LARGE LANGUAGE MODELS

David Kauchak
CS 159 – Fall 2024

1

Admin

Assignment 7 due Wednesday

Final project proposals due Thursday

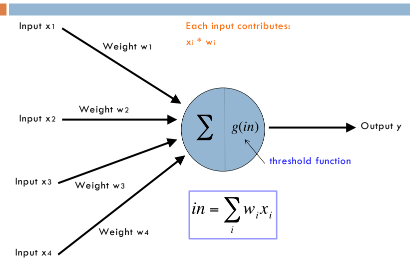
Start working on the projects!
- Log hours that you work

No class Thursday

Quiz 3 back today

2

A Single Neuron/Perceptron

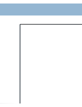


3

Activation functions

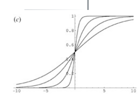
hard threshold:

$$g(in) = \begin{cases} 1 & if\ in \geq T \\ 0 & otherwise \end{cases}$$

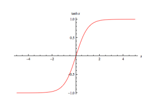


sigmoid

$$g(x) = \frac{1}{1 + e^{-ax}}$$



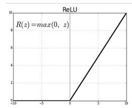
tanh x



4

Many other activation functions

Rectified Linear Unit

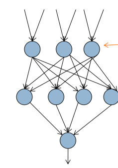


Softmax (for probabilities)

5

Neural network

inputs

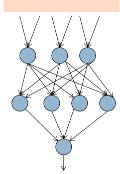


Individual perceptrons/neurons

6

Neural network

inputs

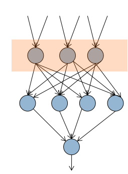


some inputs are provided/entered

7

Neural network

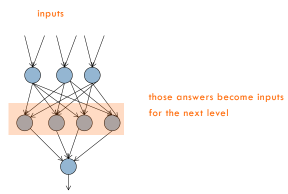
inputs



each perceptron computes and calculates an answer

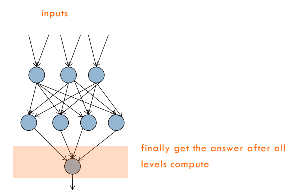
8

Neural network



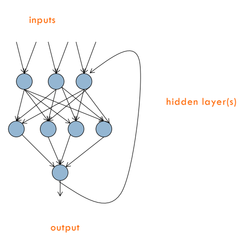
9

Neural network



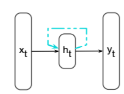
10

Recurrent neural networks



11

Recurrent neural nets



x_t = input

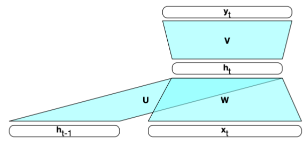
h_t = hidden layer output

y_t = output

Figure 9.1 from Jurafsky and Martin

12

Recurrent neural networks



x_t = input

h_{t-1} = hidden layer output from previous input

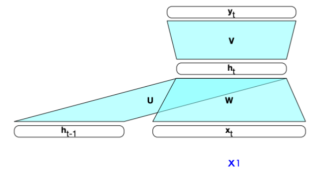
h_t = hidden layer output

y_t = output

Figure 9.2 from Jurafsky and Martin

13

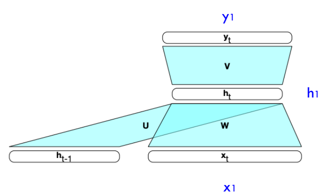
Recurrent neural networks



Say you want the output of x1, x2, x3, ….

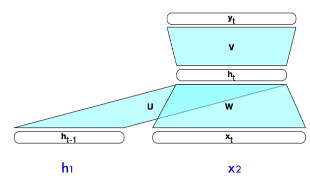
14

Recurrent neural networks



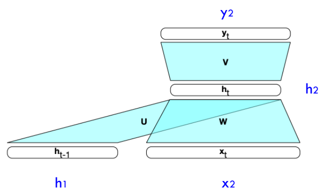
15

Recurrent neural networks



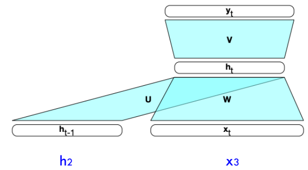
16

Recurrent neural networks



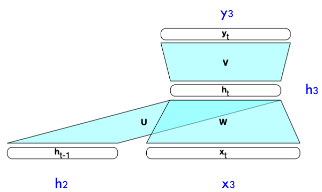
17

Recurrent neural networks



18

Recurrent neural networks



19

RNNs unrolled

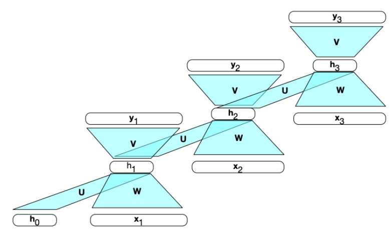
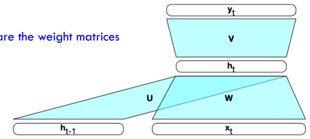


Figure 9.2 from Jurafsky and Martin

20

Still just a single neural network

U, W and V are the weight matrices



x_t = input

h_{t-1} = hidden layer output from previous input

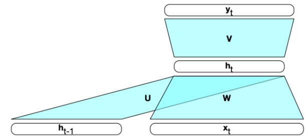
h_t = hidden layer output

y_t = output

Figure 9.2 from Jurafsky and Martin

21

RNN language models



How can we use RNNs as language models p(w1, w2, …, wn)?

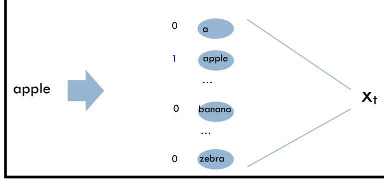
How do we input a word into a NN?

22

"One-hot" encoding

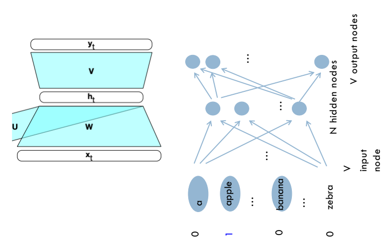
For a vocabulary of V words, have V input nodes

All inputs are 0 except the for the one corresponding to the word



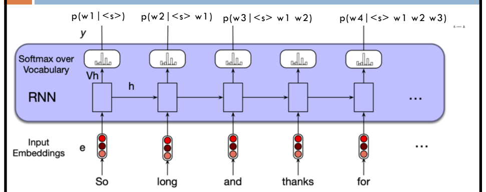
23

RNN language model



24

RNN language model

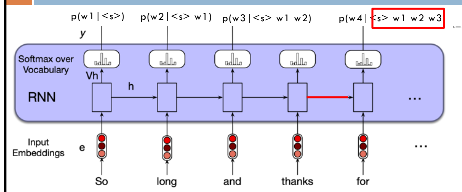


Softmax = turn into probabilities

Figure 9.6 from Jurafsky and Martin

25

RNN language model

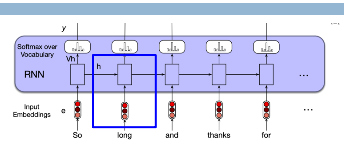


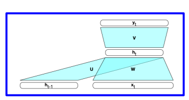
Softmax = turn into probabilities

Figure 9.6 from Jurafsky and Martin

26

RNN language model





27

Training RNN LM

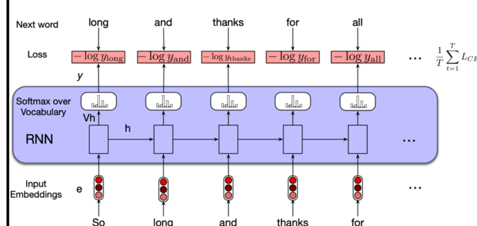


Figure 9.6 from Jurafsky and Martin

28

Generation with RNN LM

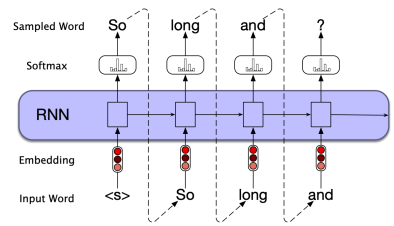


Figure 9.9 from Jurafsky and Martin

29

Stacked RNNs

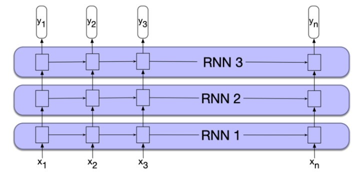
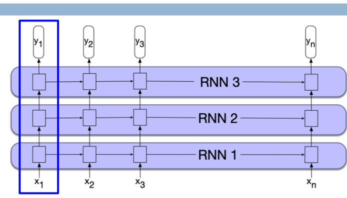


Figure 9.10 from Jurafsky and Martin

30

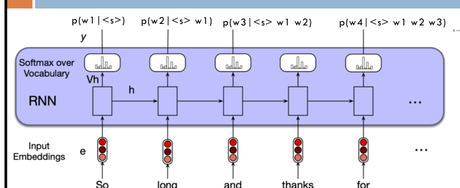
Stacked RNNs



- Multiple hidden layers
- Still just a single network run over a sequence
- Allows for better generalization, but can take longer to train and more data!

31

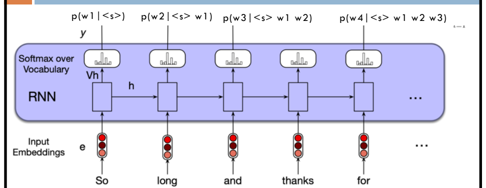
Challenges with RNN LMs



What context is incorporated for predicting w_i?

32

Challenges with RNN LMs

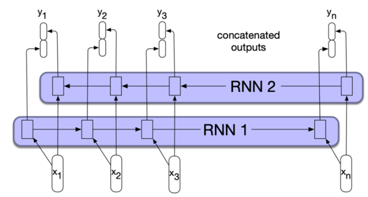


Just like with an n-gram LM, only use previous history.

What are we missing if we're predicting p(w1, w2, …, wn)?

33

Bidirectional RNN

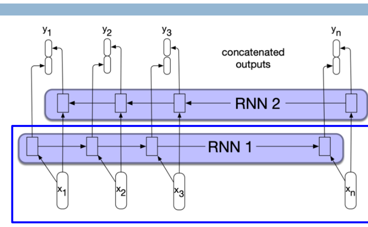


Normal forward RNN

Figure 9.11 from Jurafsky and Martin

34

Bidirectional RNN

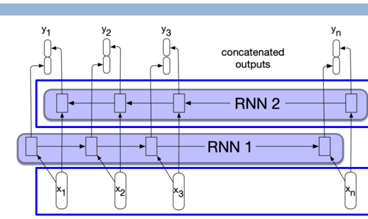


Normal forward RNN

Figure 9.11 from Jurafsky and Martin

35

Bidirectional RNN

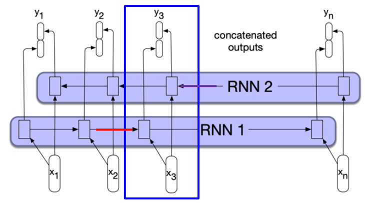


Backward RNN, starting from the last word

Figure 9.11 from Jurafsky and Martin

36

Bidirectional RNN

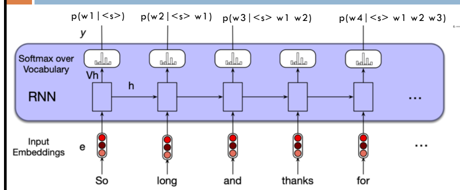


Prediction uses collected information from the words before (**left**) and words after (**right**)

Figure 9.11 from Jurafsky and Martin

37

Challenges with RNN LMs

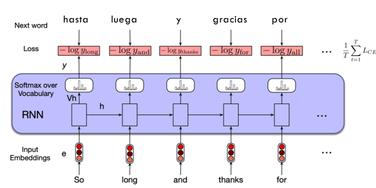


Can we use them for translation (and related tasks)?

Any challenges?

38

Challenges with RNN LMs

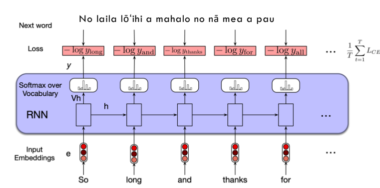


Can we use them for translation (and related tasks)?

Any challenges?

39

Challenges with RNN LMs

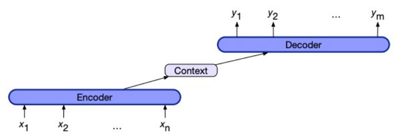


Translation isn't word-to-word

Worse for other tasks like summarization

40

Encoder-decoder models

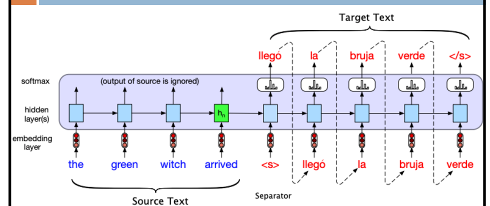


Idea:
- Process the input sentence (e.g., sentence to be translated) with a network
- Represent the sentence as some function of the hidden states (encoding)
- Use this context to generate the output

Figure 9.16 from Jurafsky and Martin

41

Encoder-decoder models: simple version

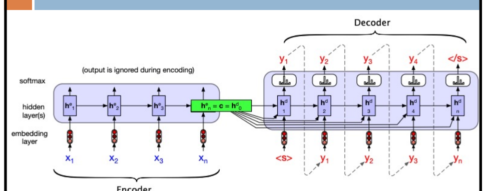


The context is the final hidden state of the encoder and is provided as input to the first step of the decoder

Figure 9.17 from Jurafsky and Martin

42

Encoder-decoder models: improved



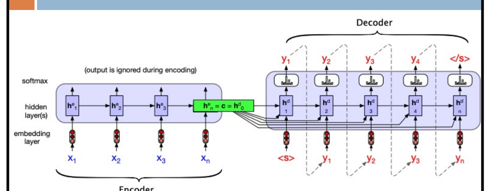
The context is some combination of **all of the** hidden states of the encoder

How is this better?

Figure 9.18 from Jurafsky and Martin

43

Encoder-decoder models: improved



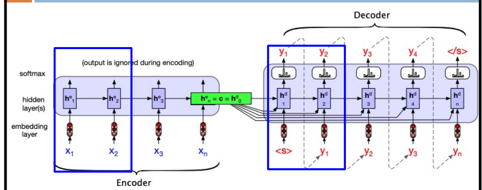
The context is some combination of **all of the** hidden states of the encoder

Each step of decoding has access to the original, full encoding/context

Figure 9.18 from Jurafsky and Martin

44

Encoder-decoder models: improved

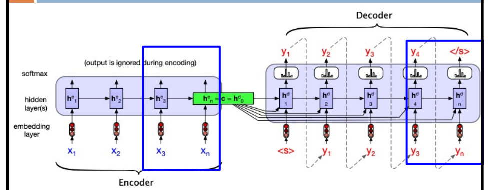


Even with this model, different encoding steps may care about different parts of the context

Figure 9.18 from Jurafsky and Martin

45

Encoder-decoder models: improved

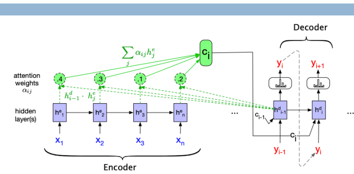


Even with this model, different encoding steps may care about different parts of the context

Figure 9.18 from Jurafsky and Martin

46

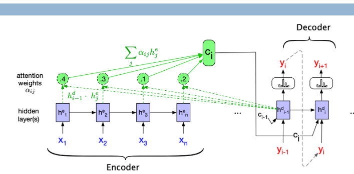
Attention



Context is dependent on where we are in decoding step and the relationship between encoder and decoder hidden states

47

Attention

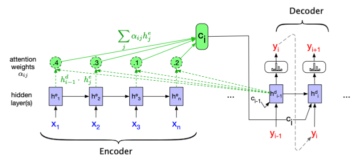


Simple version attention is static, but can learn attention mechanism (i.e., relationship between encoder and decoder hidden states)

Figure 9.23 from Jurafsky and Martin

48

Attention

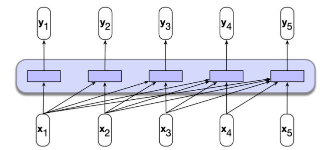


Key RNN challenge: computation is sequential
- This prevents parallelization
- Harder to model contextual dependencies

Figure 9.23 from Jurafsky and Martin

49

Another model

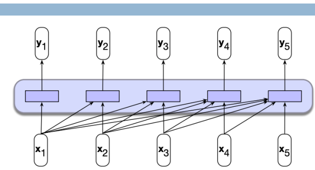


How is this setup different from the RNN?

Figure 10.1 from Jurafsky and Martin

50

Another model



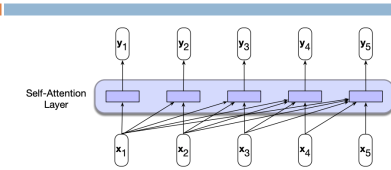
Do not rely on the hidden states for context information

Parallel: computation can all happen at once

Figure 10.1 from Jurafsky and Martin

51

Self-attention



Self-attention:
- Input is some context (for LMs, the previous words)
- Learn what parts of the context are important based

Figure 10.1 from Jurafsky and Martin

52

Self-attention

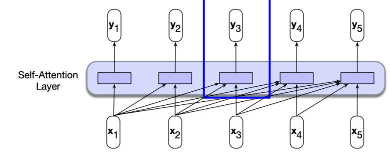


Figure 10.1 from Jurafsky and Martin

53

Transformer block

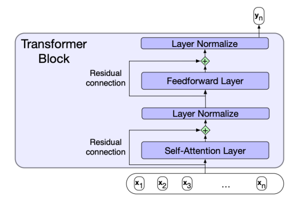
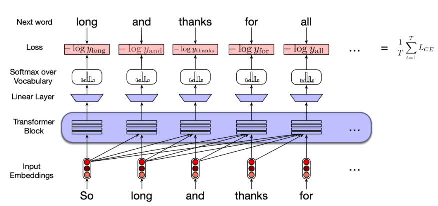


Figure 10.4 from Jurafsky and Martin

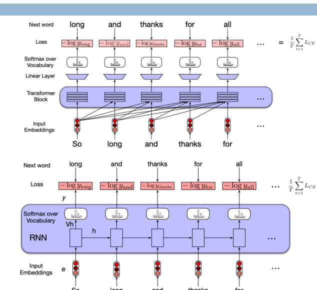
54

Transformer network



55

Transformer network vs. RNN



56

GPT

Generative: outputs things

Pre-trained: previously trained on a large corpus

Transformer: uses the transformer network

57

Pre-trained language models

Pre-trained language models are general purpose and are trained on a very large corpus

They can be used as/is to:
- Ask p(w1 w2 … wn)
- Generate text given some seed, p(wi | w1 w2 … wi-1

They can also be "fine-tuned" for particular tasks: take the current weights and update them based on a specific application

58

ChatGPT

ChatGPT is based on particular GPT foundation models, namely GPT-4, GPT-4o and GPT-4o mini, that were fine-tuned to target conversational usage.[17] The fine-tuning process leveraged supervised learning and reinforcement learning from human feedback (RLHF).[18][19] Both approaches employed human trainers to improve model performance. In the case of supervised learning, the trainers played both sides: the user and the AI assistant. In the reinforcement learning stage, human trainers first ranked responses that the model had created in a previous conversation.[20] These rankings were used to create "reward models" that were used to fine-tune the model further by using several iterations of proximal policy optimization.[18][21]

59