

Natural Language Processing with Deep Learning

CS224N/Ling284

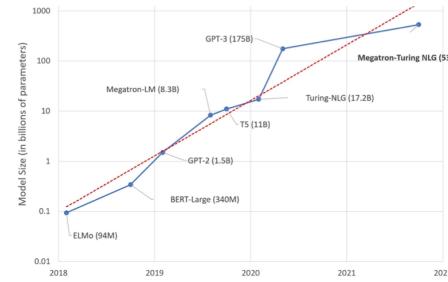


Tatsunori Hashimoto

Lecture 5: Language Models and Recurrent Neural Networks

(Slides mostly from Chris Manning's 2023 version)

Modern neural networks (esp. language models) are enormous

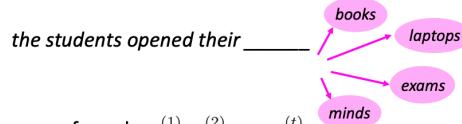


- Large, deep neural nets are a cornerstone of modern NLP systems

<https://huggingface.co/blog/large-language-models>

2. Language Modeling

- Language Modeling is the task of predicting what word comes next



- More formally: given a sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, compute the probability distribution of the next word $x^{(t+1)}$:

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

where $x^{(t+1)}$ can be any word in the vocabulary $V = \{w_1, \dots, w_{|V|}\}$

- A system that does this is called a Language Model

Language Modeling

- You can also think of a Language Model as a system that assigns a probability to a piece of text
- For example, if we have some text $x^{(1)}, \dots, x^{(T)}$, then the probability of this text (according to the Language Model) is:

$$\begin{aligned} P(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}) &= P(\mathbf{x}^{(1)}) \times P(\mathbf{x}^{(2)} | \mathbf{x}^{(1)}) \times \dots \times P(\mathbf{x}^{(T)} | \mathbf{x}^{(T-1)}, \dots, \mathbf{x}^{(1)}) \\ &= \prod_{t=1}^T P(\mathbf{x}^{(t)} | \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(1)}) \end{aligned}$$

This is what our LM provides

You use Language Models every day!



14

n-gram Language Models

the students opened their _____

- **Question:** How to learn a Language Model?
- **Answer (pre- Deep Learning):** learn an *n*-gram Language Model!
- **Definition:** An *n*-gram is a chunk of *n* consecutive words.
 - unigrams: "the", "students", "opened", "their"
 - bigrams: "the students", "students opened", "opened their"
 - trigrams: "the students opened", "students opened their"
 - four-grams: "the students opened their"
- **Idea:** Collect statistics about how frequent different n-grams are and use these to predict next word.

15

n-gram Language Models

- First we make a **Markov assumption**: $x^{(t+1)}$ depends only on the preceding *n*-1 words

$$P(x^{(t+1)}|x^{(t)}, \dots, x^{(1)}) = P(x^{(t+1)}|\underbrace{x^{(t)}, \dots, x^{(t-n+2)}}_{n-1 \text{ words}}) \quad (\text{assumption})$$

prob of a n-gram
 = $P(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})$
 prob of a (n-1)-gram
 = $P(x^{(t)}, \dots, x^{(t-n+2)})$

(definition of conditional prob)

- **Question:** How do we get these *n*-gram and (*n*-1)-gram probabilities?
- **Answer:** By **counting** them in some large corpus of text!

$$\approx \frac{\text{count}(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{\text{count}(x^{(t)}, \dots, x^{(t-n+2)})} \quad (\text{statistical approximation})$$

16

n-gram Language Models: Example

Suppose we are learning a 4-gram Language Model.

~~as the proctor started the clock, the students opened their~~ _____
 discard condition on this

$$P(w|\text{students opened their}) = \frac{\text{count(students opened their } w\text{)}}{\text{count(students opened their)}}$$

For example, suppose that in the corpus:

- "students opened their" occurred 1000 times
- "students opened their books" occurred 400 times
 - $\rightarrow P(\text{books} | \text{students opened their}) = 0.4$
- "students opened their exams" occurred 100 times
 - $\rightarrow P(\text{exams} | \text{students opened their}) = 0.1$

Should we have discarded the "proctor" context?

17

Sparsity Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if “students opened their w ” never occurred in data? Then w has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called *smoothing*.

$$P(w|\text{students opened their } w) = \frac{\text{count(students opened their } w\text{)}}{\text{count(students opened their)}}$$

Sparsity Problem 2

Problem: What if “students opened their” never occurred in data? Then we can't calculate probability for *any* w !

(Partial) Solution: Just condition on “opened their” instead. This is called *backoff*.

Note: Increasing n makes sparsity problems worse. Typically, we can't have n bigger than 5.

18

Storage Problems with n-gram Language Models

Storage: Need to store count for all n -grams you saw in the corpus.

$$P(w|\text{students opened their } w) = \frac{\text{count(students opened their } w\text{)}}{\text{count(students opened their)}}$$

Increasing n or increasing corpus increases model size!

19

n-gram Language Models in practice

- You can build a simple trigram Language Model over a 1.7 million word corpus (Reuters) in a few seconds on your laptop*

today the _____

Business and financial news

company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

Sparsity problem:
not much granularity
in the probability
distribution

Otherwise, seems reasonable!

* Try for yourself: <https://nlpforhackers.io/language-models/>

20

Generating text with a n-gram Language Model

You can also use a Language Model to generate text

today the _____

condition
on this

get probability
distribution

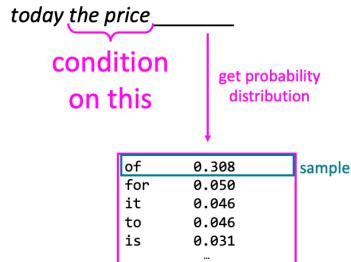
company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

sample

21

Generating text with a n-gram Language Model

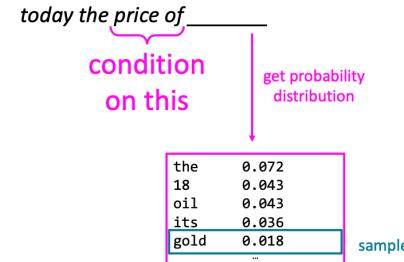
You can also use a Language Model to generate text



22

Generating text with a n-gram Language Model

You can also use a Language Model to generate text



23

Generating text with a n-gram Language Model

You can also use a Language Model to generate text

*today the price of gold per ton , while production of shoe
lasts and shoe industry , the bank intervened just after it
considered and rejected an imf demand to rebuild depleted
european stocks , sept 30 end primary 76 cts a share .*

Surprisingly grammatical!

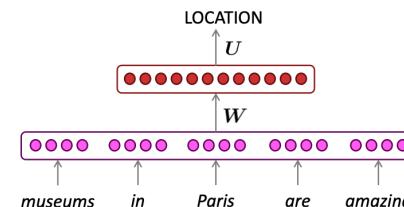
...but **incoherent**. We need to consider more than
three words at a time if we want to model language well.

But increasing n worsens sparsity problem,
and increases model size...

24

How to build a *neural* language model?

- Recall the Language Modeling task:
 - Input: sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(t)}$
 - Output: prob. dist. of the next word $P(x^{(t+1)} | x^{(1)}, \dots, x^{(t)})$
- How about a **window-based neural model**?
 - We saw this applied to Named Entity Recognition in Lecture 2:



25

A fixed-window neural Language Model

as the proctor started the clock
the students opened their _____
fixed window

26

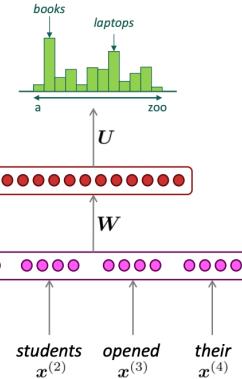
A fixed-window neural Language Model

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

hidden layer
 $h = f(We + b_1)$

concatenated word embeddings
 $e = [e^{(1)}, e^{(2)}; e^{(3)}; e^{(4)}]$

words / one-hot vectors
 $x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}$



27

A fixed-window neural Language Model

Approximately: Y. Bengio, et al. (2000/2003): A Neural Probabilistic Language Model

Improvements over n-gram LM:

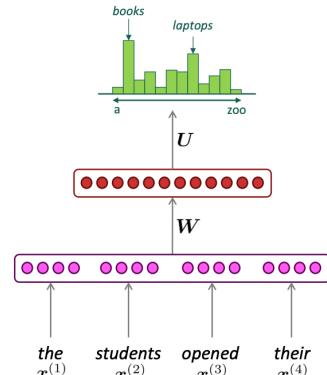
- No sparsity problem
- Don't need to store all observed n-grams

Remaining problems:

- Fixed window is **too small**
- Enlarging window enlarges W
- Window can never be large enough!
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W.
- No symmetry** in how the inputs are processed.

We need a neural architecture that can process *any length input*

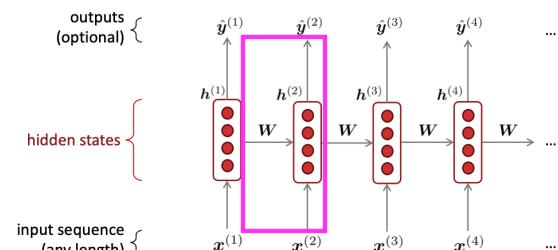
28



3. Recurrent Neural Networks (RNN)

A family of neural architectures

Core idea: Apply the same weights W repeatedly



29

A Simple RNN Language Model

output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}h^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(\mathbf{W}_h h^{(t-1)} + \mathbf{W}_e e^{(t)} + b_1)$$

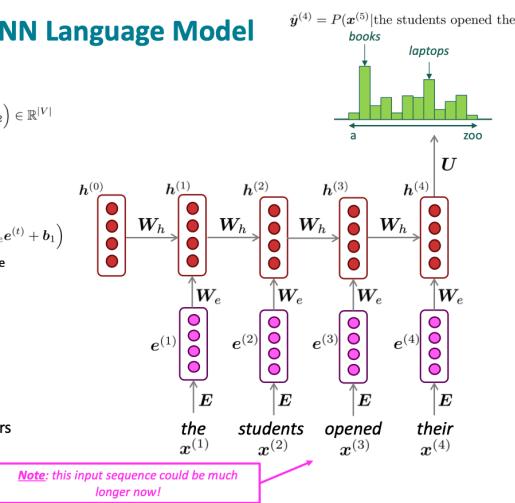
$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = \mathbf{E}x^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$



30

RNN Language Models

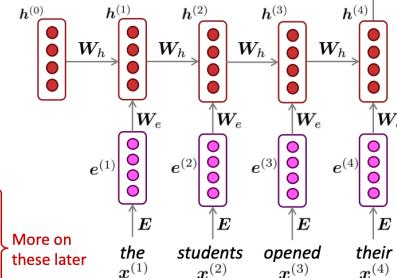
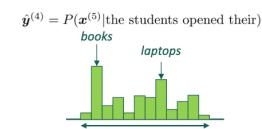
RNN Advantages:

- Can process **any length** input
- Computation for step t can (in theory) use information from **many steps back**
- Model size **doesn't increase** for longer input context
- Same weights applied on every timestep, so there is **symmetry** in how inputs are processed.

RNN Disadvantages:

- Recurrent computation is **slow**
- In practice, difficult to access information from **many steps back**

More on these later



31

Training an RNN Language Model

- Get a **big corpus of text** which is a sequence of words $x^{(1)}, \dots, x^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{y}^{(t)}$ for **every step t**.
 - i.e., predict probability dist of **every word**, given words so far
- **Loss function** on step t is **cross-entropy** between predicted probability distribution $\hat{y}^{(t)}$, and the true next word $y^{(t)}$ (one-hot for $x^{(t+1)}$):

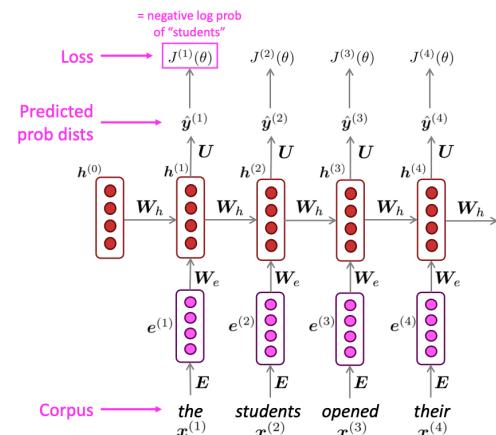
$$J^{(t)}(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = - \sum_{w \in V} y_w^{(t)} \log \hat{y}_w^{(t)} = -\log \hat{y}_{x_{t+1}}^{(t)}$$

- Average this to get **overall loss** for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T -\log \hat{y}_{x_{t+1}}^{(t)}$$

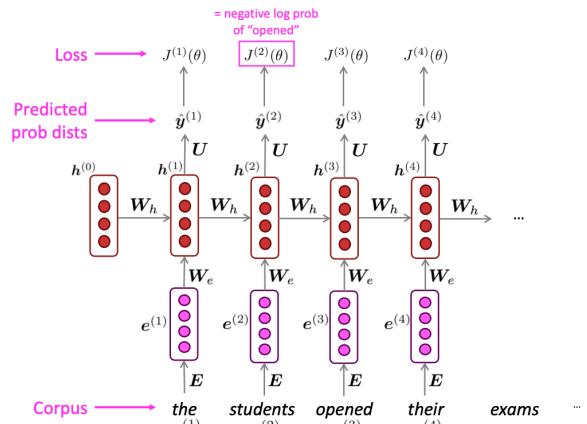
32

Training an RNN Language Model



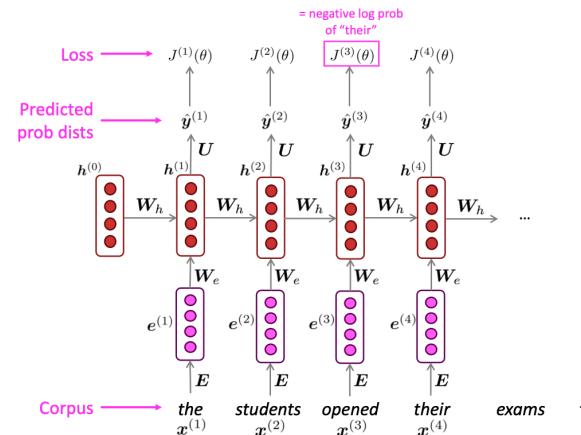
33

Training an RNN Language Model



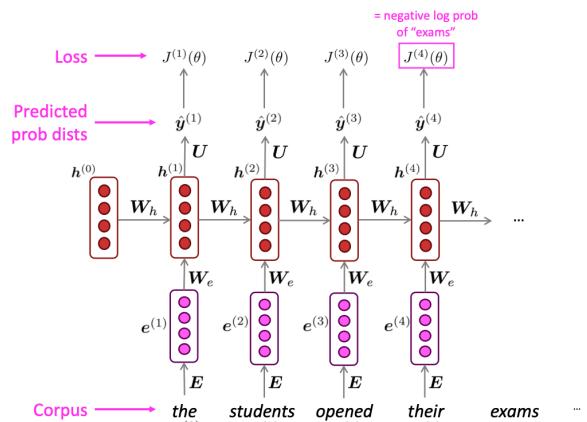
34

Training an RNN Language Model



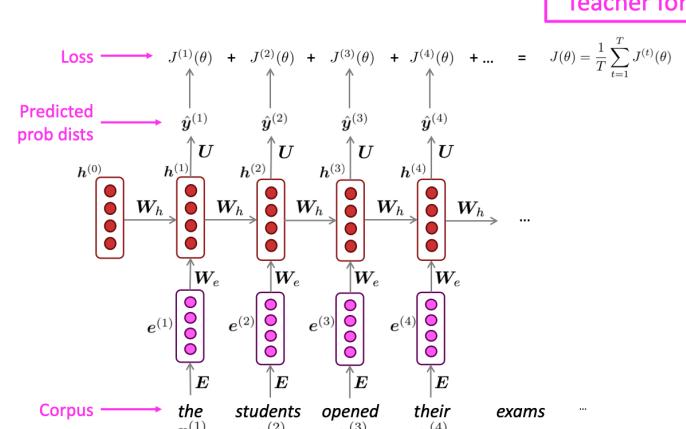
35

Training an RNN Language Model



36

Training an RNN Language Model



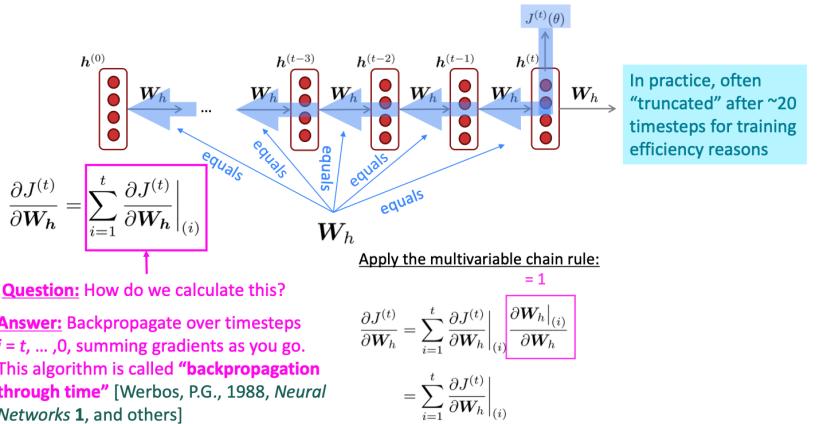
37

Training a RNN Language Model

- However: Computing loss and gradients across **entire corpus** $x^{(1)}, \dots, x^{(T)}$ at once is **too expensive** (memory-wise)!
- $$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta)$$
- In practice, consider $x^{(1)}, \dots, x^{(T)}$ as a **sentence** (or a **document**)
 - **Recall:** **Stochastic Gradient Descent** allows us to compute loss and gradients for small chunk of data, and update.
 - Compute loss $J(\theta)$ for a sentence (actually, a batch of sentences), compute gradients and update weights. Repeat on a new batch of sentences.

38

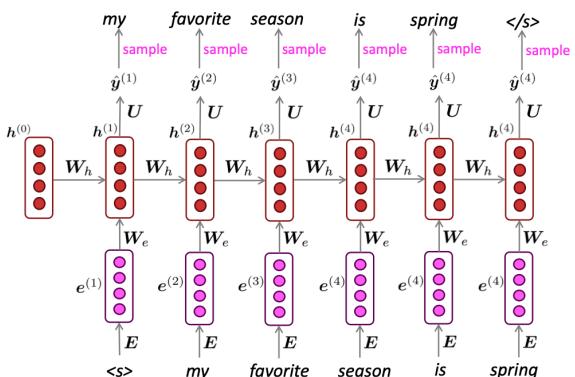
Training the parameters of RNNs: Backpropagation for RNNs



41

Generating with an RNN Language Model (“Generating roll outs”)

Just like an n-gram Language Model, you can use a RNN Language Model to generate text by **repeated sampling**. Sampled output becomes next step’s input.



42

Generating text with an RNN Language Model

Let’s have some fun!

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on **Harry Potter**:



“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>

44

Generating text with an RNN Language Model

Let's have some fun!

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM 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/nyiki/1efbaa36635956d35bcc>



45

Generating text with a RNN Language Model

Let's have some fun!

- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on **paint color names**:

Ghasty Pink	231	137	165
Power Gray	151	124	112
Navel Tan	199	173	140
Bock Cow White	221	215	236
Horbly Gray	178	181	196
Homestar Brown	133	104	85
Snader Brown	144	106	74
Golder Craam	237	217	177
Hurky White	232	223	215
Burf Pink	223	173	179
Rose Hork	230	215	198
Sand Dan	201	172	143
Grade Bat	48	94	83
Light Of Blast	175	150	147
Grass Bat	176	99	108
Sindis Poop	204	205	194
Dope	219	209	179
Testing	156	101	106
Stone Blue	152	165	159
Burble Simp	226	181	132
Stanky Bean	197	162	171
Turdly	190	164	116

This is an example of a **character-level RNN-LM** (predicts what character comes next)

Source: <http://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network>

46

Evaluating Language Models

- The standard **evaluation metric** for Language Models is **perplexity**.

$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Inverse probability of corpus, according to Language Model

Normalized by number of words

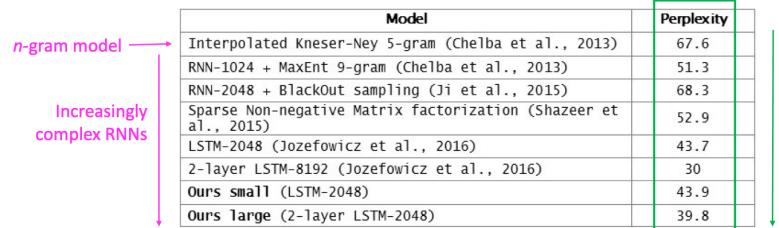
- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^T \left(\frac{1}{y_{\mathbf{x}^{(t+1)}}^{(t)}} \right)^{1/T} = \exp \left(\frac{1}{T} \sum_{t=1}^T -\log y_{\mathbf{x}^{(t+1)}}^{(t)} \right) = \exp(J(\theta))$$

Lower perplexity is better!

47

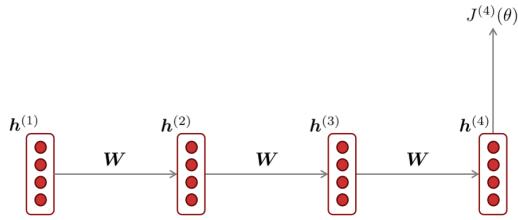
RNNs greatly improved perplexity over what came before



Source: <https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/>

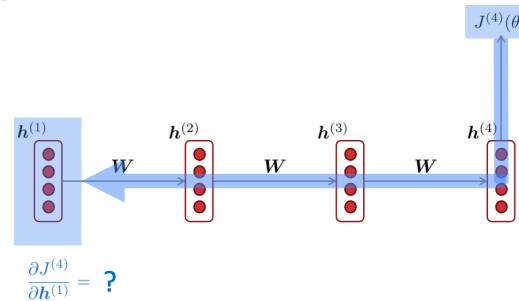
48

4. Problems with RNNs: Vanishing and Exploding Gradients



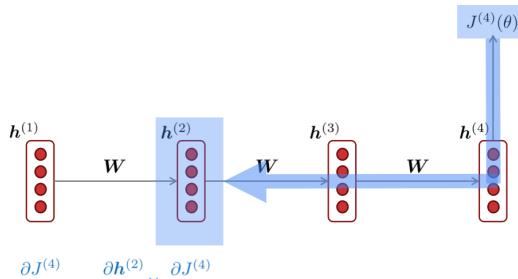
49

Vanishing gradient intuition



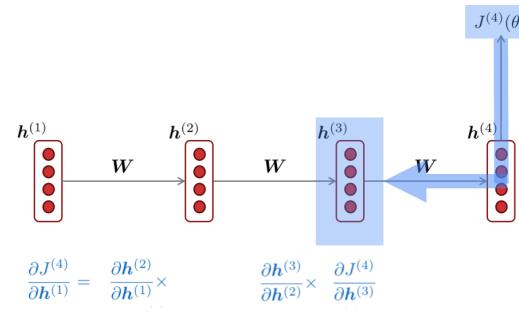
50

Vanishing gradient intuition



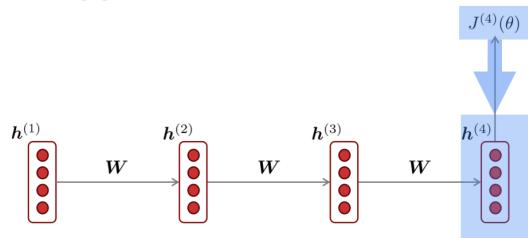
51

Vanishing gradient intuition



52

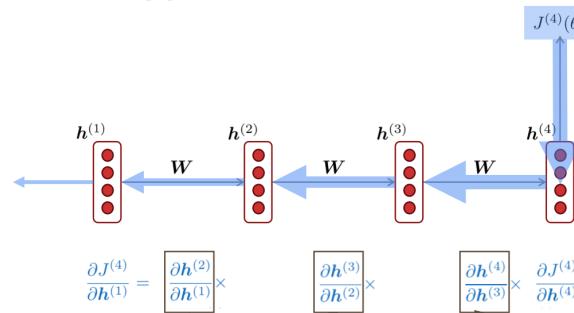
Vanishing gradient intuition



chain rule!

53

Vanishing gradient intuition

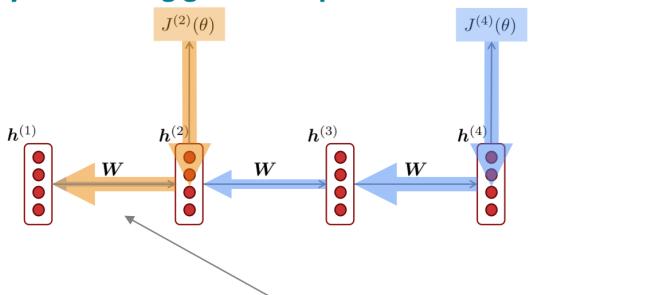


What happens if these are small?

Vanishing gradient problem:
When these are small, the gradient signal gets smaller and smaller as it backpropagates further

54

Why is vanishing gradient a problem?



Gradient signal from far away is lost because it's much smaller than gradient signal from close-by.

So, model weights are updated only with respect to near effects, not long-term effects.

57

Effect of vanishing gradient on RNN-LM

- **LM task:** When she tried to print her tickets, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her _____
- To learn from this training example, the RNN-LM needs to model the dependency between “tickets” on the 7th step and the target word “tickets” at the end.
- But if the gradient is small, the model can't learn this dependency
 - So, the model is unable to predict similar long-distance dependencies at test time

58

Why is exploding gradient a problem?

- If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta^{new} = \theta^{old} - \underbrace{\alpha \nabla_{\theta} J(\theta)}_{\text{gradient}}$$

- This can cause **bad updates**: we take too large a step and reach a weird and bad parameter configuration (with large loss)
 - You think you've found a hill to climb, but suddenly you're in Iowa
- In the worst case, this will result in **Inf** or **NaN** in your network (then you have to restart training from an earlier checkpoint)

59

Gradient clipping: solution for exploding gradient

- Gradient clipping**: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

Algorithm 1 Pseudo-code for norm clipping

```
 $\hat{g} \leftarrow \frac{\partial E}{\partial \theta}$ 
if  $\|\hat{g}\| \geq \text{threshold}$  then
     $\hat{g} \leftarrow \frac{\text{threshold}}{\|\hat{g}\|} \hat{g}$ 
end if
```

- Intuition**: take a step in the same direction, but a smaller step
- In practice, **remembering to clip gradients is important**, but exploding gradients are an easy problem to solve

60

Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013. <http://proceedings.mlr.press/v28/pascanu13.pdf>

How to fix the vanishing gradient problem?

- The main problem is that *it's too difficult for the RNN to learn to preserve information over many timesteps*.
- In a vanilla RNN, the hidden state is constantly being **rewritten**
$$h^{(t)} = \sigma \left(W_h h^{(t-1)} + W_x x^{(t)} + b \right)$$
- First off next time: How about an RNN with separate **memory** which is added to?
 - LSTMs
- And then: Creating more direct and linear pass-through connections in model
 - Attention, residual connections, etc.

61

5. Recap

- Language Model**: A system that **predicts the next word**
- Recurrent Neural Network**: A family of neural networks that:
 - Take **sequential input of any length**
 - Apply the **same weights on each step**
 - Can optionally produce **output on each step**
- Recurrent Neural Network \neq Language Model**
- We've shown that RNNs are a great way to build a LM (despite some problems)
- RNNs are also useful for much more!

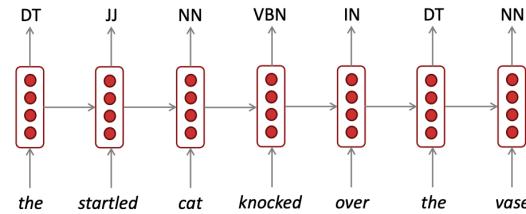
62

Why should we care about Language Modeling?

- Language Modeling is a **benchmark task** that helps us measure our progress on predicting language use
- Language Modeling is a **subcomponent** of many NLP tasks, especially those involving **generating text** or **estimating the probability of text**:
 - Predictive typing
 - Speech recognition
 - Handwriting recognition
 - Spelling/grammar correction
 - Authorship identification
 - Machine translation
 - Summarization
 - Dialogue
 - etc.
- Everything else in NLP has now been rebuilt upon Language Modeling: **GPT-3 is an LM!**

63

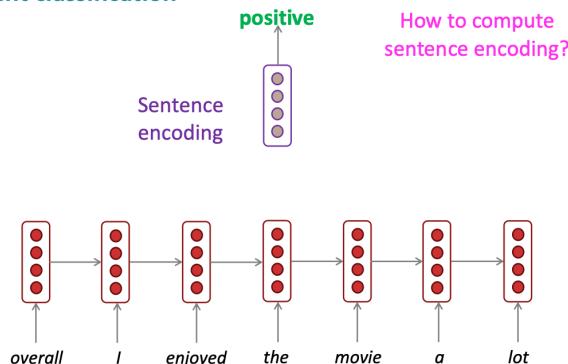
Other RNN uses: RNNs can be used for sequence tagging e.g., part-of-speech tagging, named entity recognition



64

RNNs can be used for sentence classification

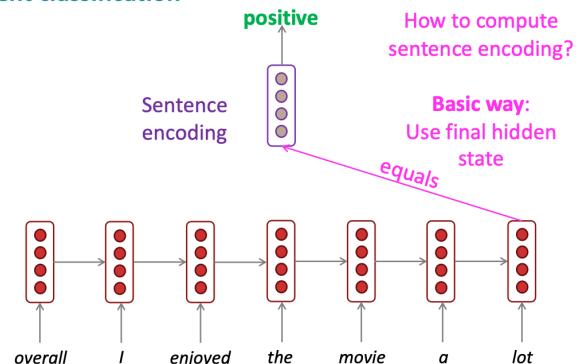
e.g., sentiment classification



65

RNNs can be used for sentence classification

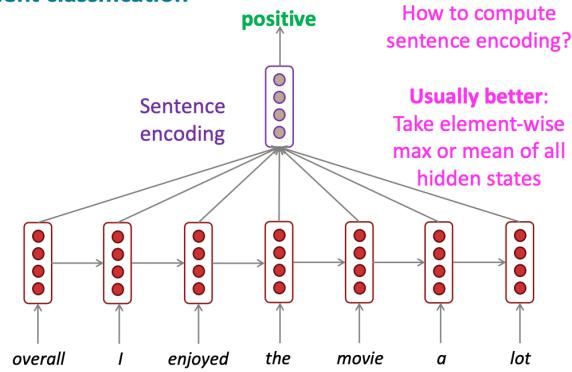
e.g., sentiment classification



66

RNNs can be used for sentence classification

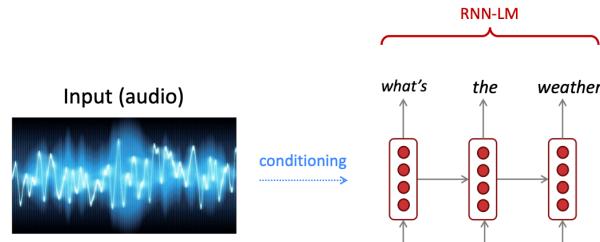
e.g., sentiment classification



67

RNN-LMs can be used to generate text

e.g., speech recognition, machine translation, summarization



This is an example of a *conditional language model*.
We'll see Machine Translation in much more detail starting next lecture.

69