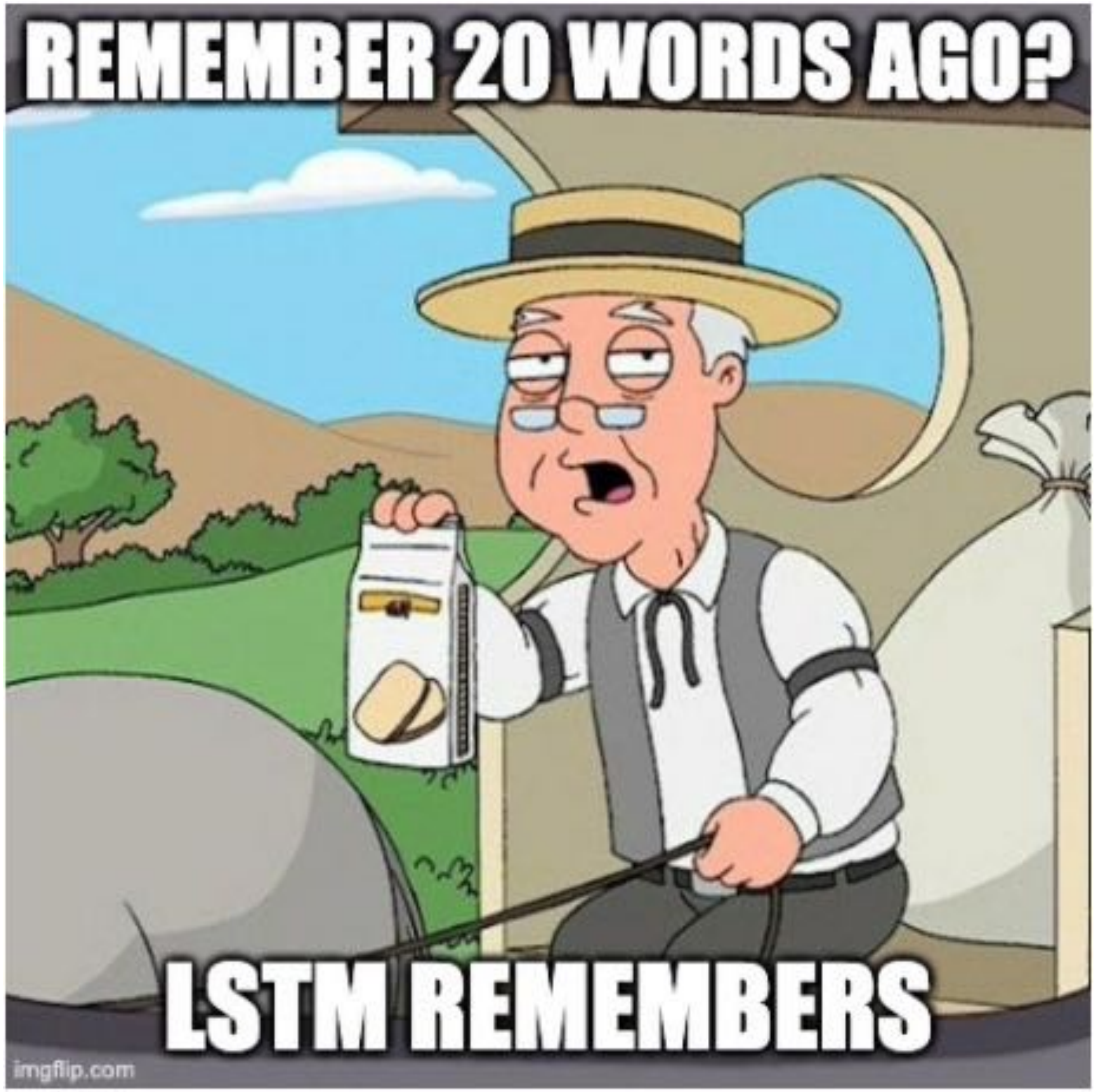
Lecture 5: Recurrent Neural Networks

Contextualized, Token-based Representations

Harvard

AC295/CS287r/CSCI E-115B

Chris Tanner

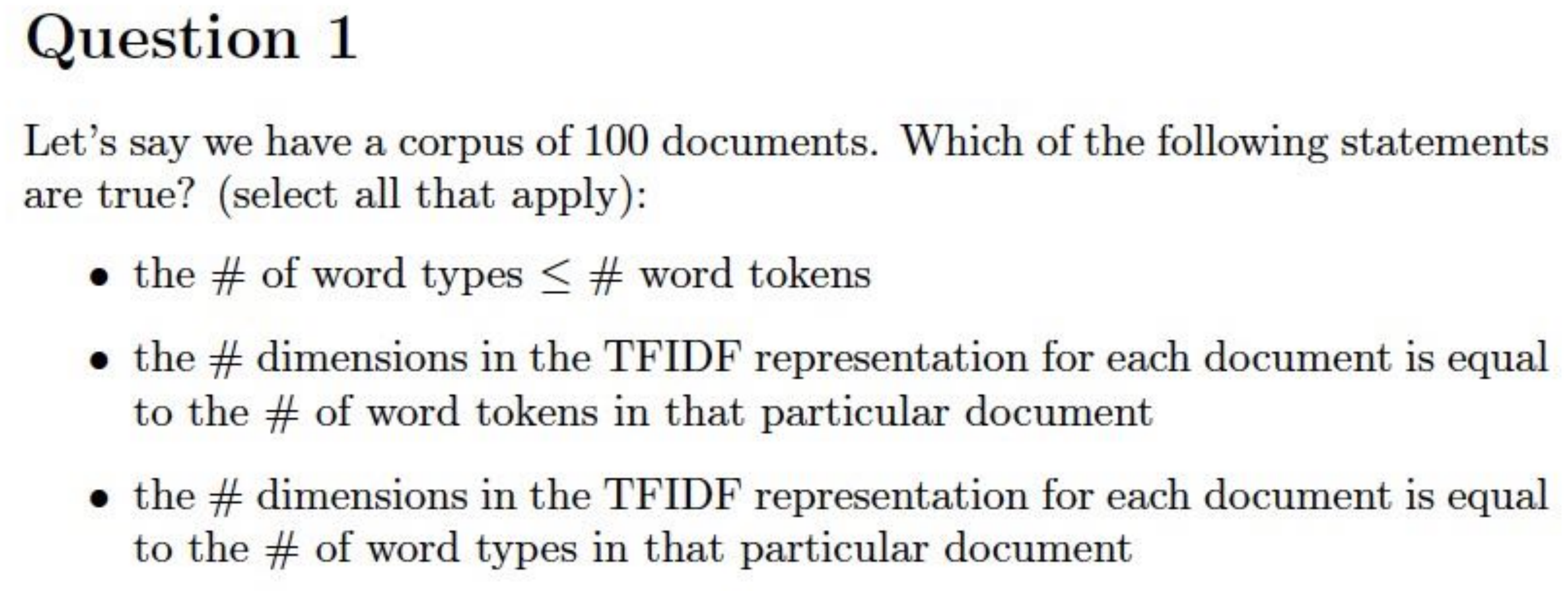


Bonus: The Last Emperor (1998) remembers, too. https://www.youtube.com/watch?v=GPPfCS7qv-8

ANNOUNCEMENTS

- Keep an eye on the **HW1 Errata**, posted on Ed. HW1 is due in 1 week!
- Research Proposals are due in 2 weeks, Sept 30. Start skimming papers and talking with teammates.
- Office Hours:
  - if Zoom room is empty, the TF is likely in a break-out room helping a student 1-on-1 w/ their code.
  - Please reserve your coding questions for the TFs and/or EdStem, as I hold office hours solo, and debugging code can easily bottleneck the queue.

QUIZ 1



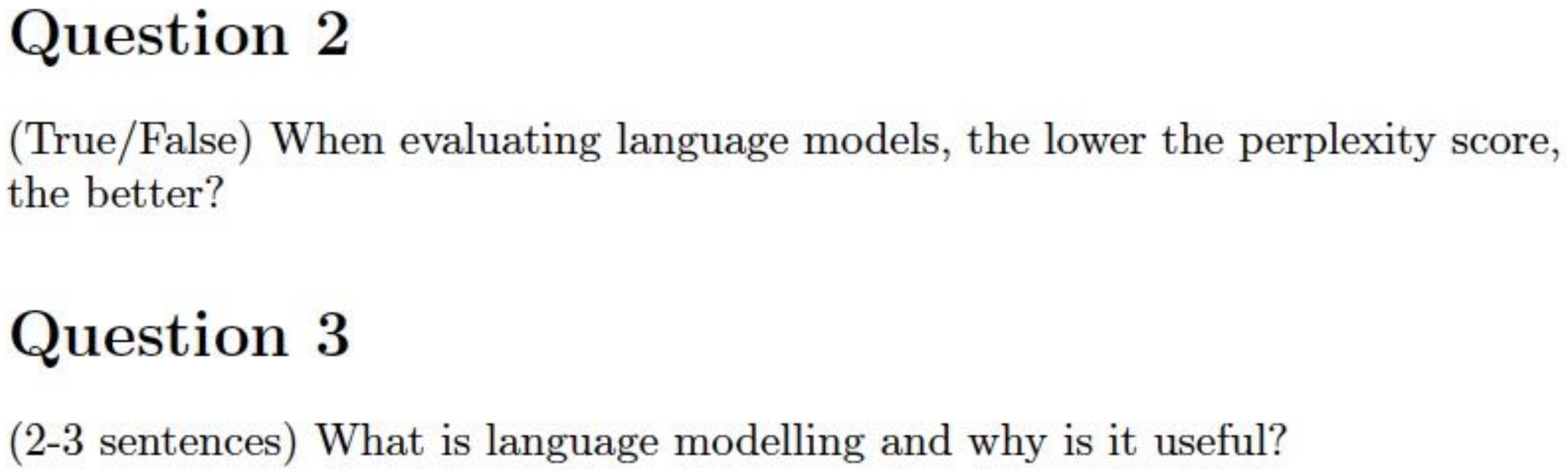
Question 1

Let's say we have a corpus of 100 documents. Which of the following statements are true? (select all that apply):

- the # of word types $\leq$ # word tokens
- the # dimensions in the TFIDF representation for each document is equal to the # of word tokens in that particular document
- the # dimensions in the TFIDF representation for each document is equal to the # of word types in that particular document

$$\text{TFIDF} = f_{w_i} * \log\left(\frac{\text{\# docs in corpus}}{\text{\# docs containing } w_i}\right)$$

QUIZ 1



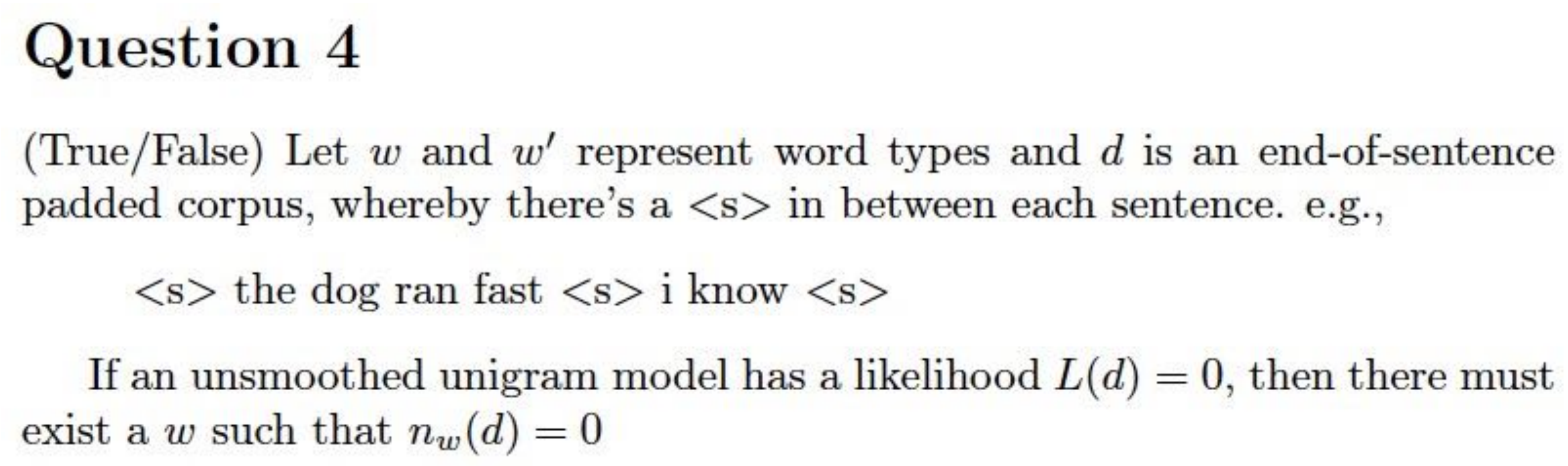
Question 2

(True/False) When evaluating language models, the lower the perplexity score, the better?

Question 3

(2-3 sentences) What is language modelling and why is it useful?

QUIZ 1



Question 4

(True/False) Let w and w' represent word types and d is an end-of-sentence padded corpus, whereby there's a <s> in between each sentence. e.g.,

<s> the dog ran fast <s> i know <s>

If an unsmoothed unigram model has a likelihood $L(d) = 0$, then there must exist a w such that $n_w(d) = 0$

RECAP: L4

Distributed Representations: dense vectors (aka embeddings) that aim to convey the meaning of tokens*

A "word embeddings" are distributed representations, and they specifically refer to when you have type-based representations. i.e., a single representation for each unique word type. All "banks" would have the same learned vector.

* the token is almost always a word, but technically could be a character or sub-words

RECAP: L4

An auto-regressive LM is one that only has access to the previous tokens.

Evaluation: Perplexity

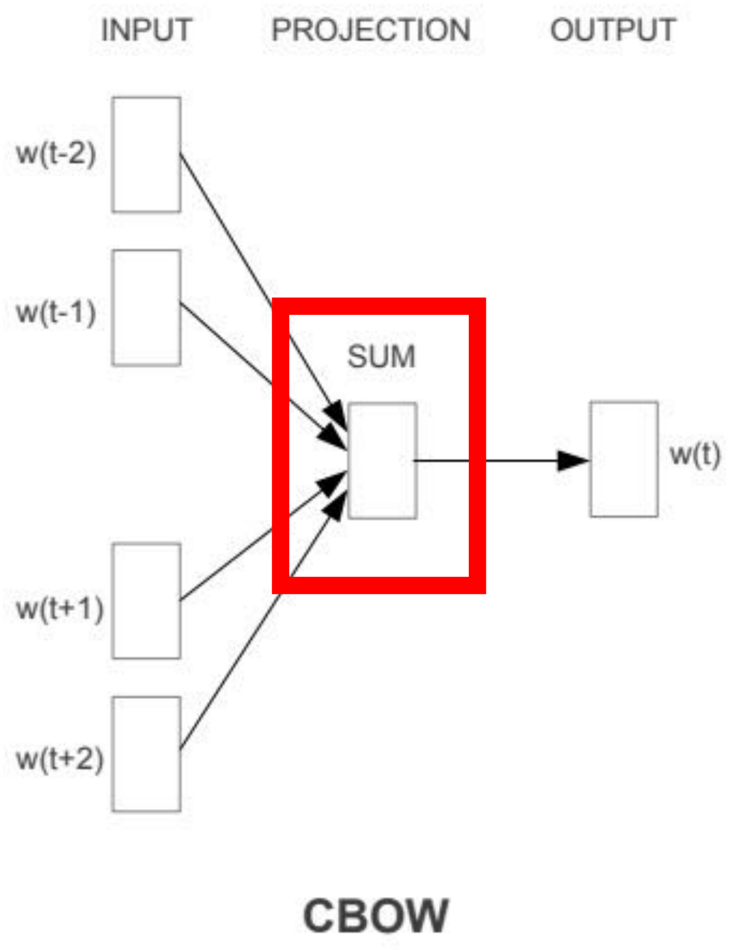
A masked LM can peak ahead, too. It "masks" a word within the context (i.e., center word).

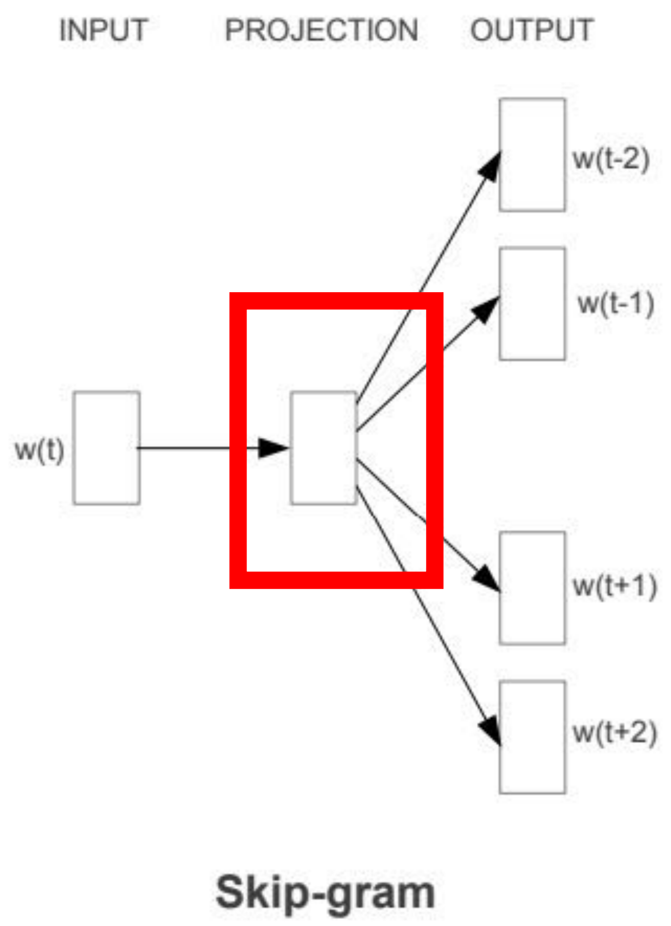
Evaluation: downstream NLP tasks that uses the learned embeddings.

Both of these can produce useful word embeddings.

RECAP: L4

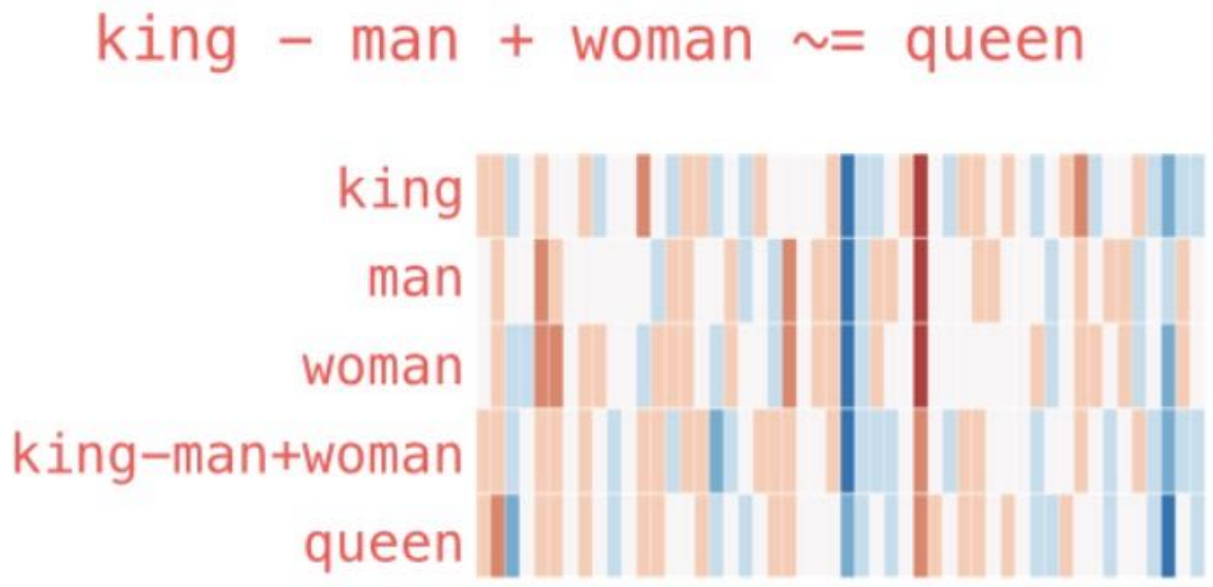
These are the learned word embeddings that we want to extract and use

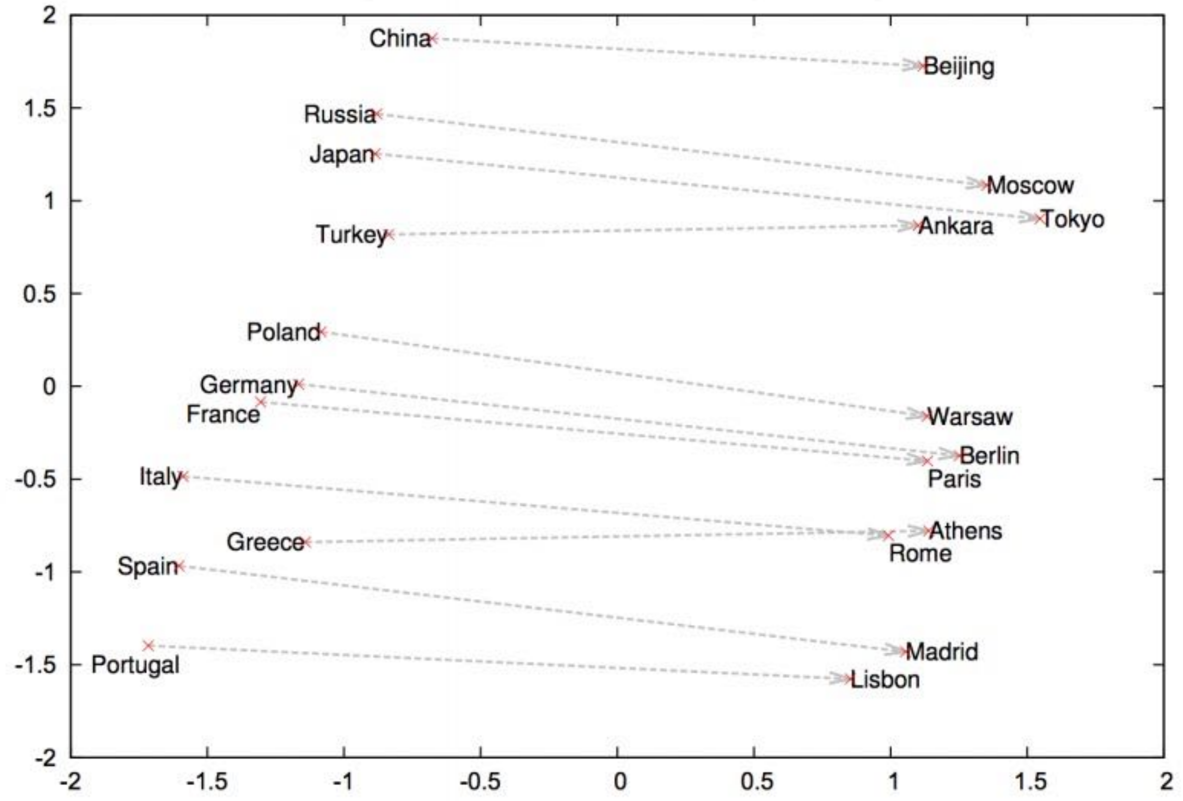




RECAP: L4

word2vec was revolutionary and yields great word embeddings





RECAP: L4

➖ 1. More context while avoiding sparsity, storage, and compute issues

➕ 2. No semantic information conveyed by counts (e.g., vehicle vs car)

New goals!

➕ 3. Cannot leverage non-consecutive patterns

Dr. West ____	Dr. Cornell West ____
Occurred 25 times	Occurred 3 times

➕ 4. Cannot capture combinatorial signals (i.e., non-linear prediction)

P(Chef cooked food)	high	P(Customer cooked food)	low
P(Chef ate food)	low	P(Customer ate food)	high

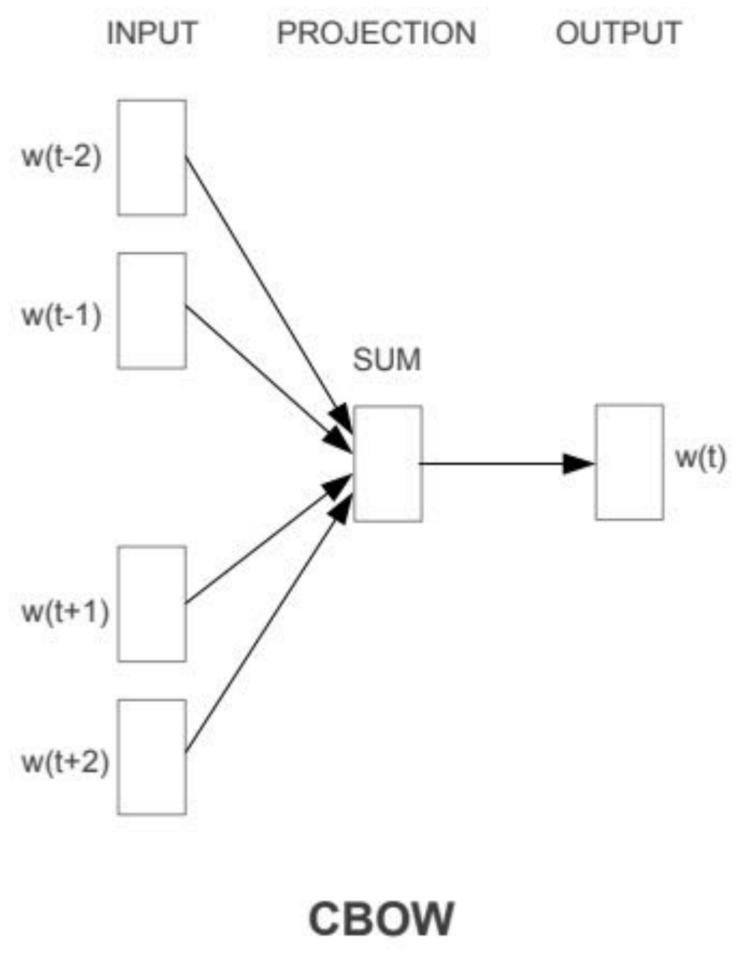
Outline

- Word Embeddings (cont.)
- Recurrent Neural Nets (RNNs)

Outline

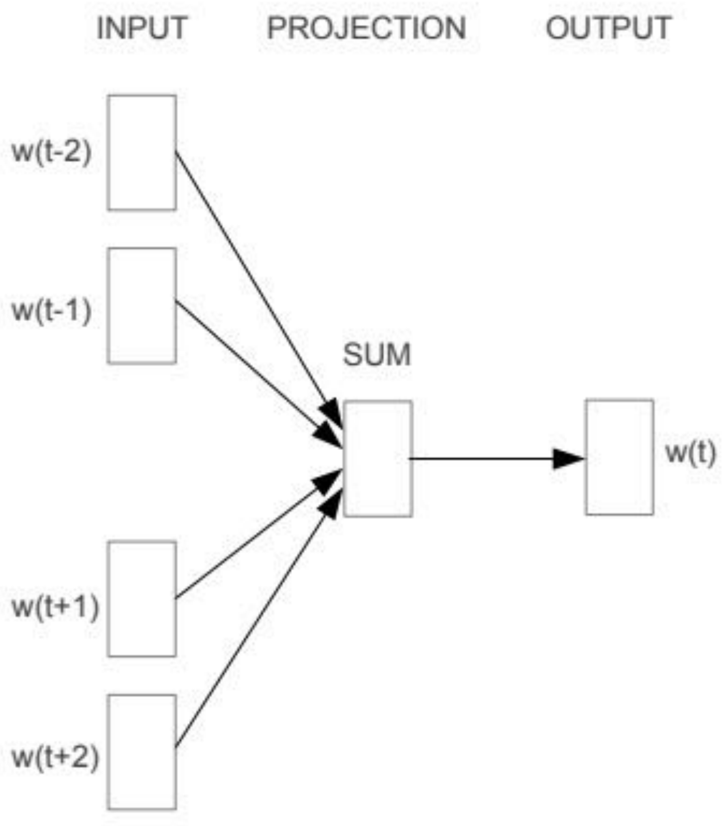
- Word Embeddings (cont.)
- Recurrent Neural Nets (RNNs)

word2vec training



Input words {w_{t-2}, w_{t-1} w_{t+1}, w_{t+2}} yield the same prediction regardless of the ordering. Hence, C<u>BOW</u>

word2vec training



CBOW

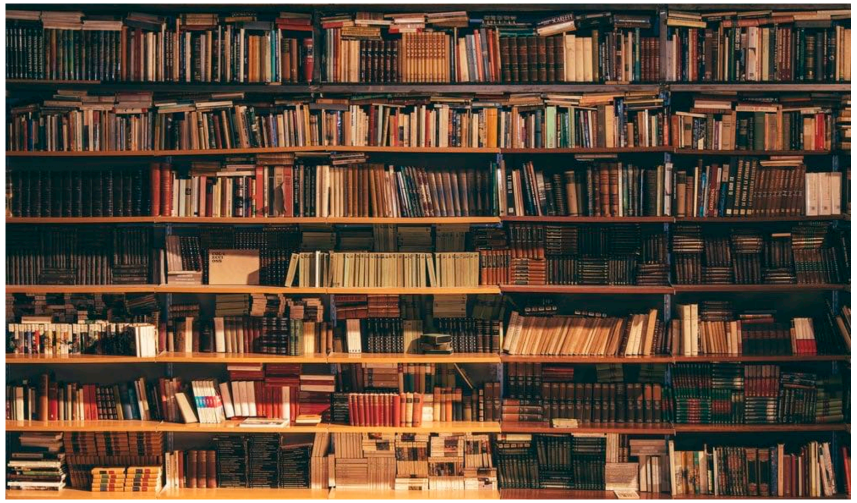
- Words that appear in the same contexts are forced to gravitate toward having the same embeddings as one another (especially if close to each other)
- Imagine two words, w_1 and w_2, that never appear together, but they each, individually have the exact same contexts with *other* words. w_1 and w_2 will have ~identical embeddings!

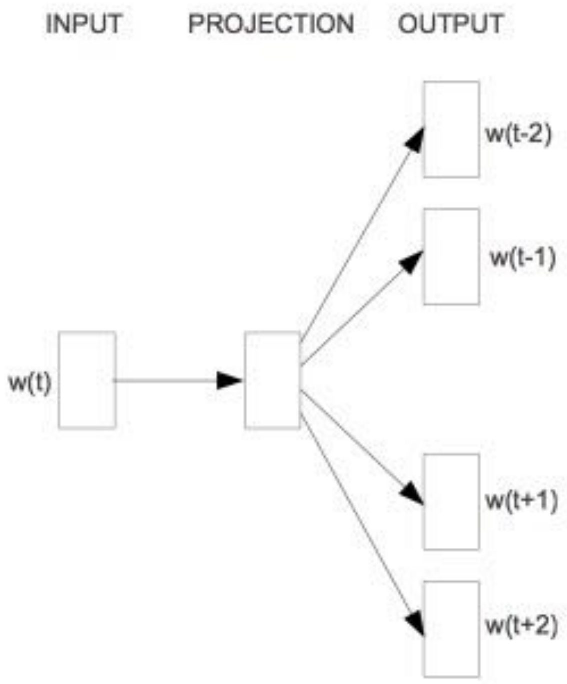
word2vec training

Disclaimer: As a heads-up, no models create embeddings such that the dimensions actually correspond to linguistic or real-world phenomenon.

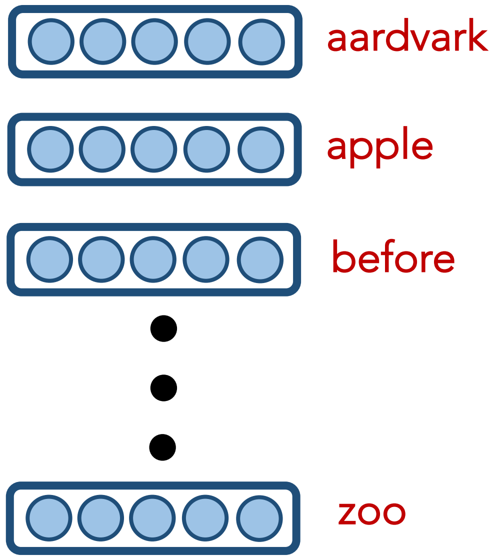
The embeddings are often really great and useful, but no single embedding (in the absence of others) is interpretable.

word2vec training


millions of books



word2vec



word embeddings

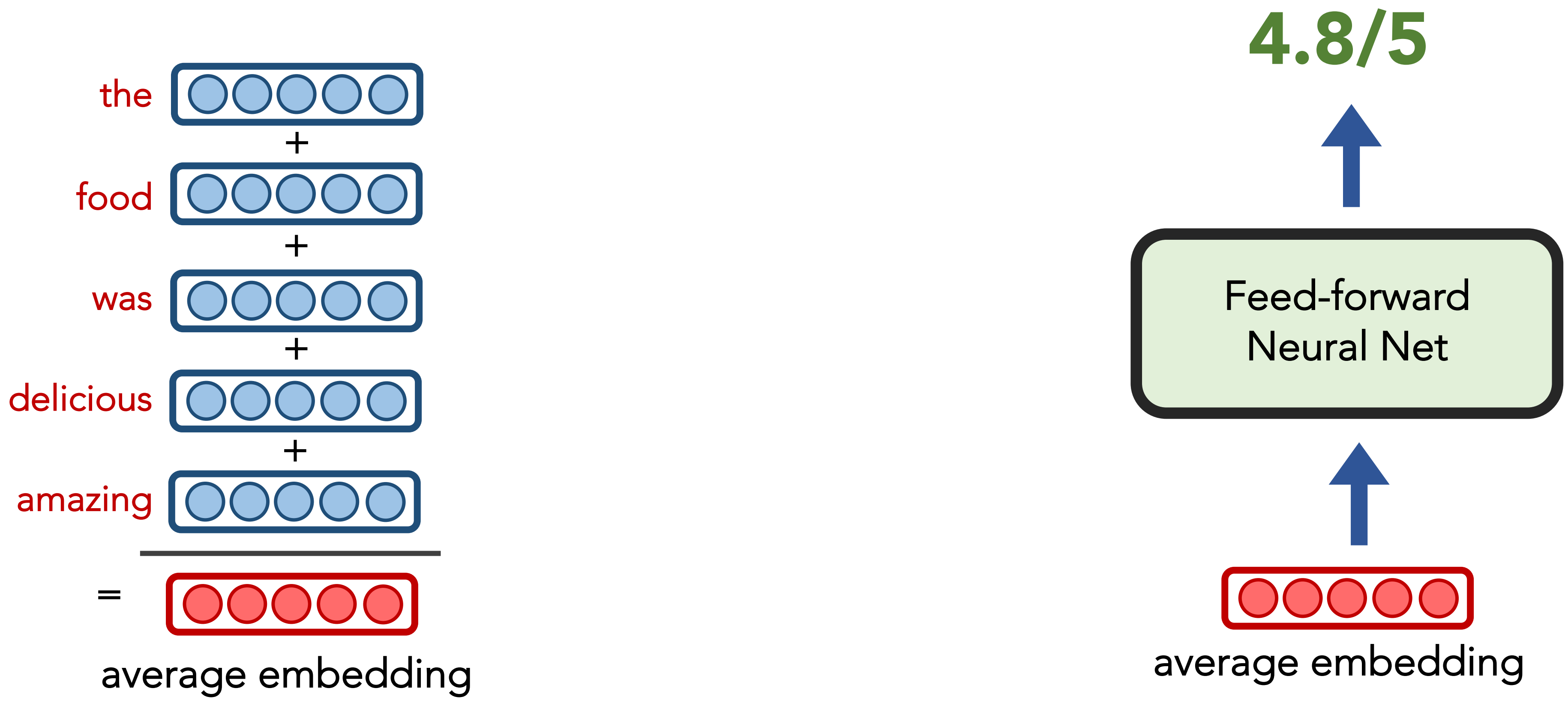
"The food was delicious. Amazing!" → **4.8/5** yelp

word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

"The food was delicious. Amazing!" → **4.8/5** yelp

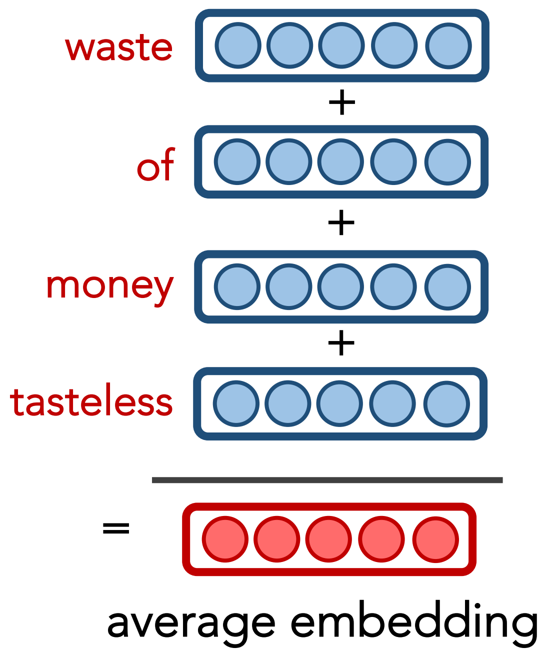


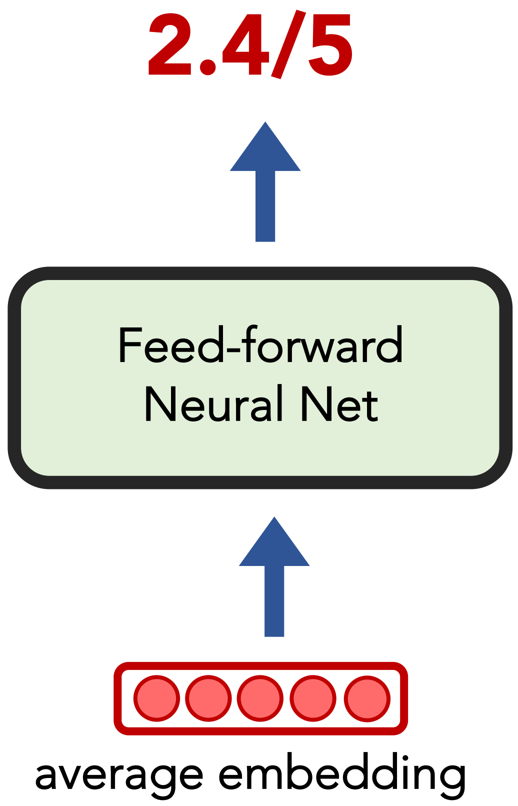
word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

"Waste of money. Tasteless!" → **2.4/5** yelp



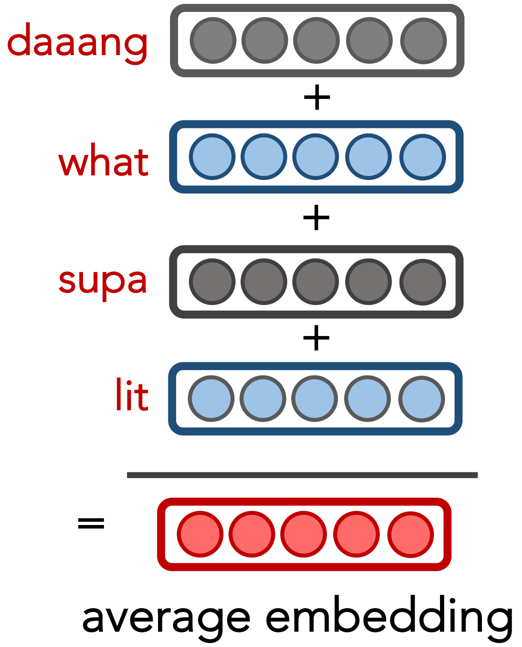


word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

"Daaang. What?! Supa Lit" → **4.9/5** yelp

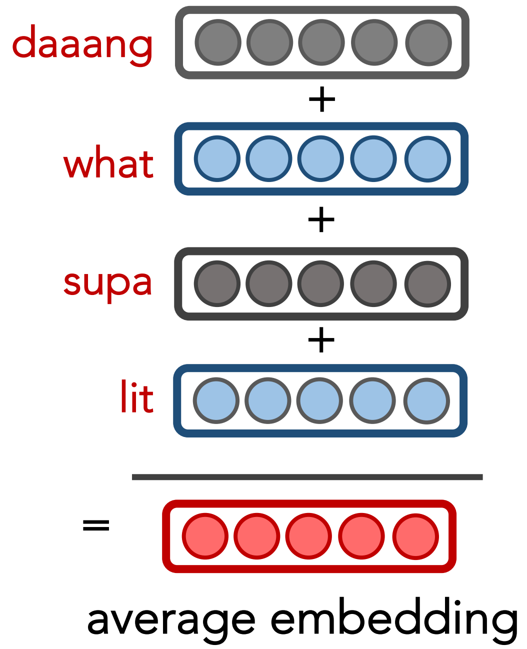


Strengths and weaknesses of word embeddings (type-based)?

word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)



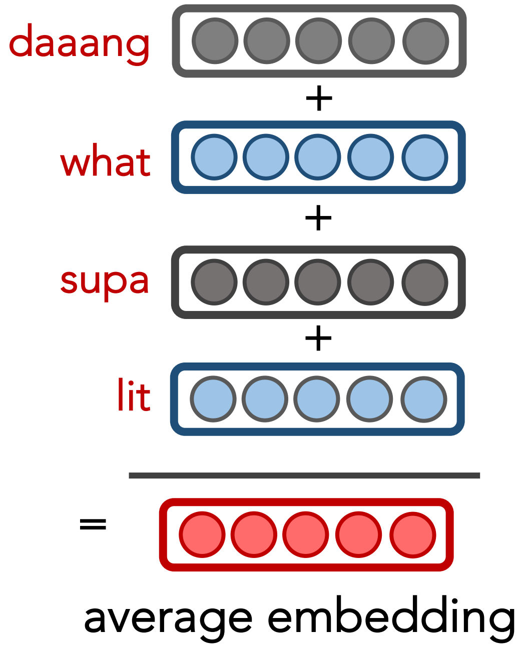
Strengths:

- Can create general-purpose, useful embeddings by leveraging tons of existing data
- Captures semantic similarity

word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)



Issues:

- Not tailored to this dataset
- Out-of-vocabulary (OOV) words
- Limited context
- Each prediction is independent from previous
- A **FFNN** is a clumsy, inefficient way to handle context; fixed context that is constantly being overwritten (no persistent hidden state).
- Requires inputting entire context just to predict 1 word

word embeddings (type-based)

approaches:

- count-based/DSMs (e.g., SVD, LSA)
- Predictive models (e.g., word2vec, GloVe)

word2vec Results

- SkipGram w/ Negative Sampling tends to outperform CBOW
- SkipGram w/ Negative Sampling is slower than CBOW
- Both SkipGram and CBOW are predictive, neural models that take a type-based approach (not token-based).
- Both SkipGram and CBOW can create rich word embeddings that capture both semantic and syntactic information.

Evaluation

We cheated by looking ahead, so it’s unfair to measure perplexity against n-gram or other auto-regressive LM

Intrinsic evaluation:

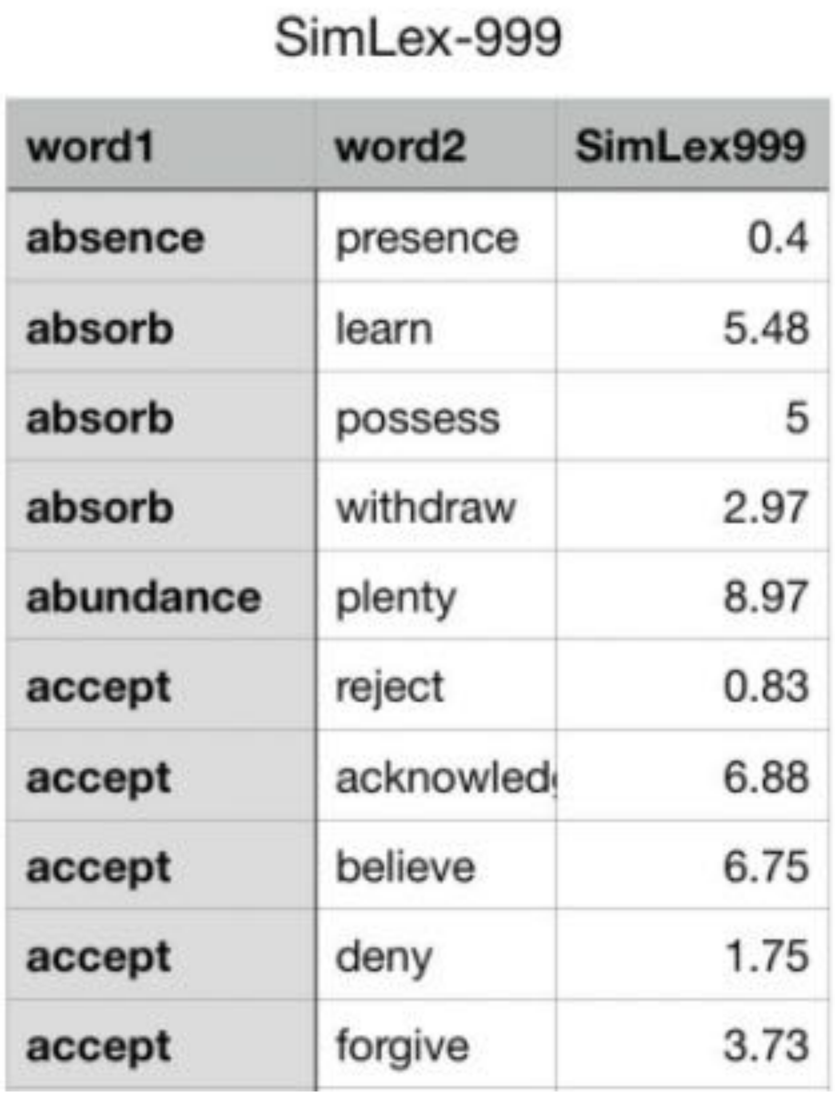
- Word similarity tasks
- Word analogy tasks

Extrinsic evaluation:

- Apply to downstream tasks (e.g., Natural language inference, entailment, question answering, information retrieval)

Evaluation

Word Similarity (not relatedness)



SimLex-999

word1	word2	SimLex999
absence	presence	0.4
absorb	learn	5.48
absorb	possess	5
absorb	withdraw	2.97
abundance	plenty	8.97
accept	reject	0.83
accept	acknowled	6.88
accept	believe	6.75
accept	deny	1.75
accept	forgive	3.73

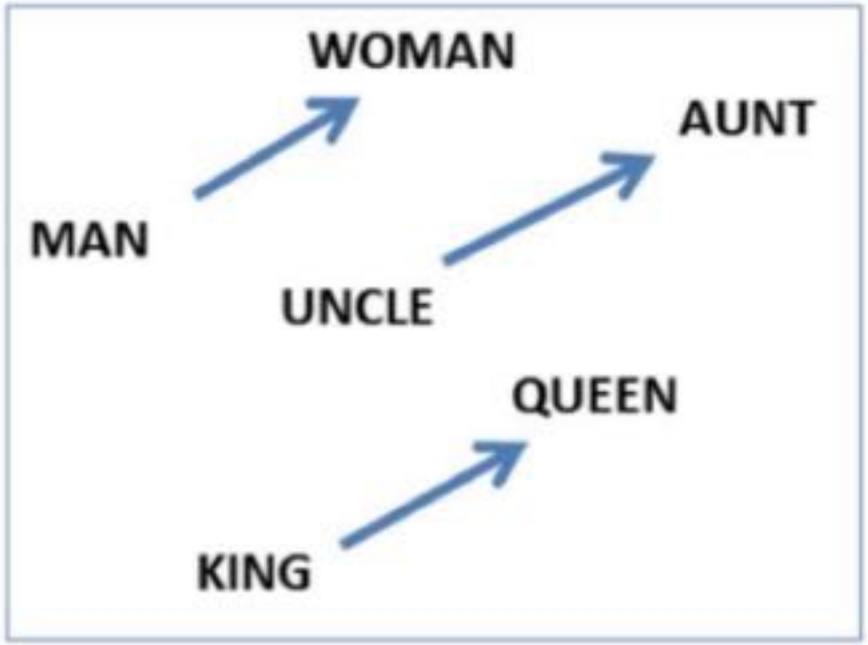
Slide adapted from or inspired by Sam Bowman's NYU NLP 2021

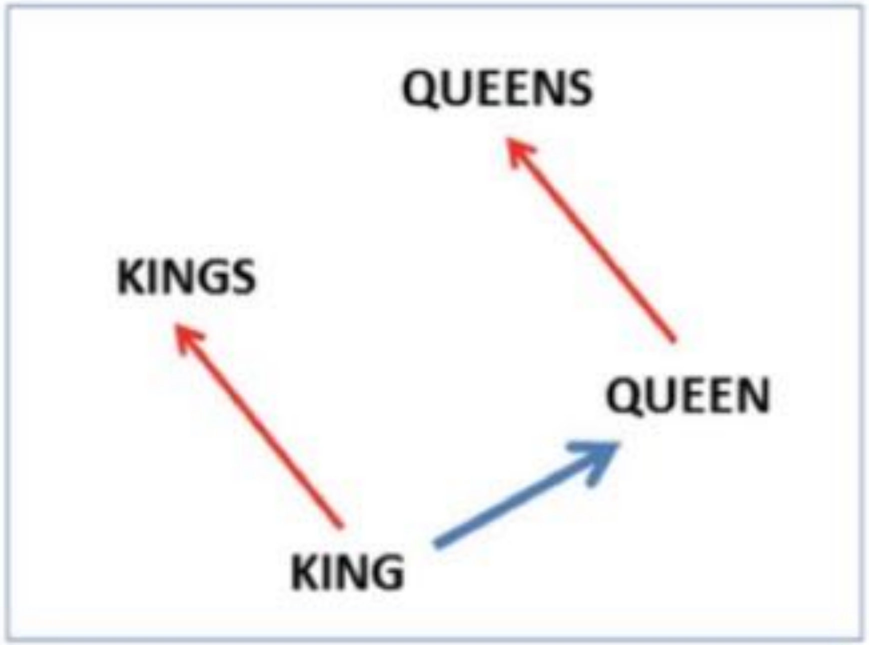
Evaluation

Word Analogy

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) $\approx$ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) $\approx$ vector('Rome')





Slide adapted from or inspired by Sam Bowman's NYU NLP 2021

Outline

- Word Embeddings (cont.)
- Recurrent Neural Nets (RNNs)

Outline

- Word Embeddings (cont.)
- Recurrent Neural Nets (RNNs)

RNNs

We especially need a system that:

- Has an "infinite" concept of the past, not just a fixed window
- For each new input, output the most likely next event (e.g., word)

Motivation

Language often has long-range dependencies:

Emily earned the top grade on the quiz! Everyone was proud of her.

Miquel earned the top grade on the quiz! Everyone was proud of him.

Motivation

Language often has long-range dependencies:

The trophy would not fit in the brown suitcase because it was too big.

The trophy would not fit in the brown suitcase because it was too small.

Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html

Motivation

Language is sequential in nature:

- characters form words.
- words form sentences.
- sentences form narratives/documents

NLP folks like to operate at the word level, as that's the smallest, convenient unit of meaning.

Motivation

Q: What are some other types of data one might model, that are sequential in nature?

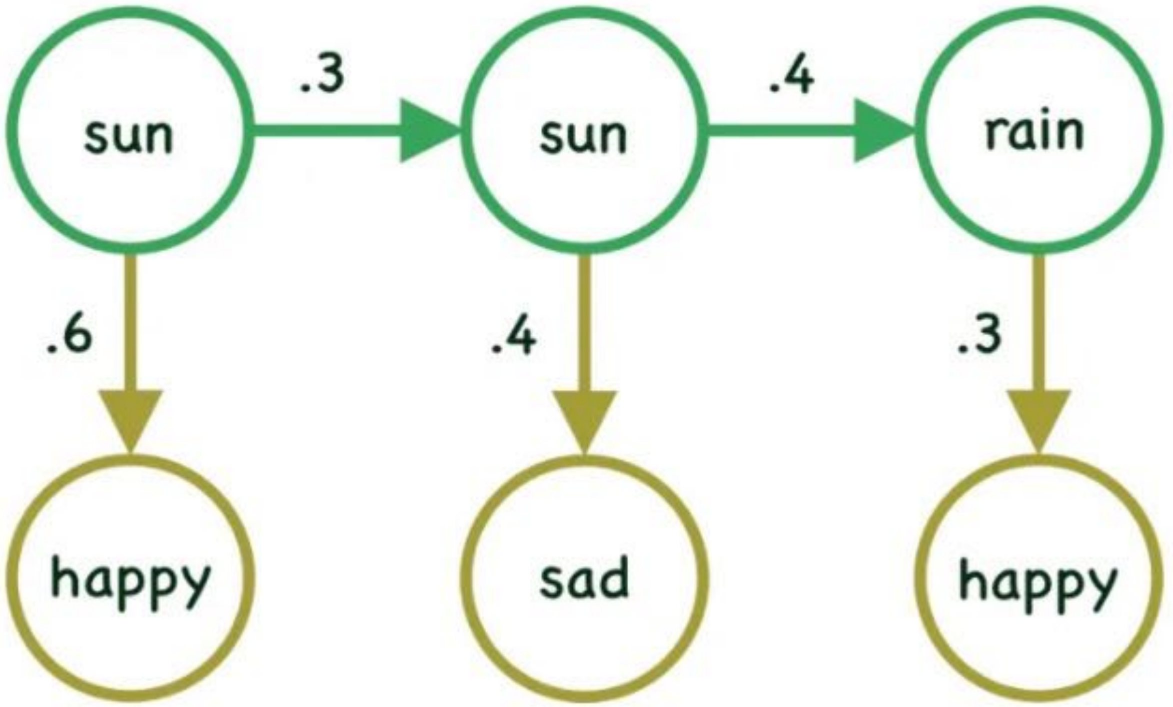
Motivation

Much of our data is inherently **sequential**

scale	examples
WORLD	Natural disasters (e.g., earthquakes) Climate change
HUMANITY	Stock market Viral outbreaks
INDIVIDUAL PEOPLE	Speech recognition Machine Translation (e.g., English -> French) Cancer treatment

Approach

Traditional, pre-deep learning models included HMMs and CRFs.

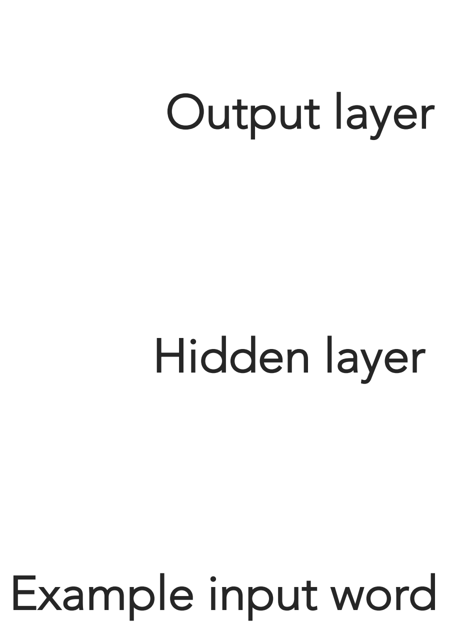


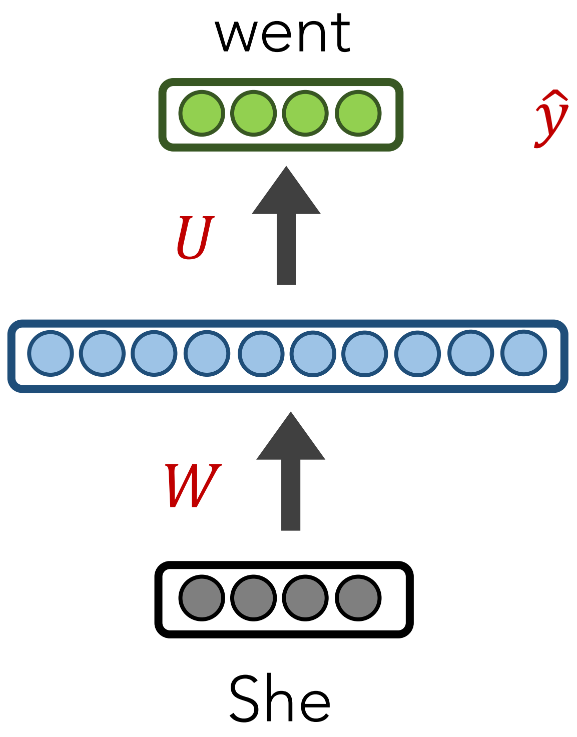
$$\begin{aligned} p(V^T, S^T) &= p(V^T|S^T)p(S^T) \\ &= \prod_{t=1}^{T} p(v(t)|s(t)) \prod_{t=1}^{T} p(s(t)|s(t-1)) \end{aligned}$$

Image: http://www.adeveloperdiary.com/data-science/machine-learning/forward-and-backward-algorithm-in-hidden-markov-model/

RNN

IDEA: for every individual input, output a prediction





$$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$$

$$h = f(Wx + b_1)$$

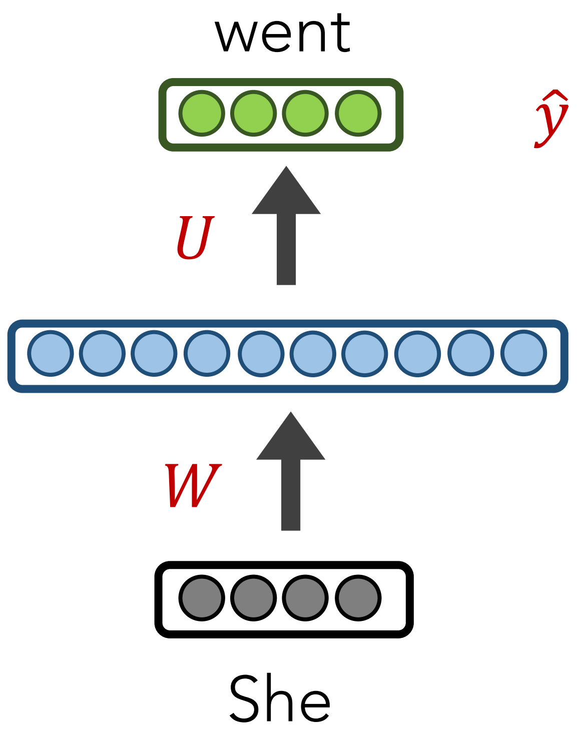
$$x = x_1$$

single word embedding

RNN

IDEA: for every individual input, output a prediction

Let's use the previous hidden state, too



Output layer $\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

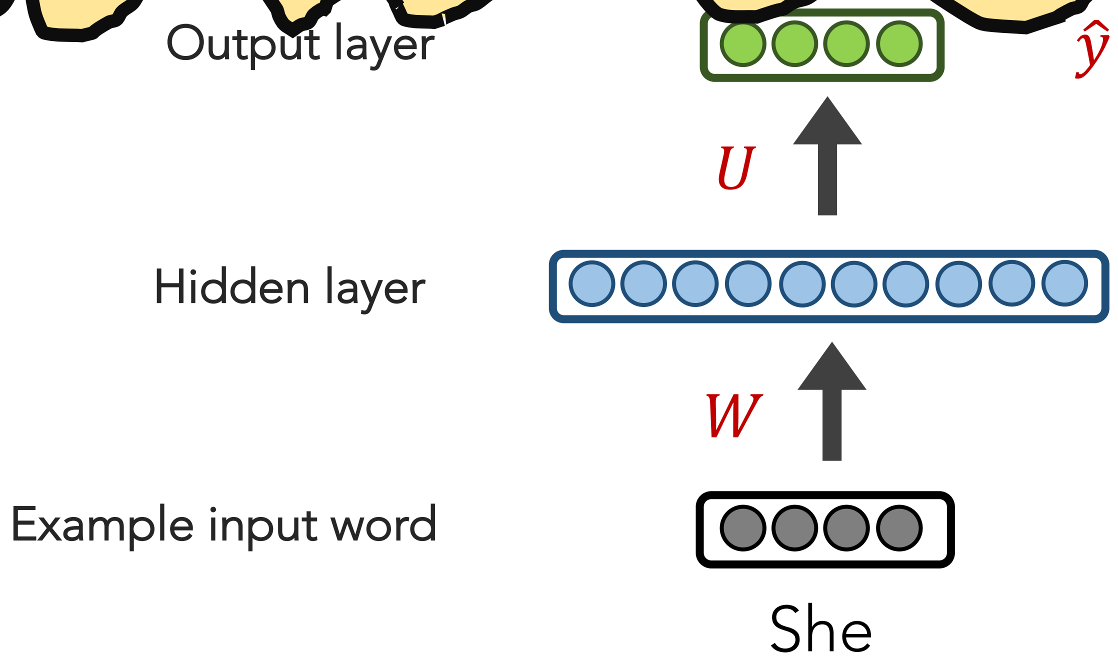
Hidden layer $h = f(Wx + b_1)$

Example input word $x = x_1$

single word embedding

RNN

Let's use the previous hidden state, too



$\hat{y} = \text{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

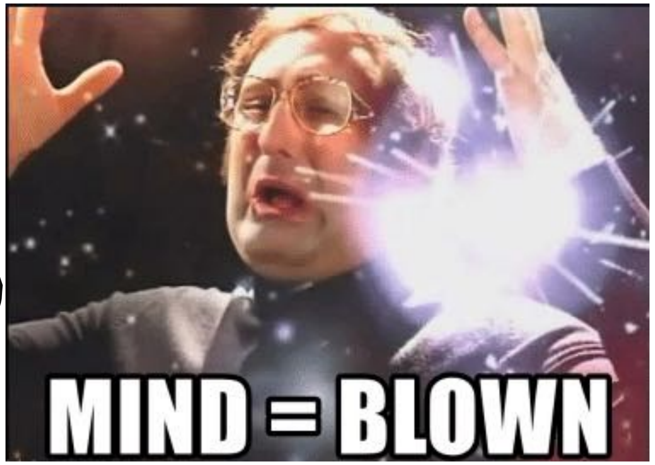
$h = f(Wx + b_1)$

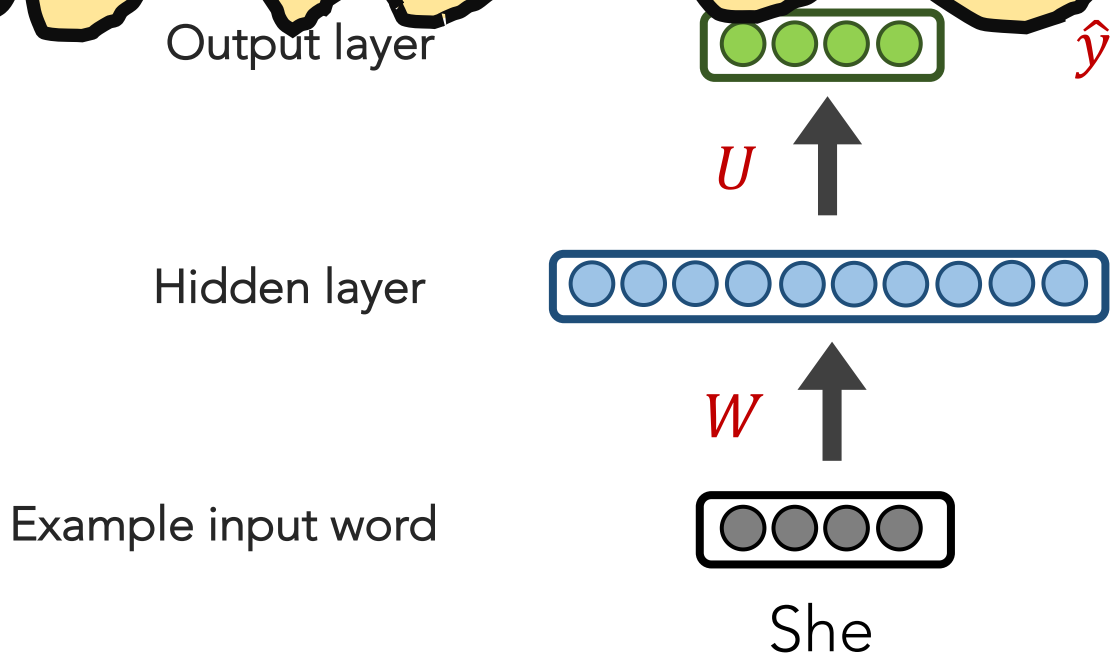
$x = x_1$

single word embedding

RNN

Let's use the previous hidden state, too





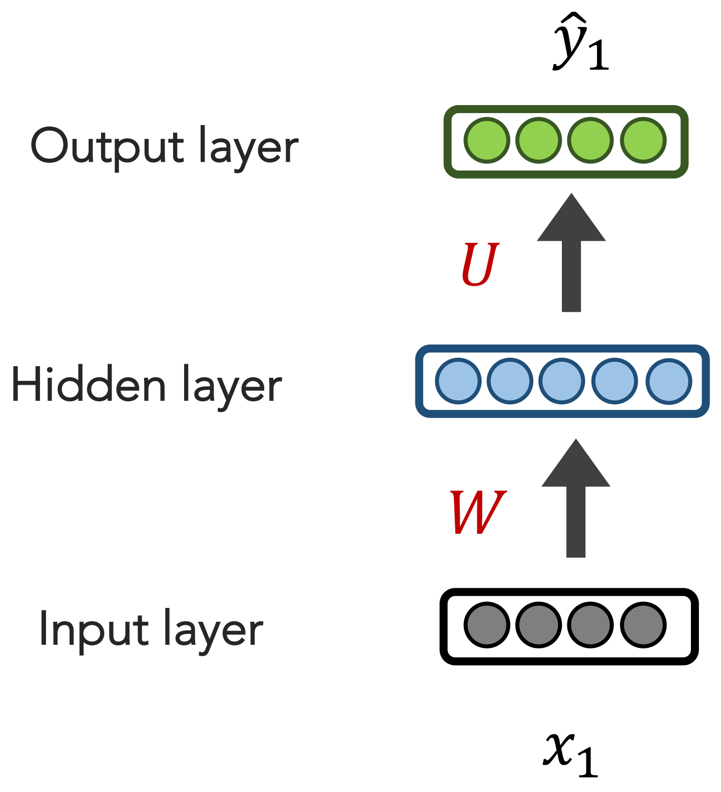
$\hat{y} = \mathrm{softmax}(Uh + b_2) \in \mathbb{R}^{|V|}$

$h = f(Wx + b_1)$

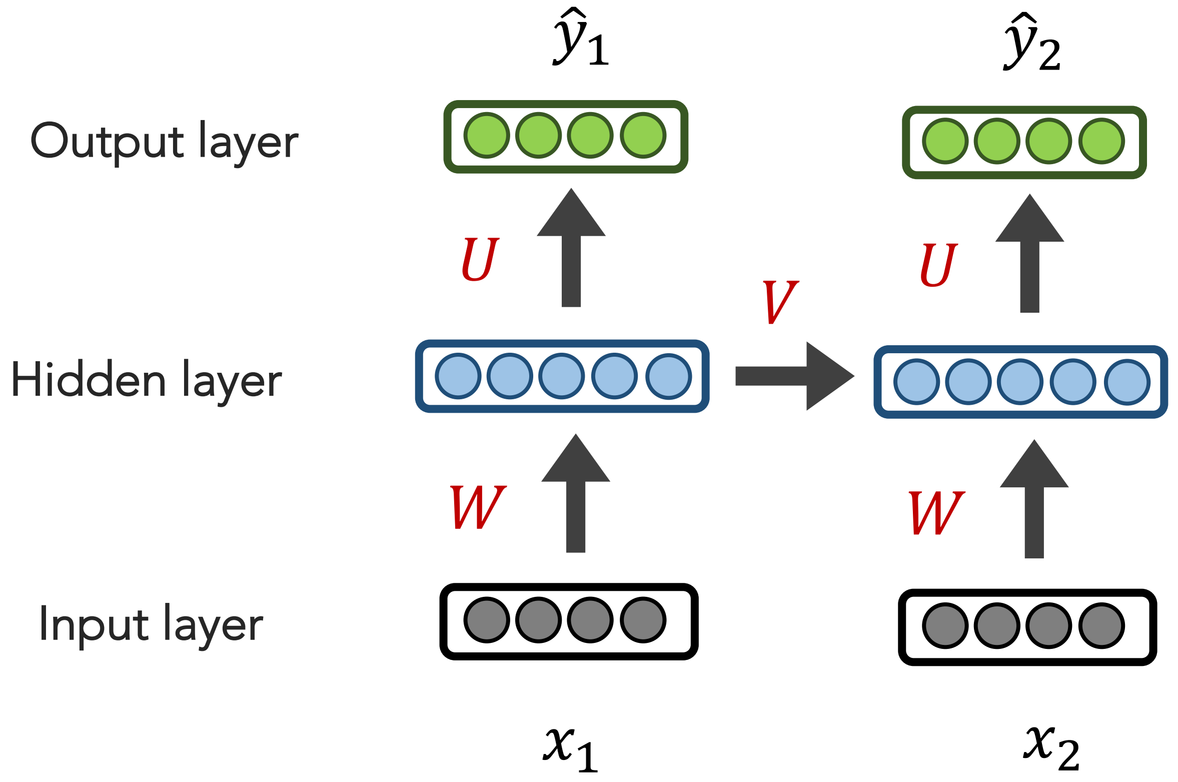
$x = x_1$

single word embedding

RNN

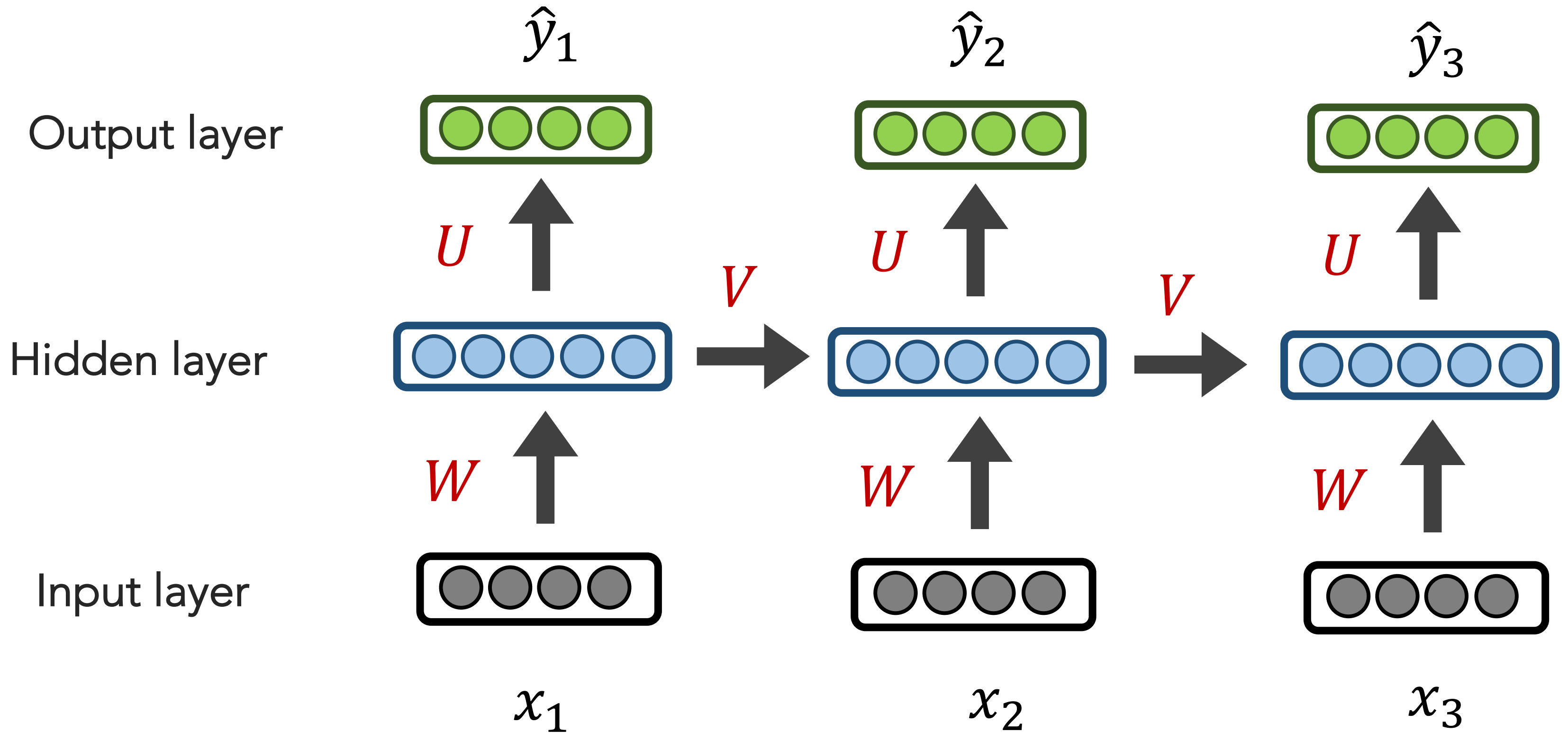


RNN

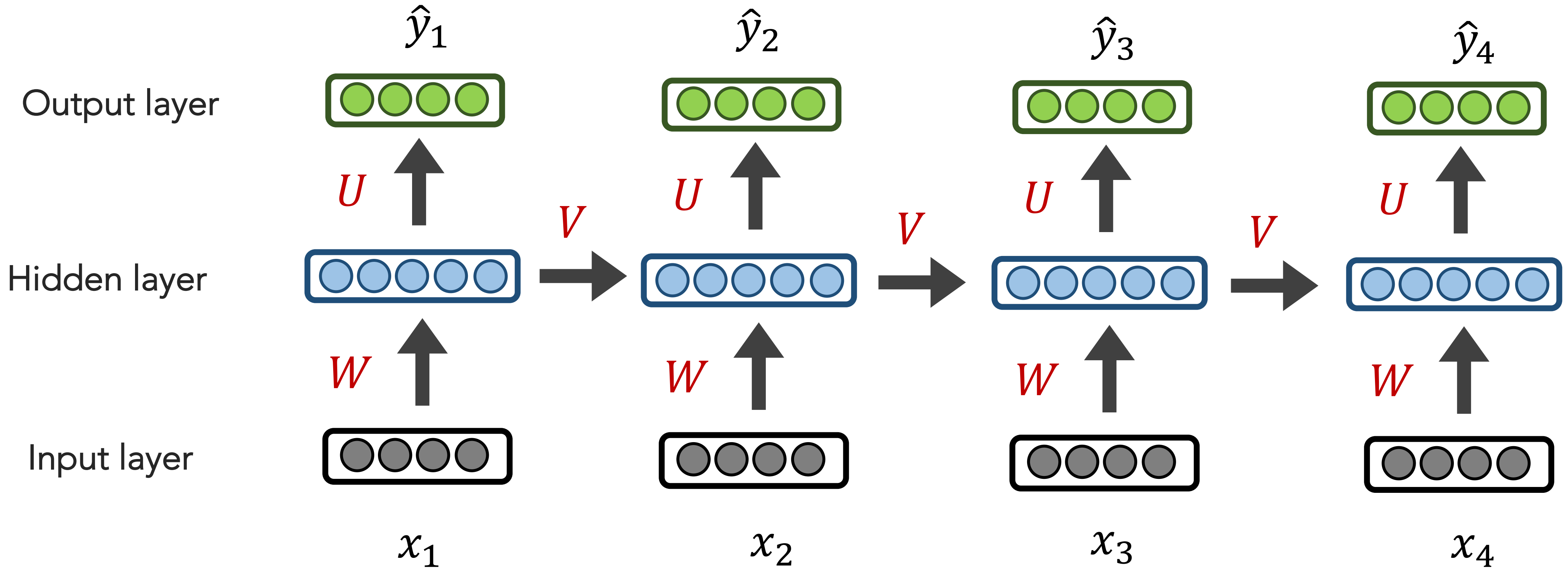


$\hat{y}_1$
$\hat{y}_2$
Output layer
U
U
V
Hidden layer
W
W
Input layer
x_1
x_2

RNN

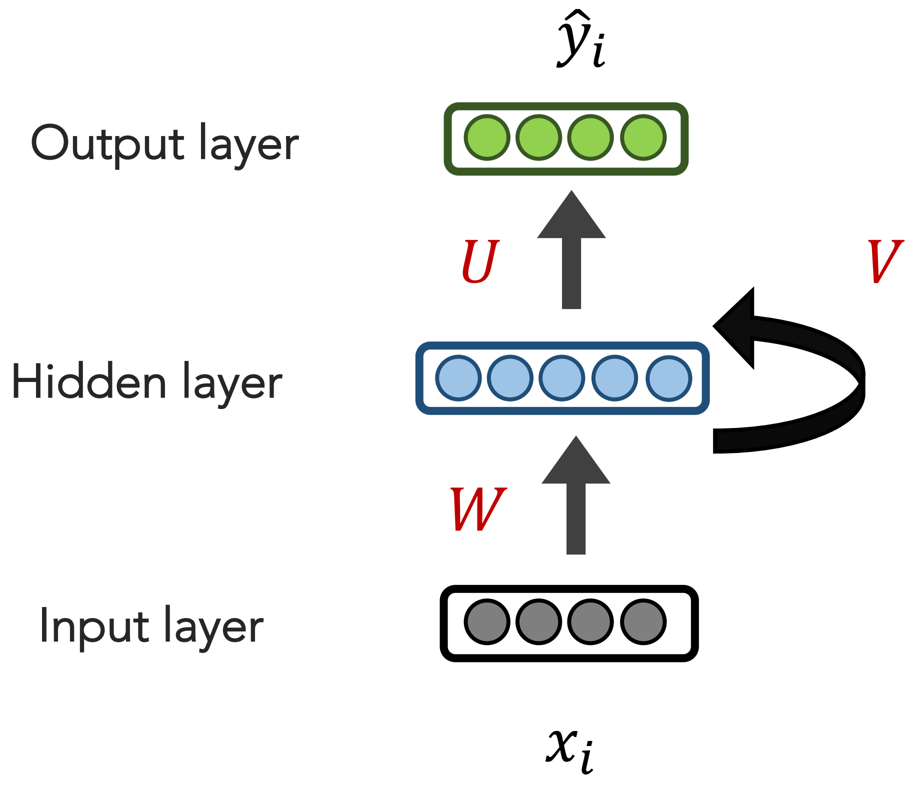

$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
Output layer
U
U
U
V
V
Hidden layer
W
W
W
Input layer
x_1
x_2
x_3

RNN


$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_4$
Output layer
Hidden layer
Input layer
U
V
W
x_1
x_2
x_3
x_4

RNN

Some people find this abstract view useful.



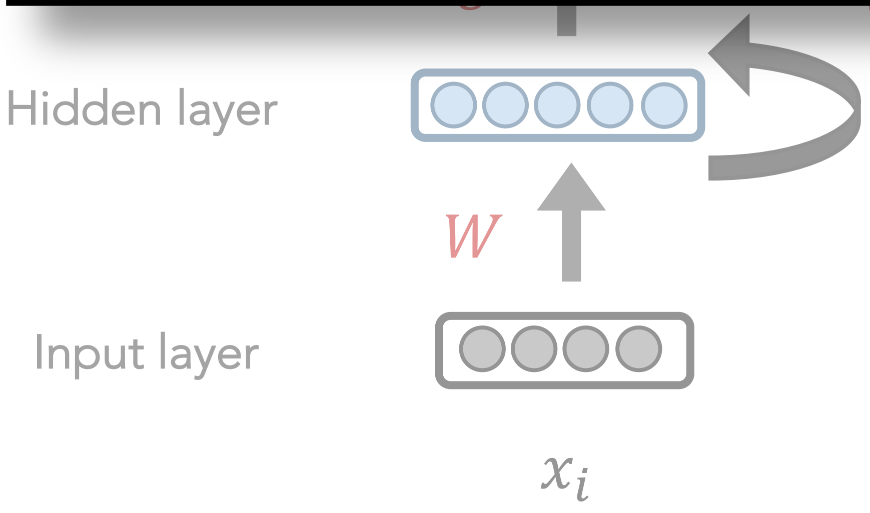
The recurrent loop V conveys that the current hidden layer is influenced by the hidden layer from the previous time step.

The initial hidden layer $\boldsymbol{h_0}$ can be initialized to 0s

RNN

Some people find this abstract view useful

Definition: an RNN is any neural net that has a non-linear combination of the recurrent state (e.g., hidden layer) and the input



The recurrent loop V conveys that the current hidden layer is influenced by the hidden layer from the previous time step.

RNN

Some people find this abstract view useful

Definition: an RNN is any neural net that has a non-linear combination of the recurrent state (e.g., hidden layer) and the input

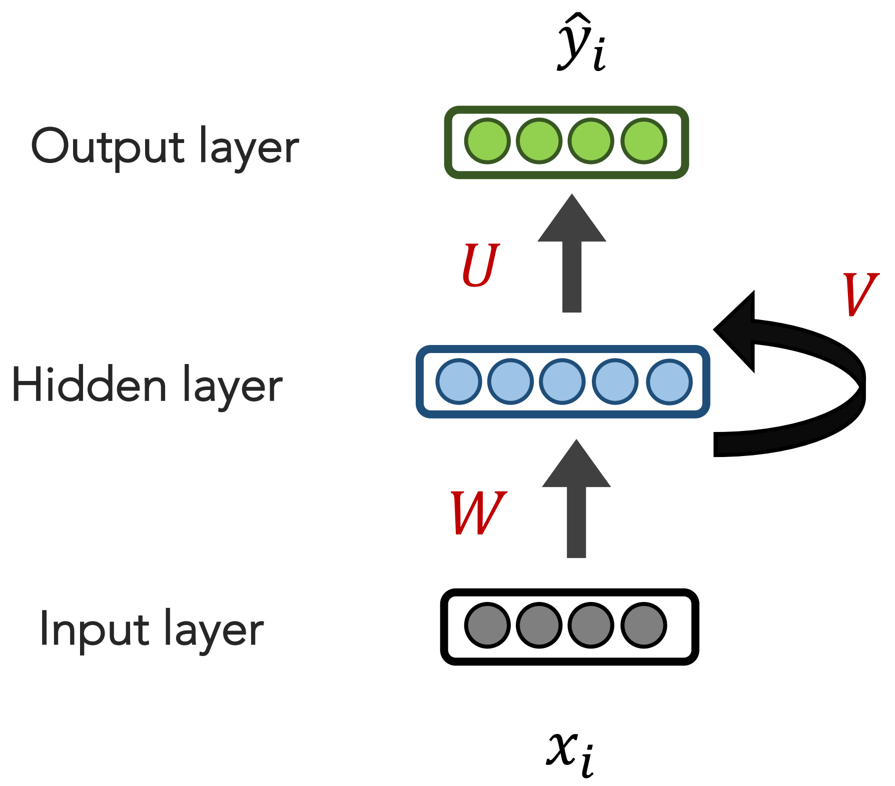
NOTE: The Embedding layer for all of our NN's never has a non-linear activation. Why?

hat the
ed by the
time step.

x_i

RNN

What exactly are we learning?

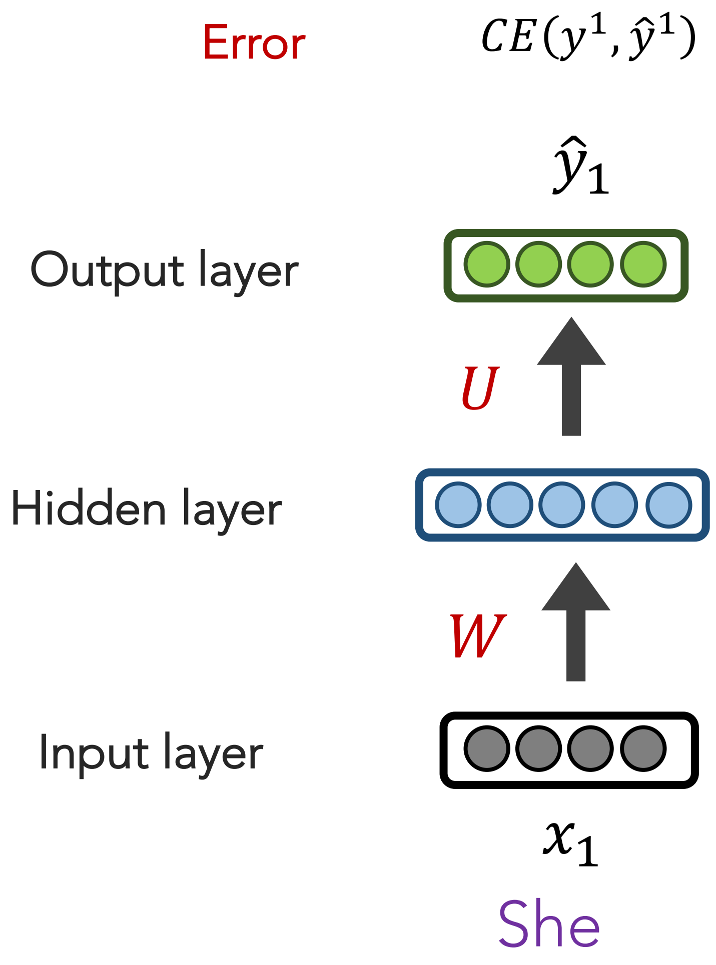


CHALKBOARD EXERCISE: Let's write out all weight matrices and their sizes.

RNN

Training Process

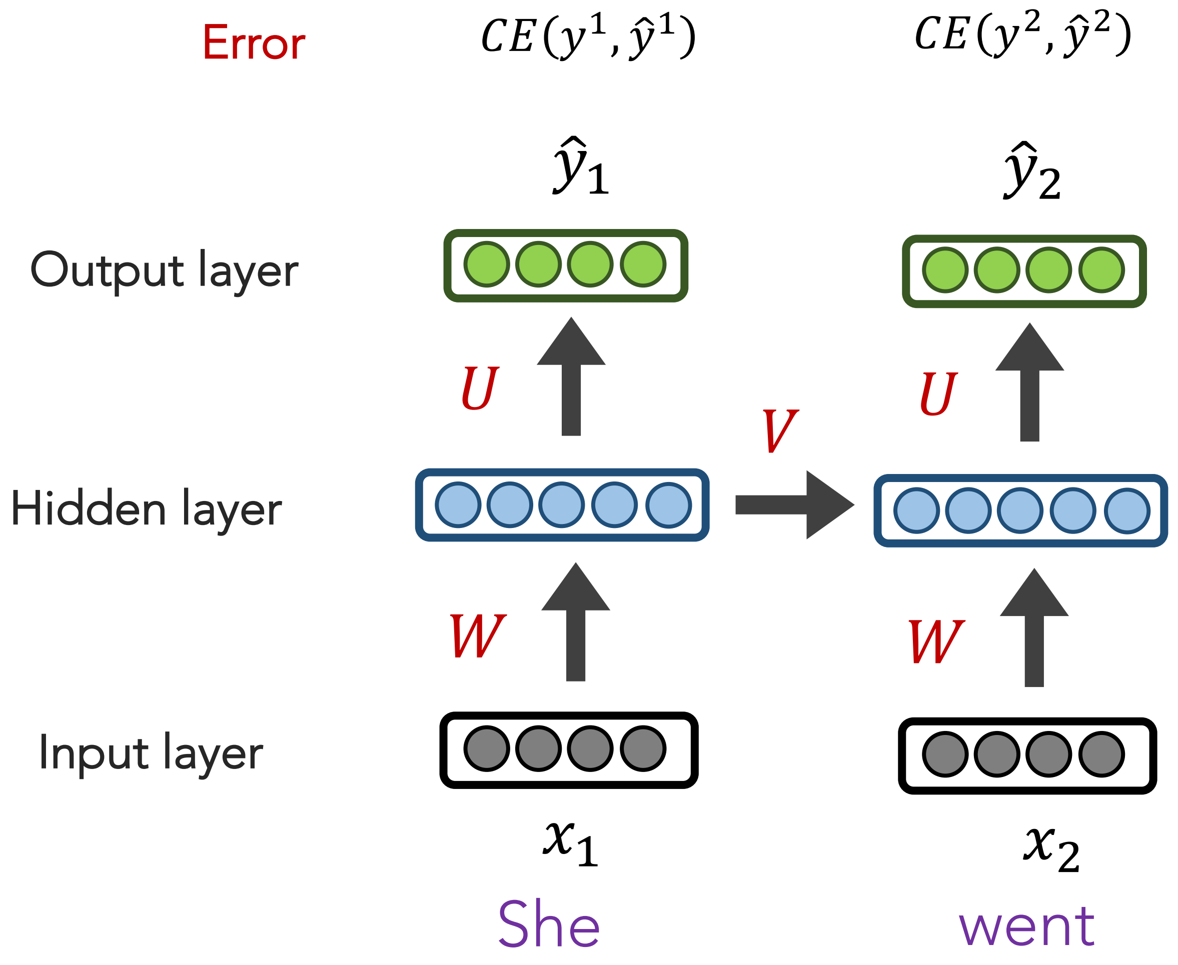
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN

Training Process

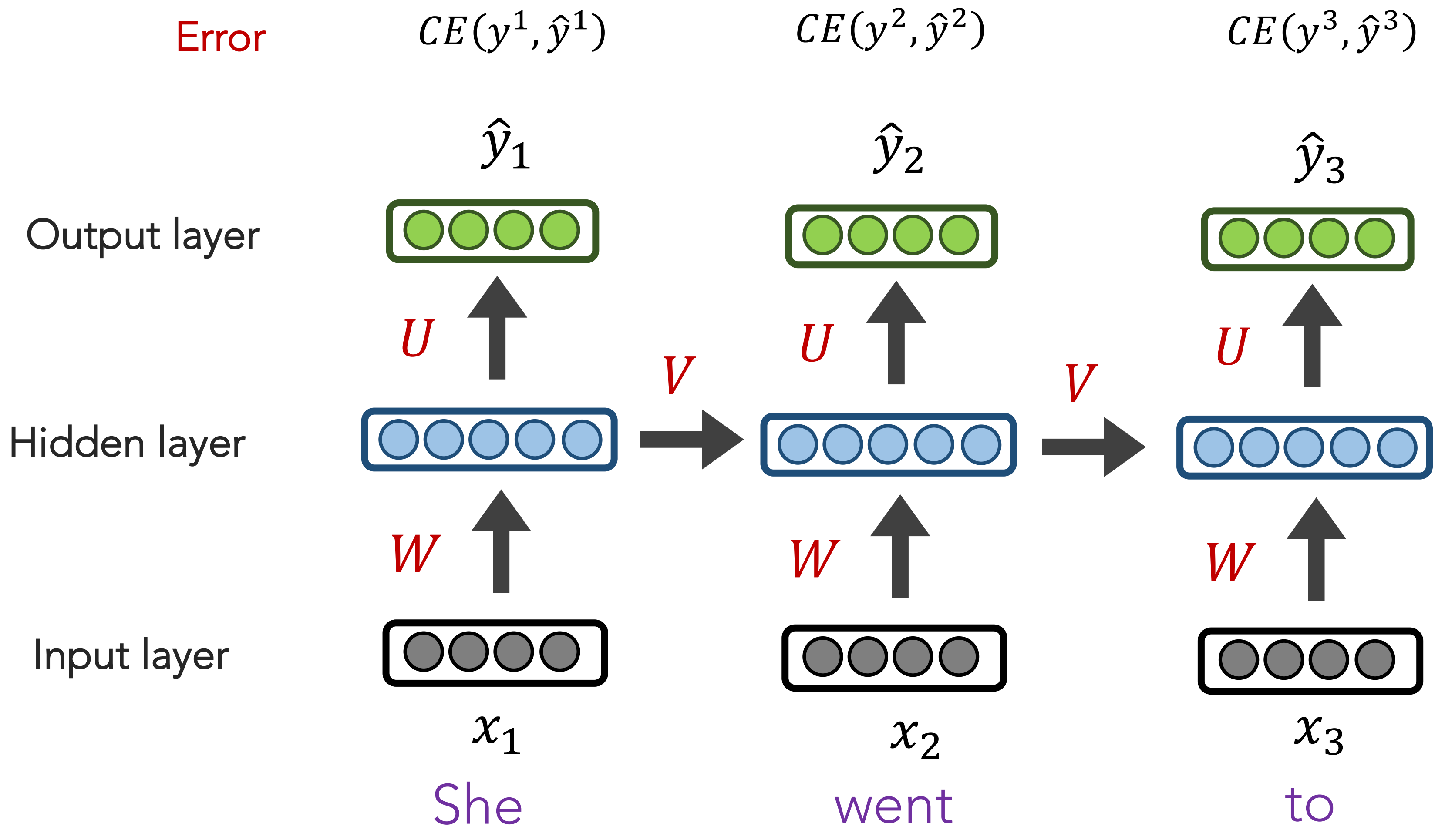
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN

Training Process

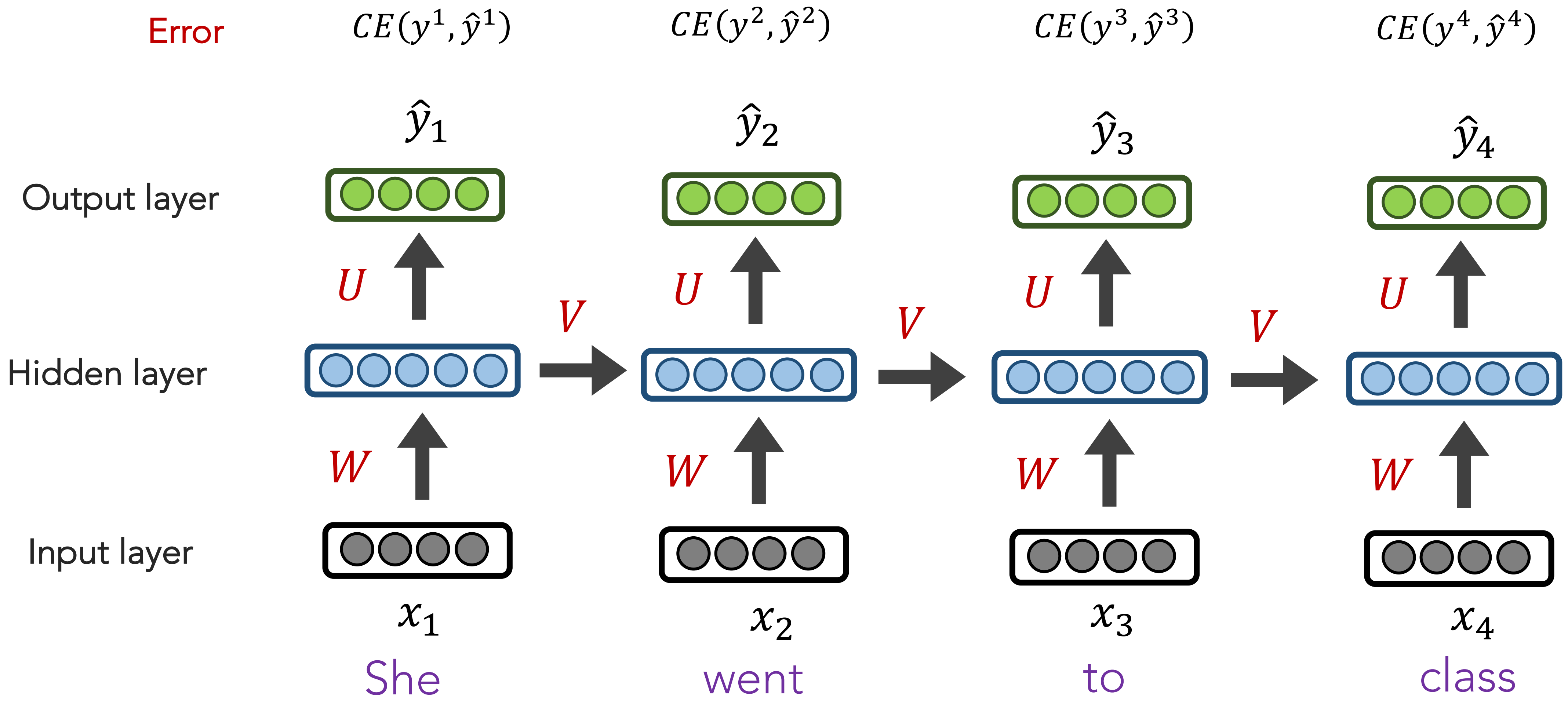
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN

Training Process

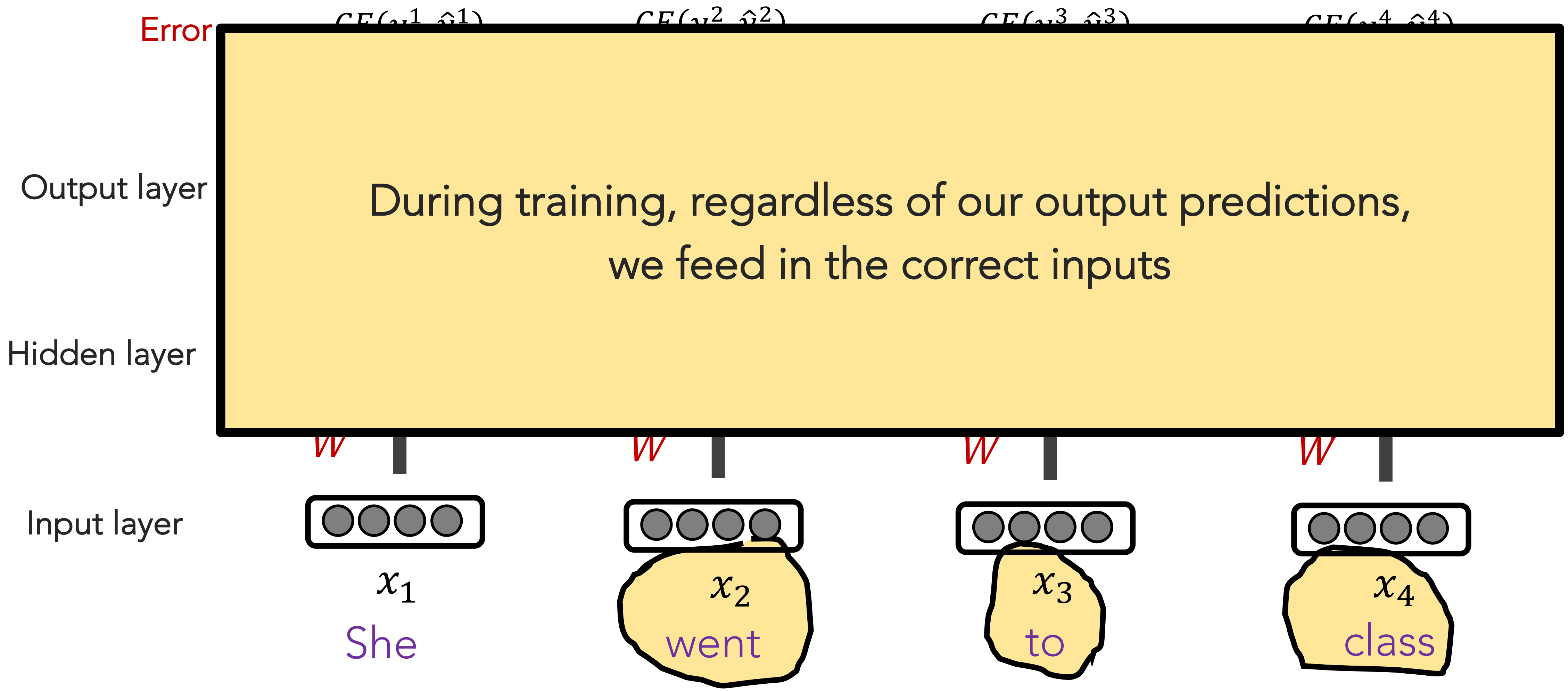
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y^i_w \log(\hat{y}^i_w)$$



RNN

Training Process

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

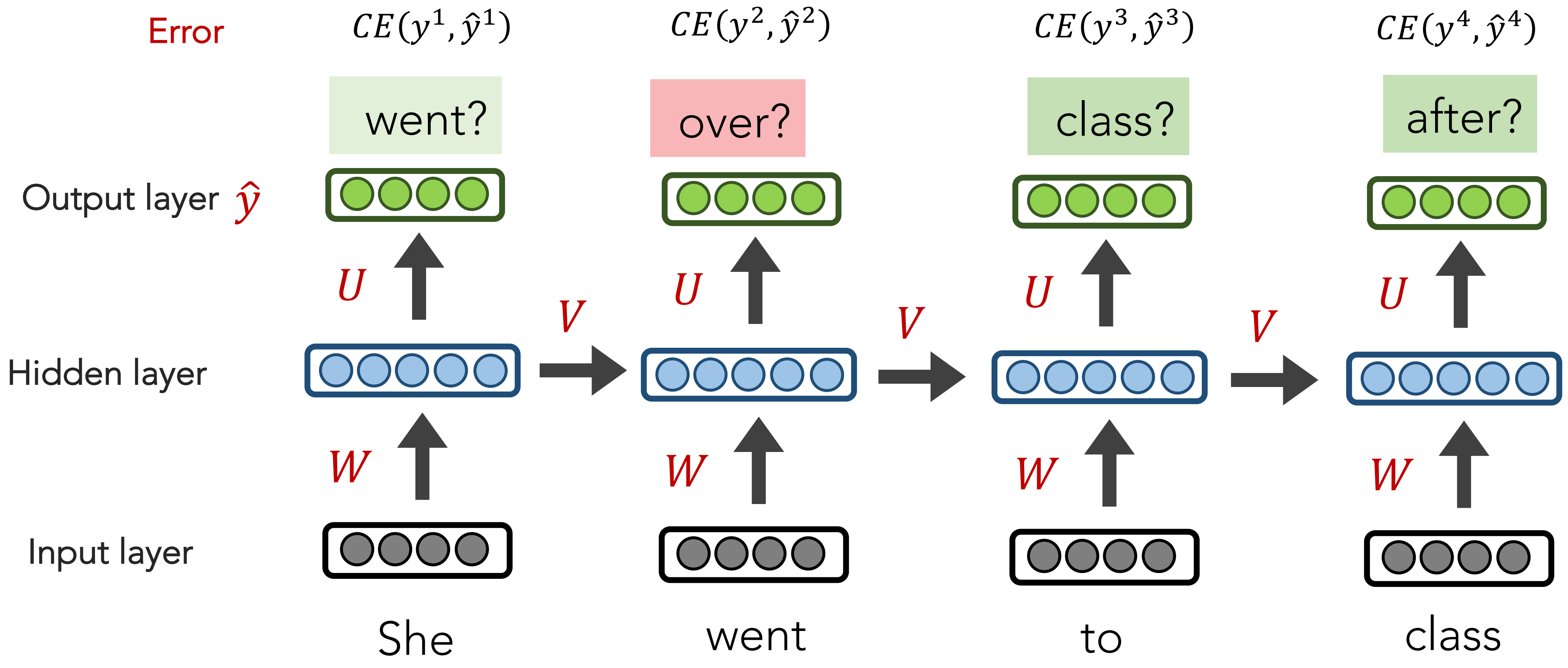


During training, regardless of our output predictions, we feed in the correct inputs

RNN

Training Process

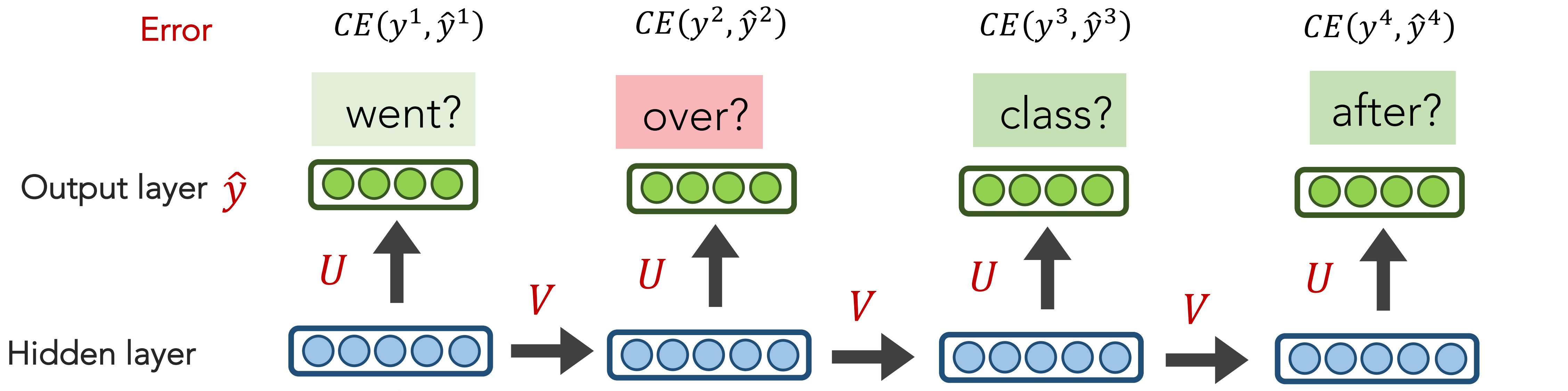
$$CE(y^i, \hat{y}^i) = -\sum_{w \in V} y_w^i \log(\hat{y}_w^i)$$



RNN

Training Process

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



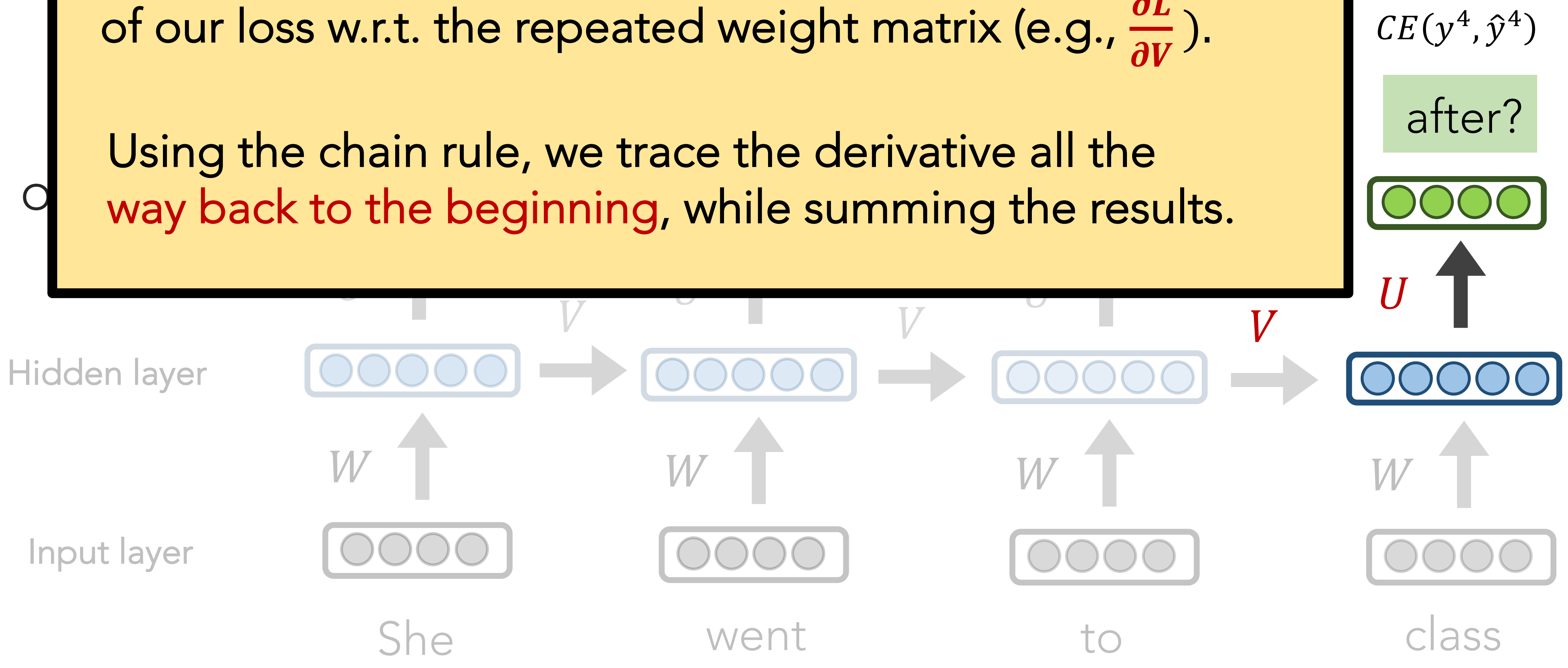
Our total loss is simply the average loss across all T time steps

RNN

Training Details

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

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



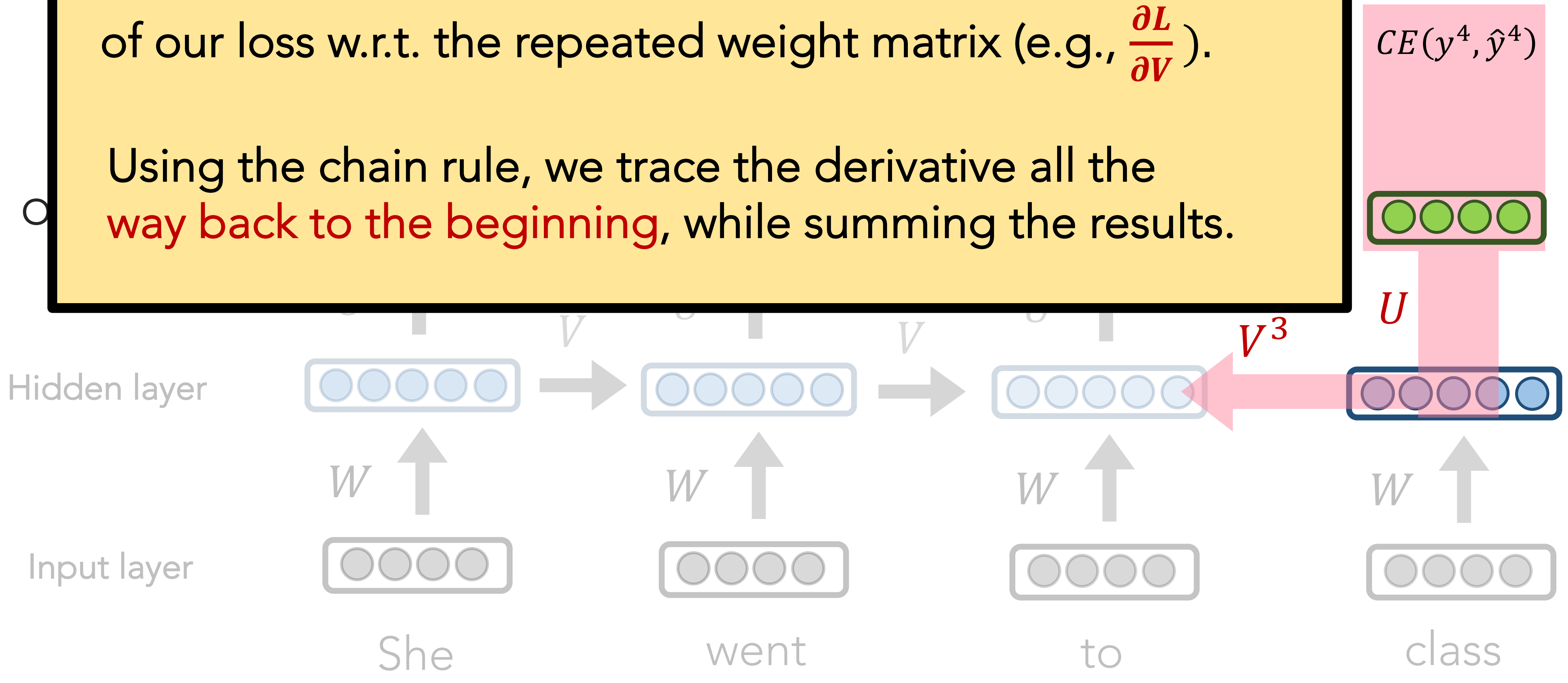
RNN

Training Details

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

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

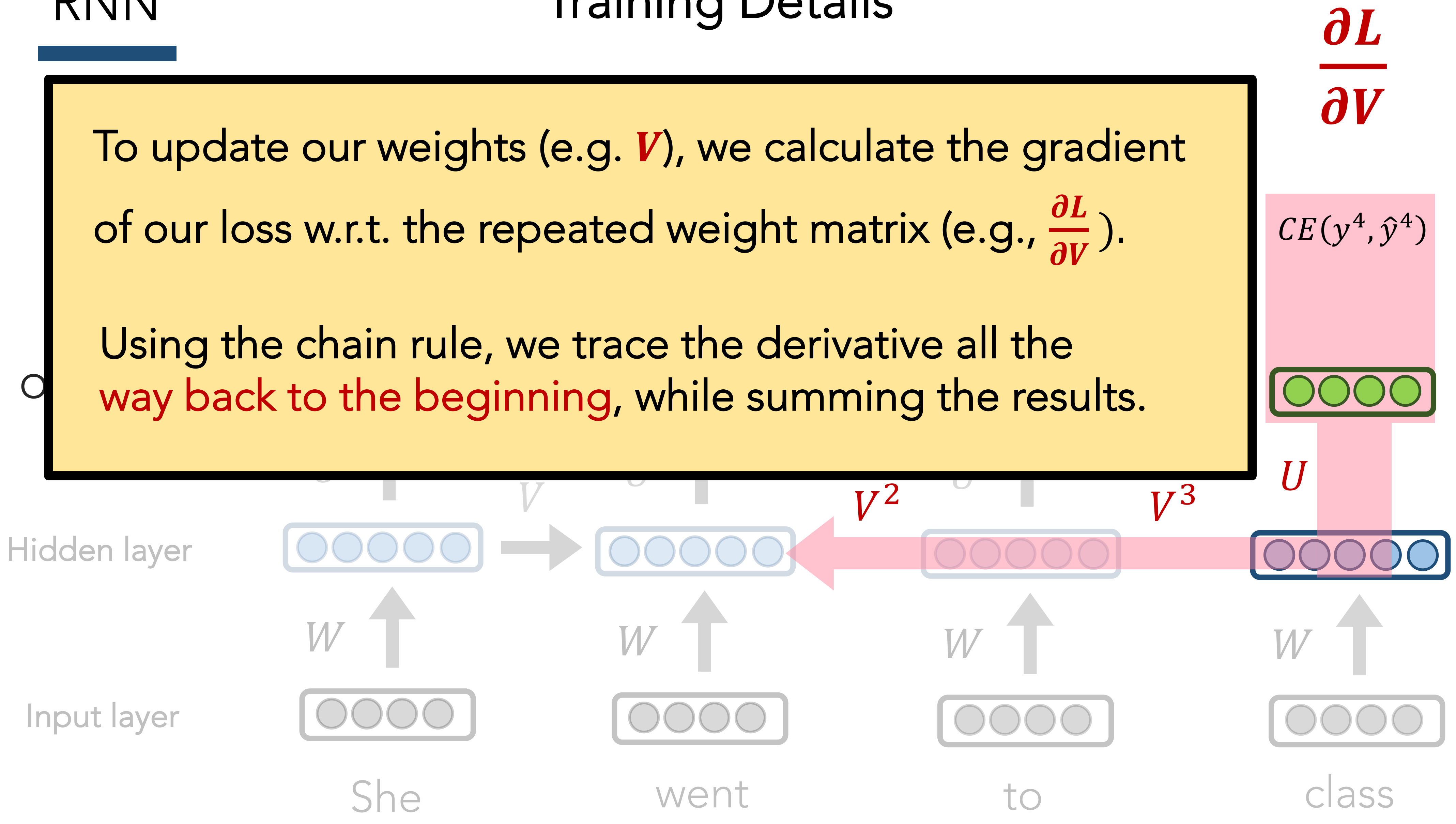
$\frac{\partial L}{\partial V}$



Training Details

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

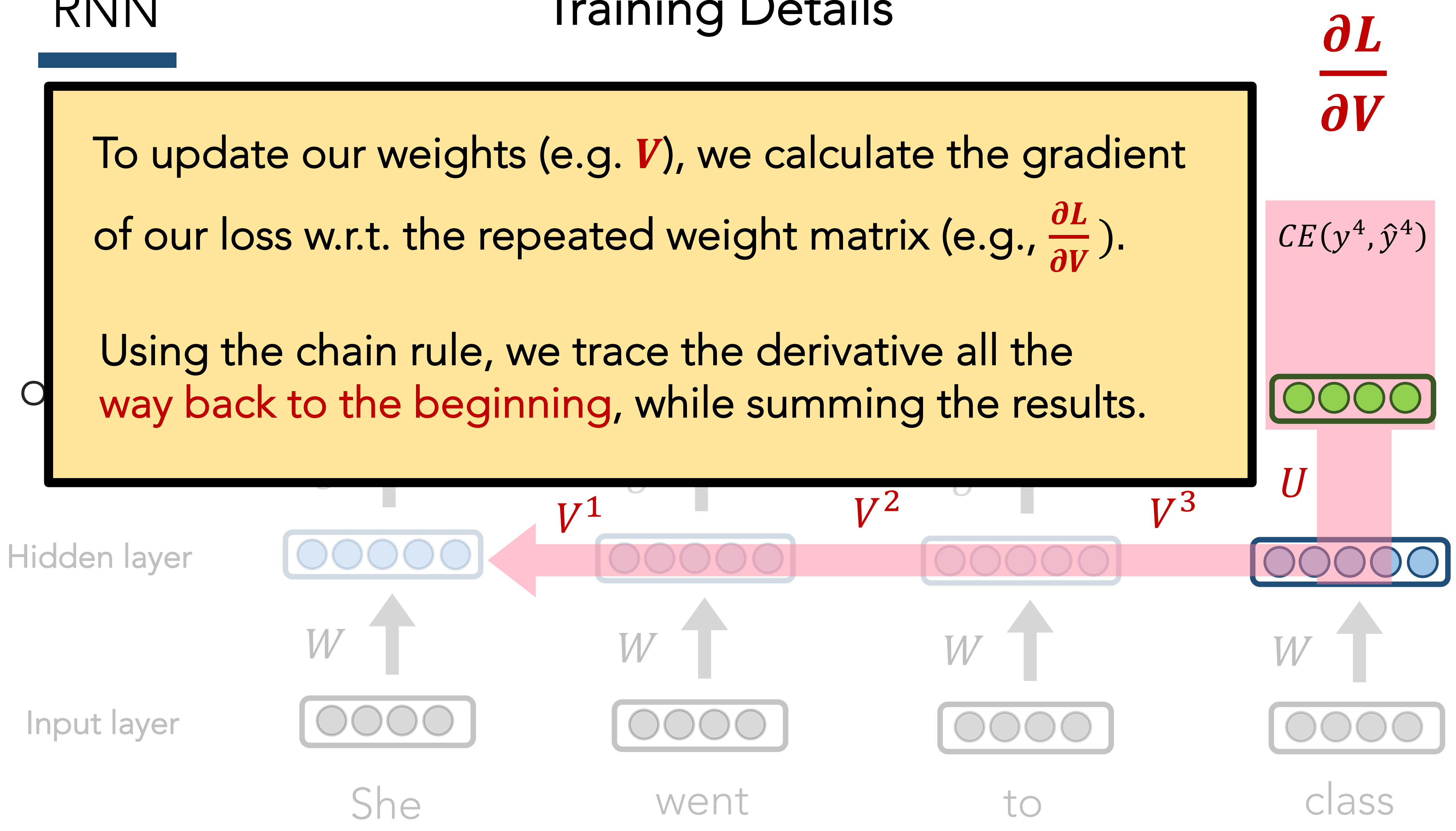
Using the chain rule, we trace the derivative all the way back to the beginning, while summing the results.



Training Details

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

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



Training Details

- This backpropagation through time (BPTT) process is **expensive**
- Instead of updating after every timestep, we tend to do so every T steps (e.g., every <u>sentence</u> or <u>paragraph</u>)
- This isn't equivalent to using only a window size T (a la n-grams) because we still have 'infinite memory'

RNN: Generation

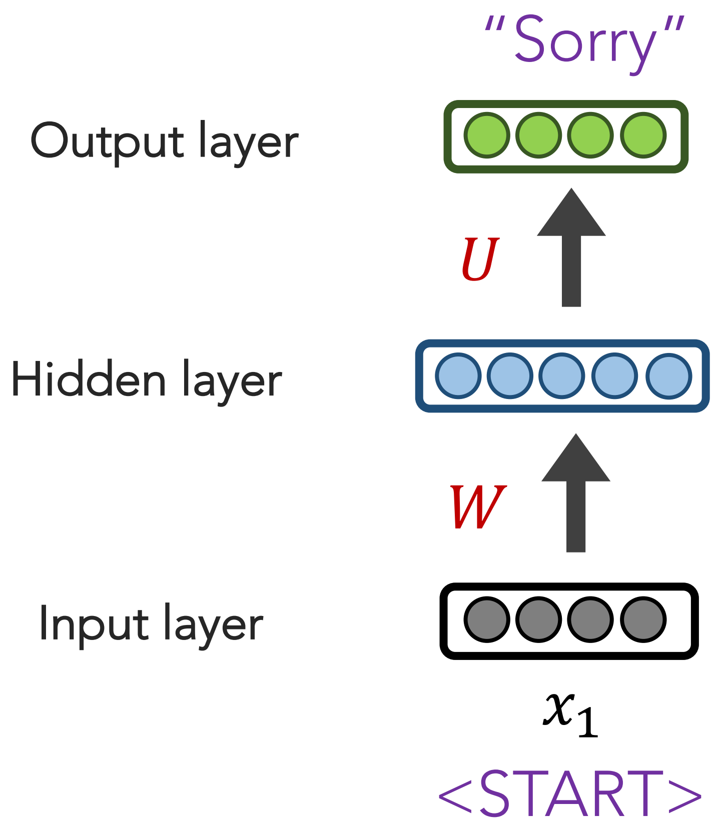
We can generate the most likely **next** event (e.g., word) by sampling from $\hat{\boldsymbol{y}}$

Continue until we generate <EOS> symbol.

RNN: Generation

We can generate the most likely **next** event (e.g., word) by sampling from $\hat{y}$

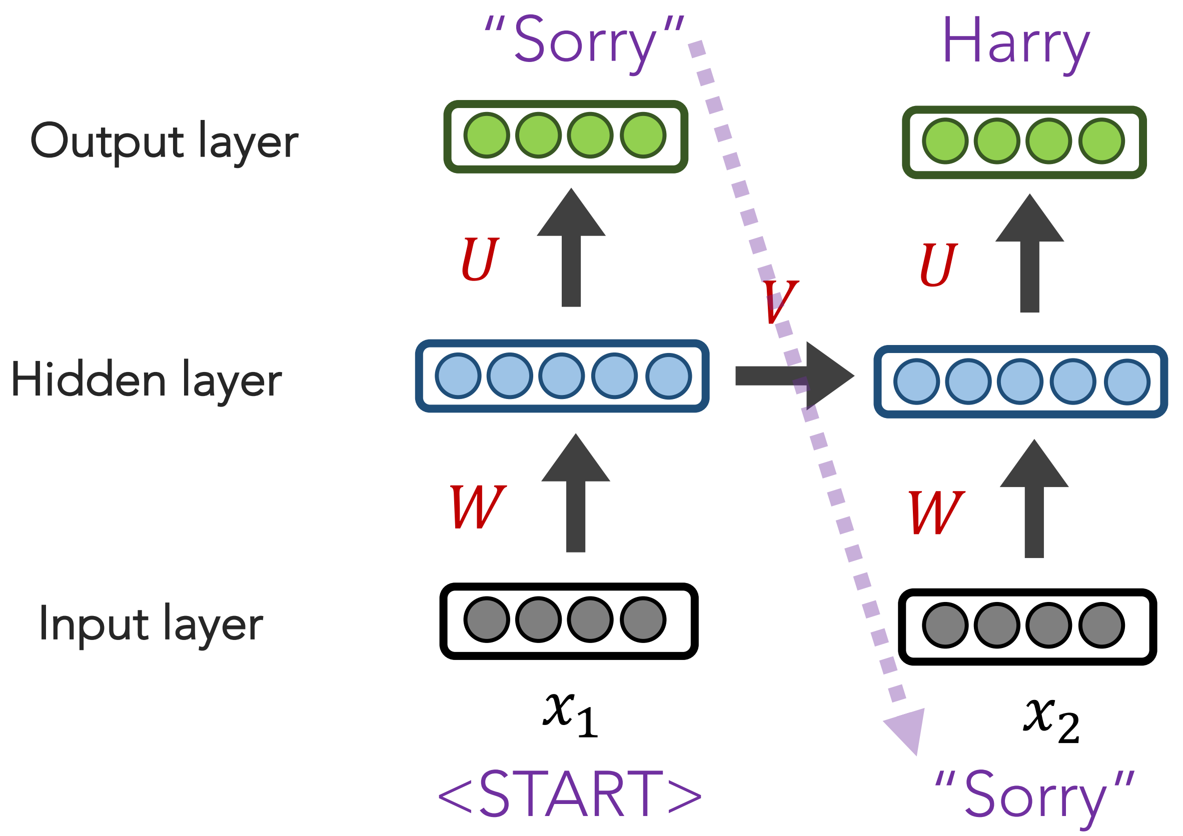
Continue until we generate <EOS> symbol.



RNN: Generation

We can generate the most likely **next** event (e.g., word) by sampling from $\hat{\boldsymbol{y}}$

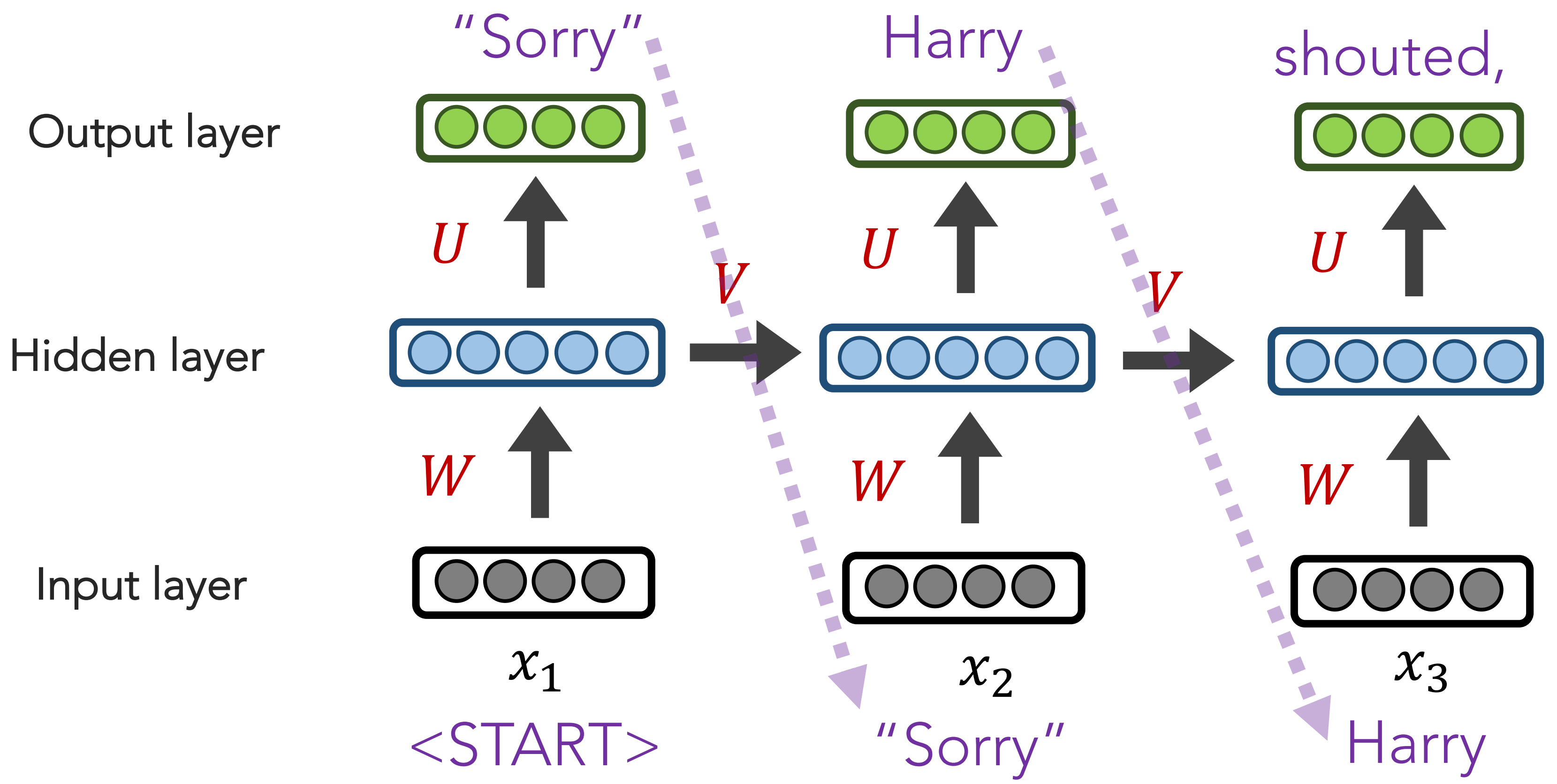
Continue until we generate <EOS> symbol.



RNN: Generation

We can generate the most likely **next** event (e.g., word) by sampling from $\hat{\boldsymbol{y}}$

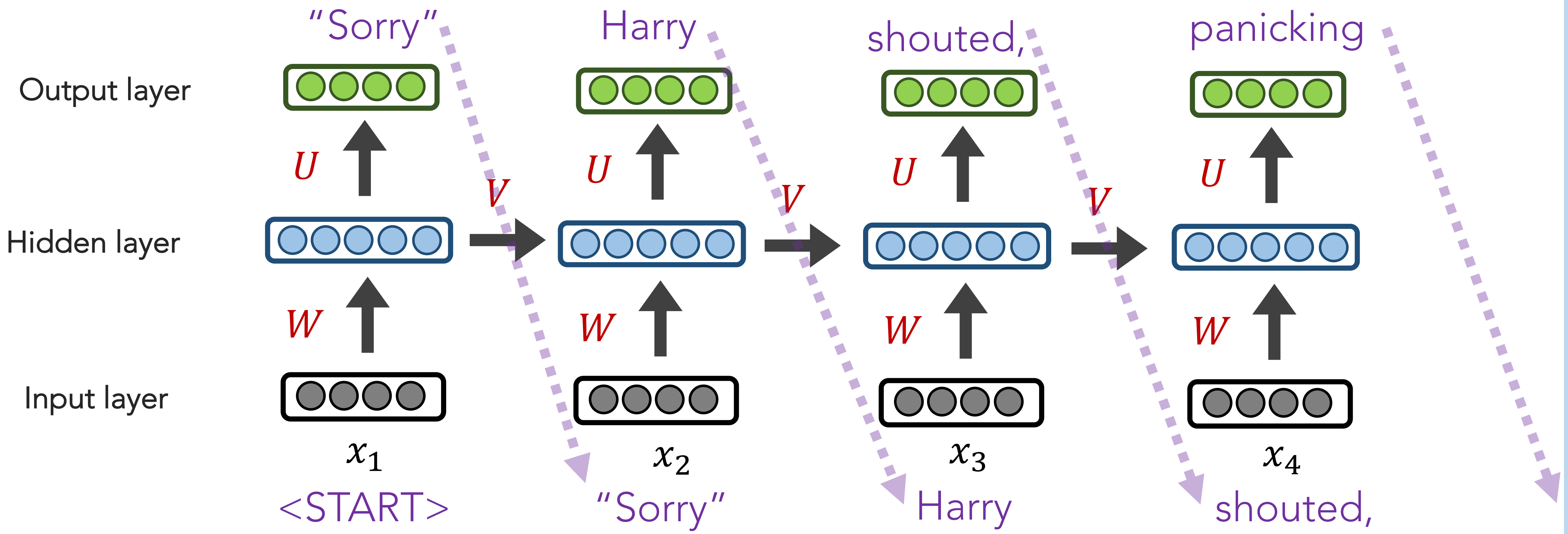
Continue until we generate <EOS> symbol.



RNN: Generation

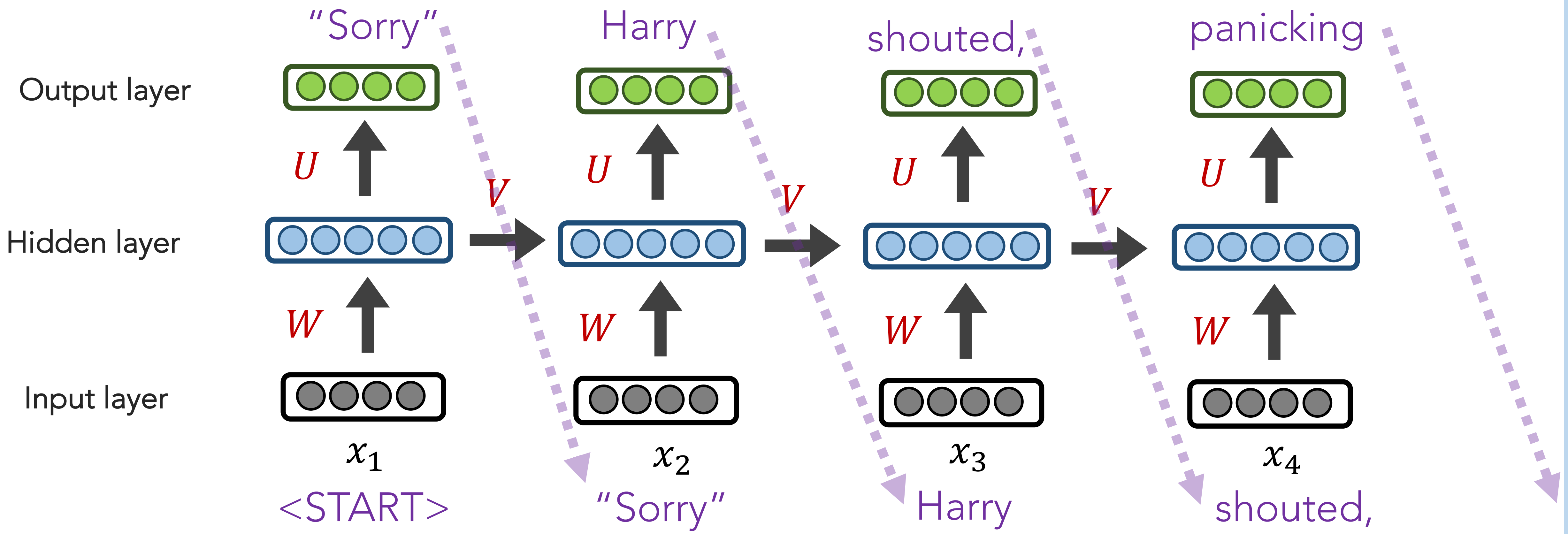
We can generate the most likely **next** event (e.g., word) by sampling from $\hat{y}$

Continue until we generate <EOS> symbol.



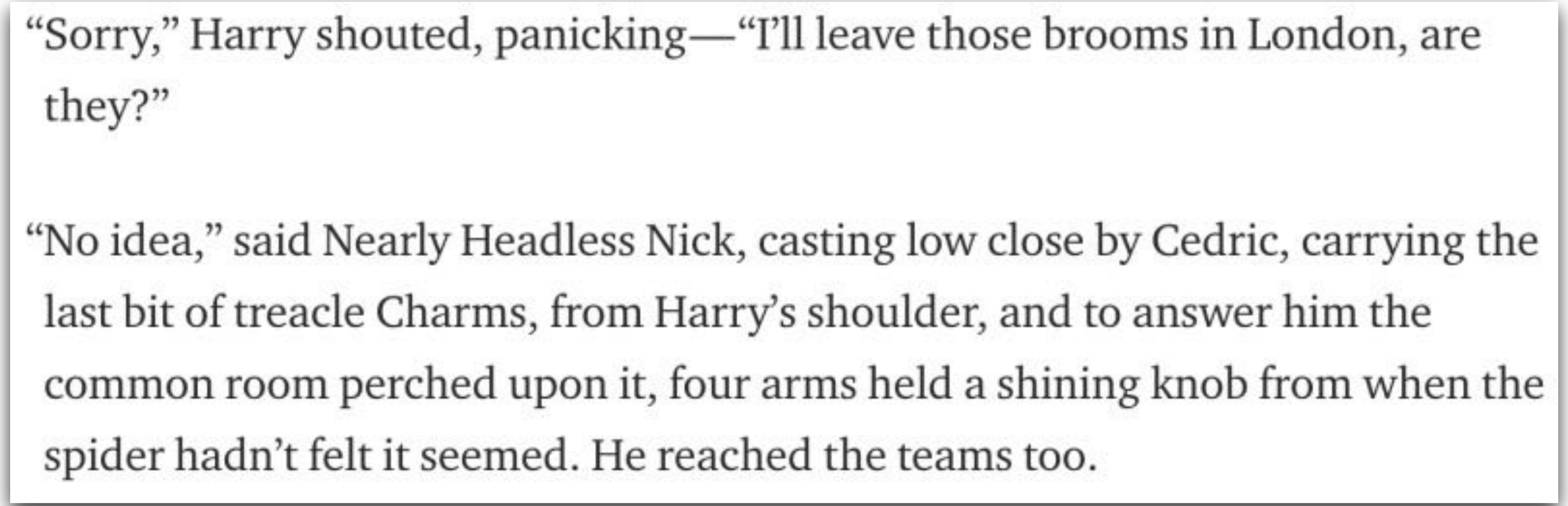
RNN: Generation

NOTE: the same input (e.g., **"Harry"**) can easily yield different outputs, depending on the context (unlike FFNNs and n-grams).



RNN: Generation

When trained on Harry Potter text, it generates:



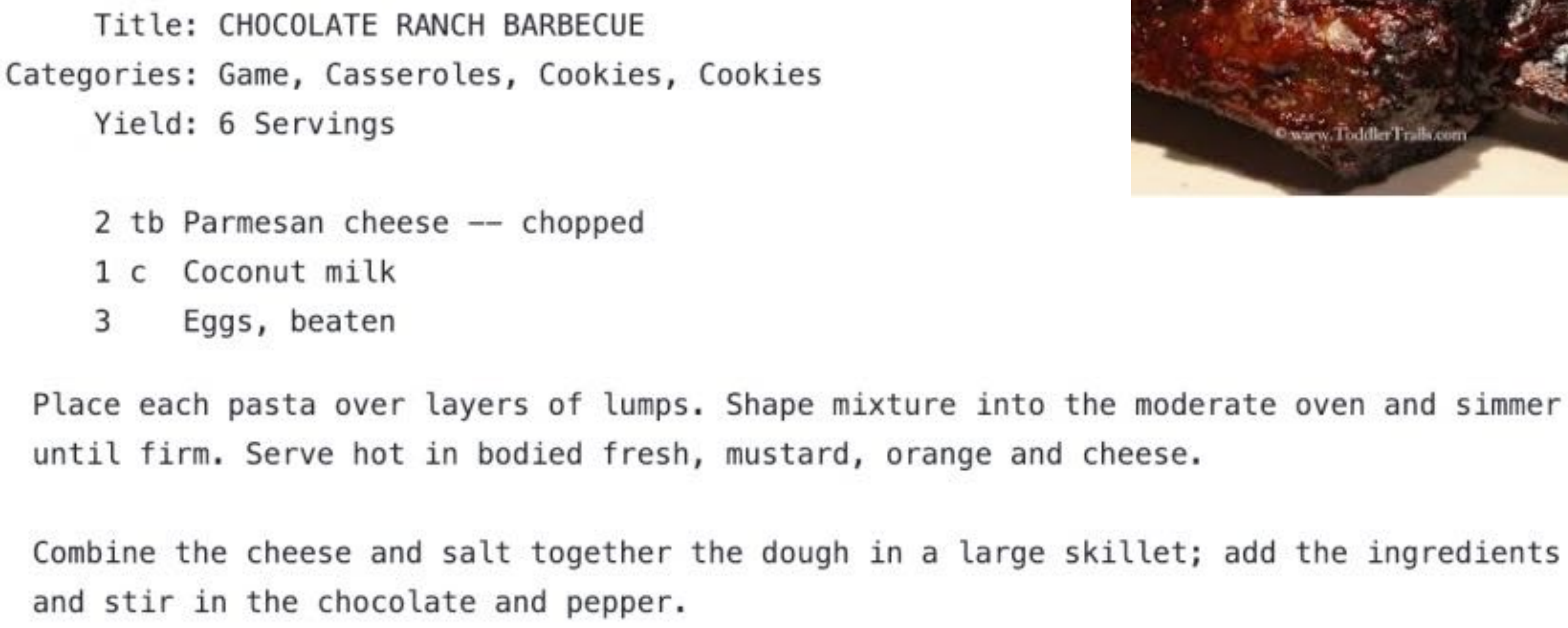
"Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"

"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

RNN: Generation

When trained on 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.
```

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

RNNs: Overview

RNN STRENGTHS?

- Can handle infinite-length sequences (not just a fixed-window)
- Has a "memory" of the context (thanks to the hidden layer's recurrent loop)
- Same weights used for all inputs, so word order isn't wonky (like FFNN)

RNN ISSUES?

- Slow to train (BPTT)
- Due to "infinite sequence", gradients can easily **vanish** or **explode**
- Has trouble actually making use of long-range context

RNNs: Overview

RNN STRENGTHS?

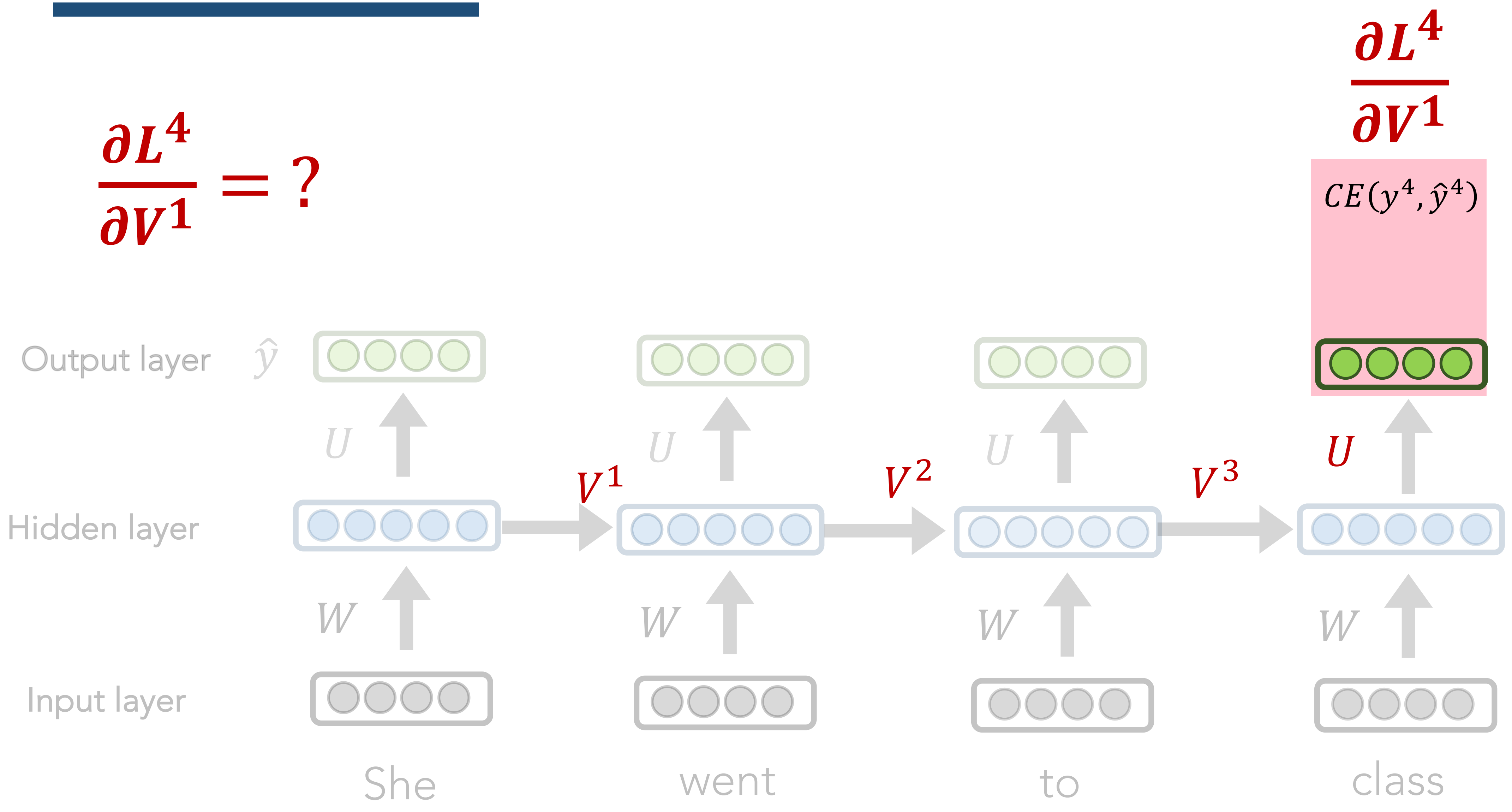
- Can handle infinite-length sequences (not just a fixed-window)
- Has a "memory" of the context (thanks to the hidden layer's recurrent loop)
- Same weights used for all inputs, so word order isn't wonky (like FFNN)

RNN ISSUES?

- Slow to train (BPTT)
- Due to "infinite sequence", gradients can easily **vanish** or **explode**
- Has trouble actually making use of long-range context

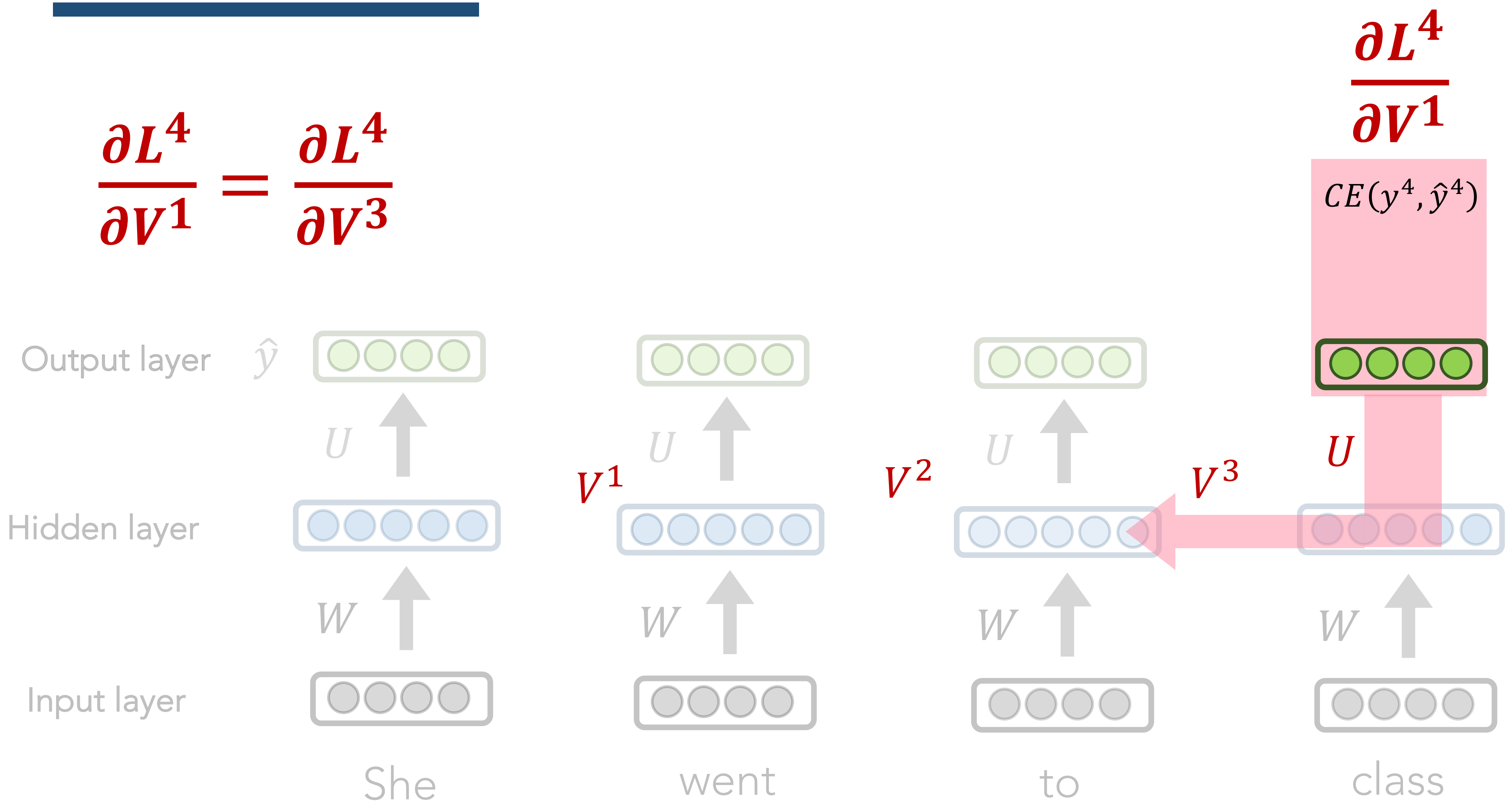
RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = ?$$



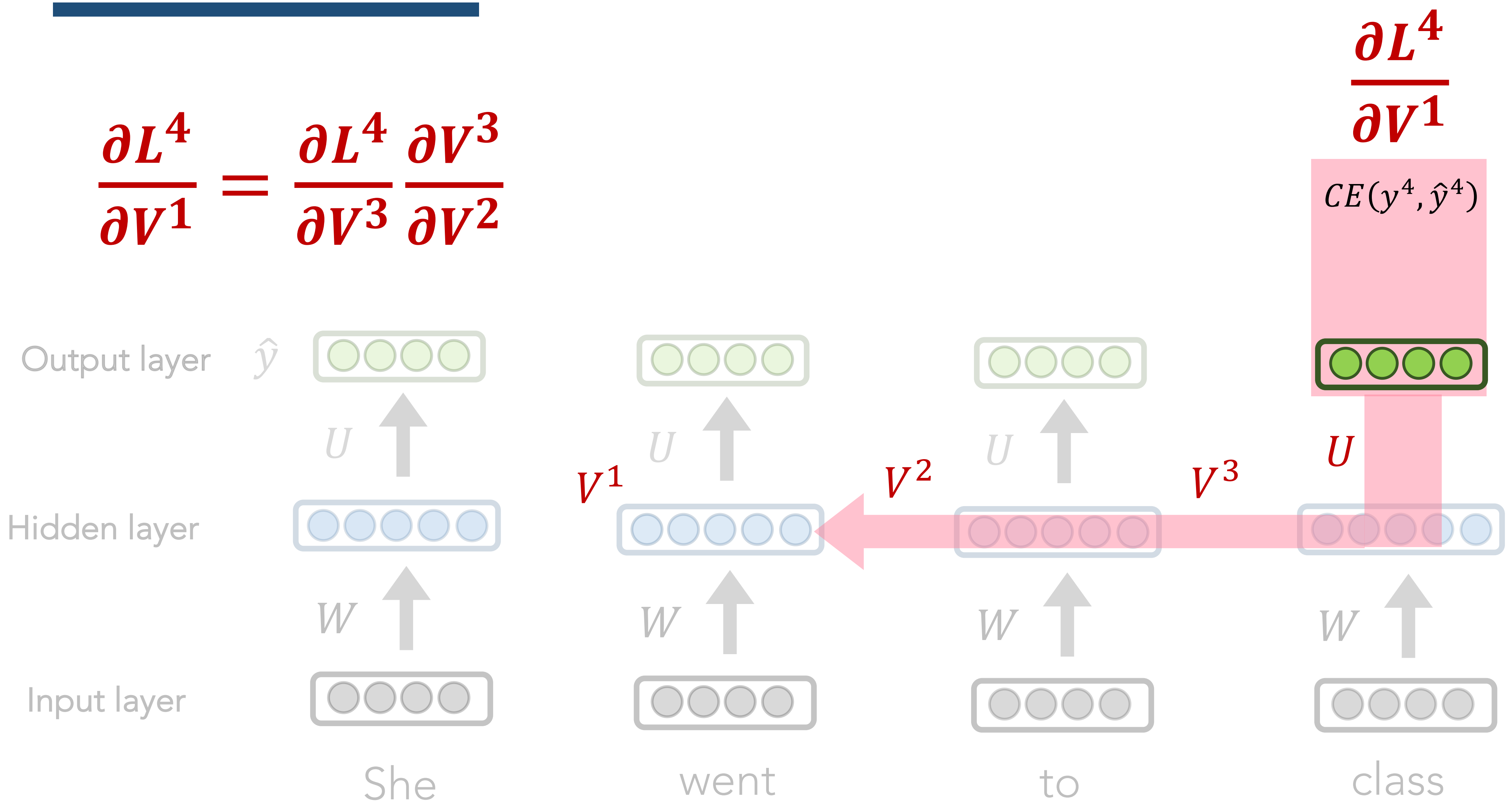
RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3}$$



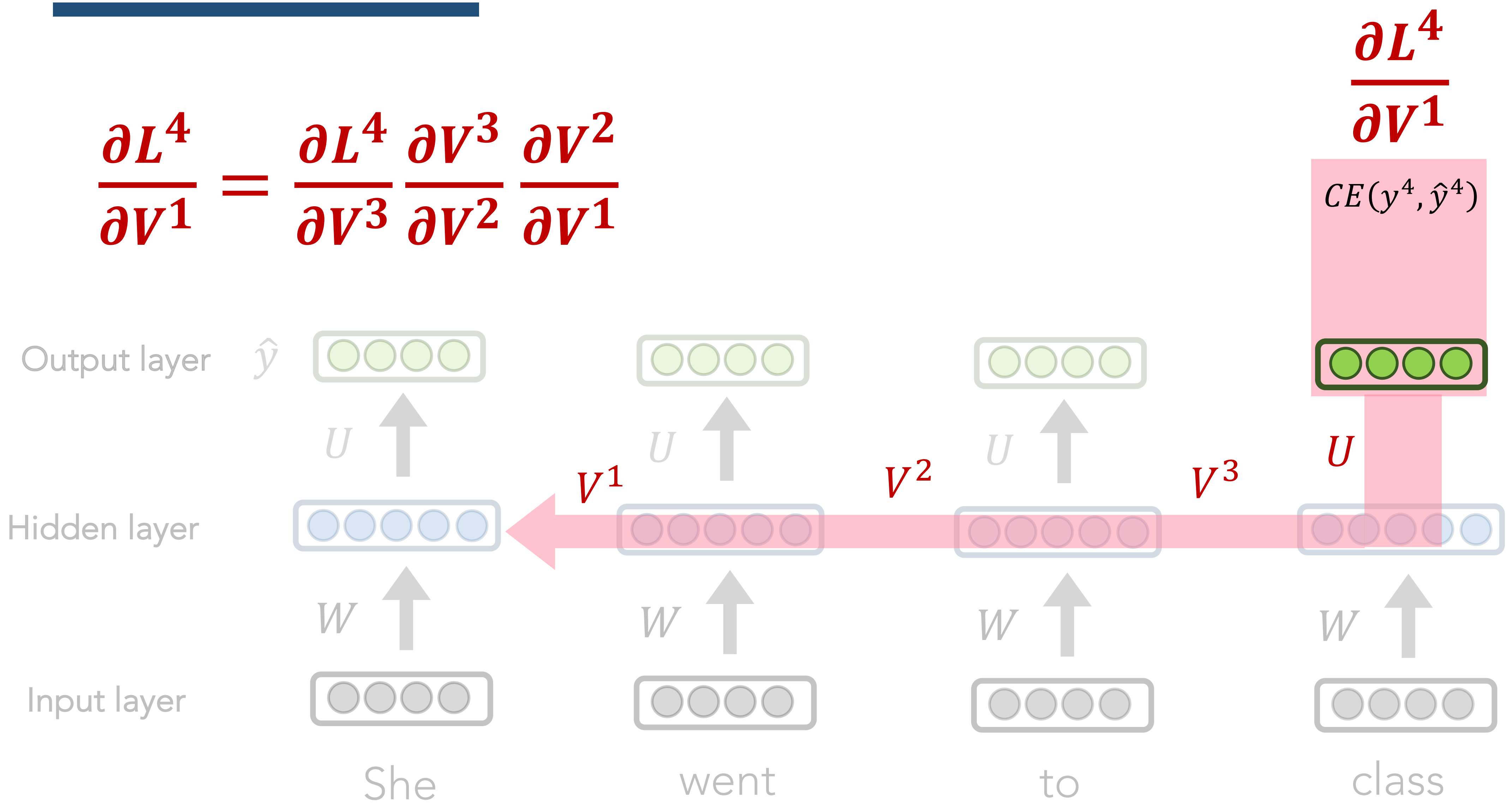
RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3}\frac{\partial V^3}{\partial V^2}$$



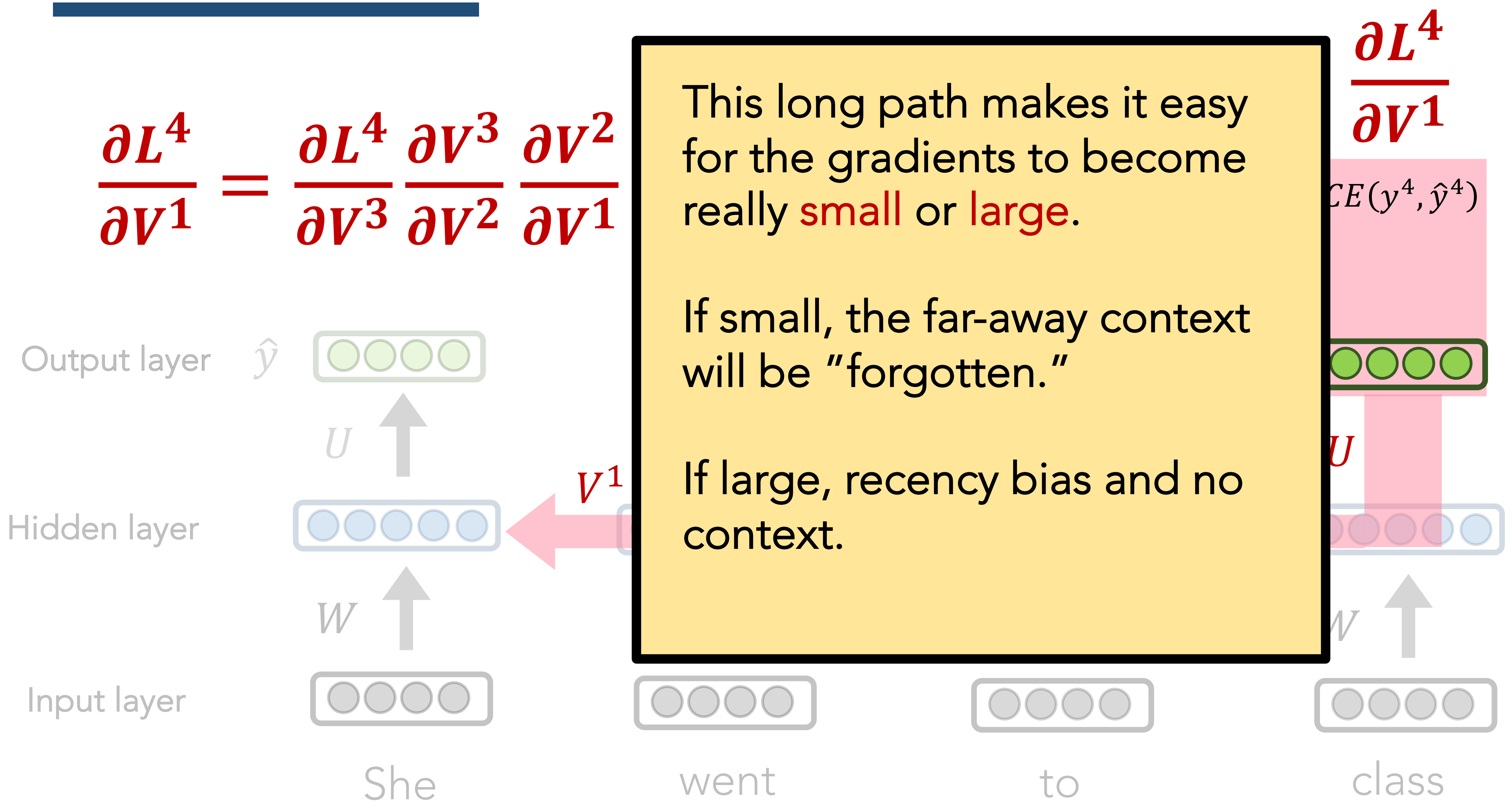
RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3}\frac{\partial V^3}{\partial V^2}\frac{\partial V^2}{\partial V^1}$$



RNNs: Vanishing and Exploding Gradients

$$\frac{\partial L^4}{\partial V^1} = \frac{\partial L^4}{\partial V^3} \frac{\partial V^3}{\partial V^2} \frac{\partial V^2}{\partial V^1}$$

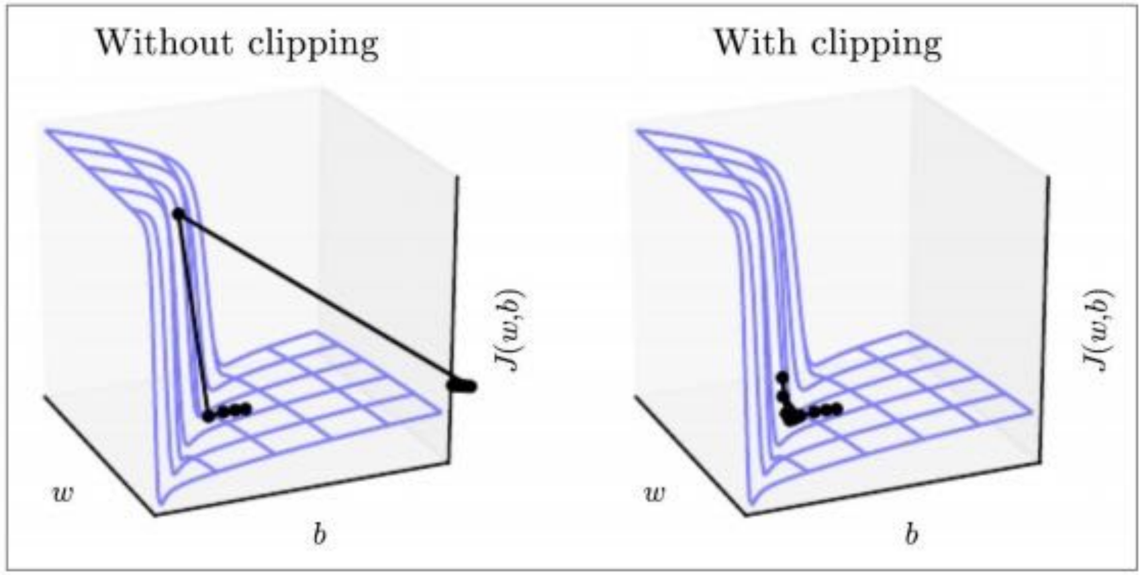


This long path makes it easy for the gradients to become really small or large.

If small, the far-away context will be "forgotten."

If large, recency bias and no context.

Exploding Gradients



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
```

Source: https://www.deeplearningbook.org/contents/rnn.html

Pascanu et al, 2013. http://proceedings.mlr.press/v28/pascanu13.pdf

BACKUP

SimLex-999's Abstract

We present SimLex-999, a gold standard resource for evaluating distributional semantic models that improves on existing resources in several important ways. First, in contrast to gold standards such as WordSim-353 and MEN, it explicitly quantifies similarity rather than association or relatedness, so that pairs of entities that are associated but not actually similar [Freud, psychology] have a low rating. We show that, via this focus on similarity, SimLex-999 incentivizes the development of models with a different, and arguably wider range of applications than those which reflect conceptual association. Second, SimLex-999 contains a range of concrete and abstract adjective, noun and verb pairs, together with an independent rating of concreteness and (free) association strength for each pair. This diversity enables fine-grained analyses of the performance of models on concepts of different types, and consequently greater insight into how architectures can be improved. Further, unlike existing gold standard evaluations, for which automatic approaches have reached or surpassed the inter-annotator agreement ceiling, state-of-the-art models perform well below this ceiling on SimLex-999. There is therefore plenty of scope for SimLex-999 to quantify future improvements to distributional semantic models, guiding the development of the next generation of representation-learning architectures.

SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. Hill et al. (2014)