are finished per batch or not.
- **num_beams** (`int`, *optional*, defaults to 1) — Number of beams for beam search. 1 means no beam search.
- **temperature** (`float`, *optional*, defaults to 1.0) — The value used to module the next token probabilities.
- **top_k** (`int`, *optional*, defaults to 50) — The number of highest probability vocabulary tokens to keep for top-k-filtering.
- **top_p** (`float`, *optional*, defaults to 1.0) — If set to float < 1, only the most probable tokens with probabilities that add up to `top_p` or higher are kept for generation.
- **repetition_penalty** (`float`, *optional*, defaults to 1.0) — The parameter for repetition penalty. 1.0 means no penalty. See this paper for more details.

https://huggingface.co/docs/transformers/main/en/main_classes/text_generation#transformers.GenerationMixin.generate

Summary on Sampling Algorithms

- Greedy decoding: a simple method; gives low quality output
- Sampling methods are a way to get more diversity and randomness
  - Good for open-ended / creative generation (poetry, stories)
  - Top-p sampling allows you to control diversity
- Others: Beam search searches for high-probability output