DEEP LEARNING

David Kauchak
CS158 – Fall 2025

1

Admin

Assignment 8

Schedule for the rest of the semester

2

Deep learning

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

Deep learning is part of a broader family of machine learning methods based on learning representations of data.

3

Deep learning

Key: learning better features that abstract from the "raw" data

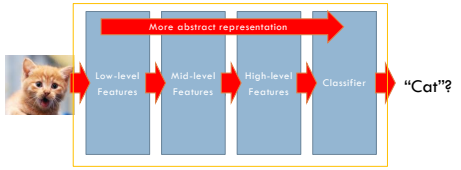
Using learned feature representations based on large amounts of data, generally unsupervised

Using classifiers with multiple layers of learning

4

Deep learning

- Train *multiple layers* of features/abstractions from data.
- Try to discover *representation* that makes decisions easy.

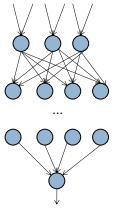


Deep Learning: train layers of features so that classifier works well.

Slide adapted from: Adam Coates

5

Deep learning for neural networks



Traditional NN models: 1-2 hidden layers

Deep learning NN models: 3+ hidden layers

6

Importance of features

Feature quality is critical to the performance of ML methods

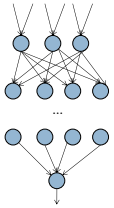
Normal process = hand-crafted features

Deep learning: find algorithms to automatically discover features from the data

7

Challenges

What makes "deep learning" hard for NNs?

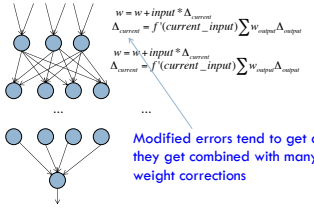


$$w = w + input * \Delta_{current}$$
$$\Delta_{current} = f'(current\_input)\sum w_{output}\Delta_{output}$$

8

Challenges

What makes "deep learning" hard for NNs?



10

Deep learning

Growing field

Driven by:

- Increase in data availability
- Increase in computational power
- Parallelizability of many of the algorithms

Involves more than just neural networks (though, they're a very popular model)

11

word2vec

How many people have heard of it?

What is it?

12

Word representations

Wine data uses word occurrences as a feature

What does this miss?

13

Word representations

Wine data uses word occurrences as a feature

What does this miss?

"The wine had a dark red color"	Zinfandel
"The wine was a deep crimson color"	label?
"The wine was a deep yellow color"	label?

Would like to recognize that words have similar meaning even though they aren't lexically the same

14

Word representations

Key idea: project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

15

Word representations

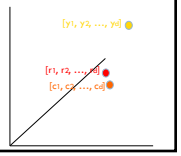
Key idea: project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

red ➡ $[r_1, r_2, \ldots, r_d]$

crimson ➡ $[c_1, c_2, \ldots, c_d]$

yellow ➡ $[y_1, y_2, \ldots, y_d]$



16

Word representations

Key idea: project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

The idea of word representations is not new:
- Co-occurrence matrices
- Latent Semantic Analysis (LSA)

New idea: learn word representation using a task-driven approach

17

A prediction problem

I like to eat bananas with cream cheese

Given a context of words

Predict what words are likely to occur in that context

18

A prediction problem

Given text, can generate lots of examples:

I like to eat bananas with cream cheese

input	prediction
___ like to eat	I
I ___ to eat bananas	like
I like ___ eat bananas with	to
I like to ___ bananas with cream	eat
...	...

19

A prediction problem

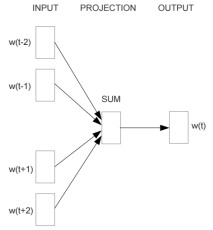
Use data like this to learn a distribution:

$$p(word \mid context)$$

$$p(w_i \mid \underbrace{w_{i-2}w_{i-1}}_{\text{words before}}\underbrace{w_{i+1}w_{i+2}}_{\text{words after}})$$

20

Train a neural network on this problem



https://arxiv.org/pdf/1301.3781v3.pdf

21

Encoding words

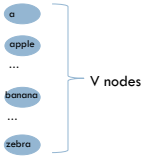
How can we input a "word" into a network?

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

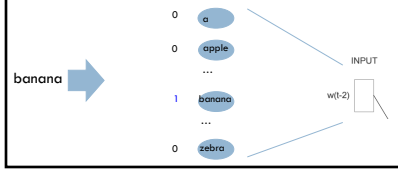


23

"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

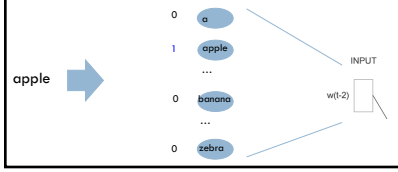


24

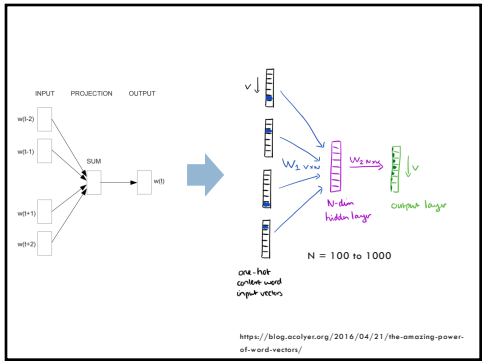
"One-hot" encoding

For a vocabulary of V words, have V input nodes

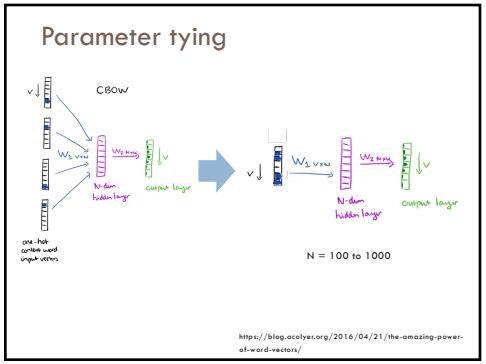
All inputs are 0 except the for the one corresponding to the word



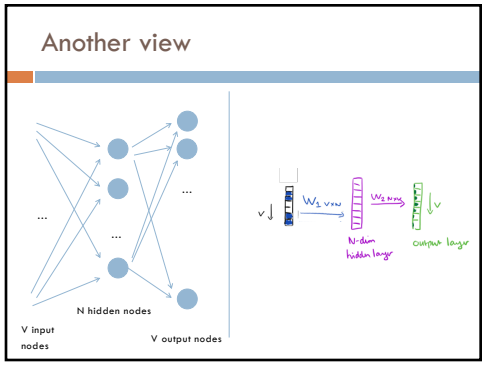
25


INPUT
PROJECTION
OUTPUT
w(t-2)
w(t-1)
w(t+1)
w(t+2)
SUM
w(t)
V
W1 V×N
W2 N×V
N-dim hidden layer
output layer
N = 100 to 1000
one-hot context word input vectors
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

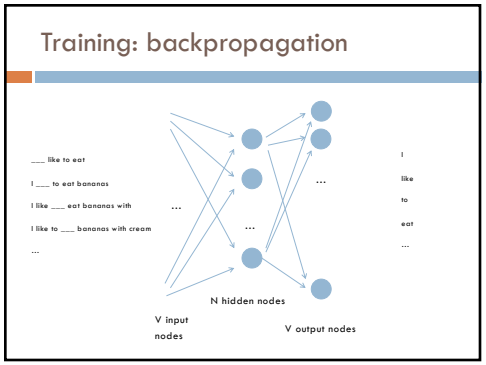
26


Parameter tying
CBOW
V
W1 V×N
W2 N×V
N-dim hidden layer
output layer
one-hot context word input vectors
N = 100 to 1000
https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

27

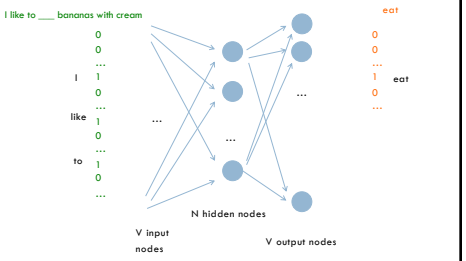

Another view
N hidden nodes
V input nodes
V output nodes
V
W1 V×N
W2 N×V
N-dim hidden layer
output layer

28


Training: backpropagation
___ like to eat
I ___ to eat bananas
I like ___ eat bananas with
I like to ___ bananas with cream
...
N hidden nodes
V input nodes
V output nodes
I
like
to
eat
...

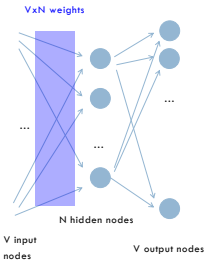
29

Training: backpropagation



30

Word representation



The weights for each word provide an N dimensional mapping of the word

Words that predict similarly should have similar weights

Why does this work?

31

Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter

32

Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Type of relationship	Word Pair 1		Word Pair 2	
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

33

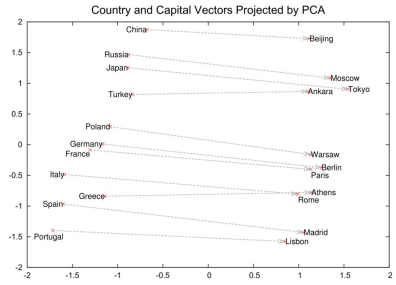
Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

34



2-Dimensional projection of the N-dimensional space

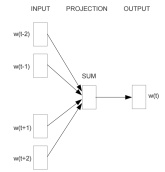
35

Visualized

https://projector.tensorflow.org/

36

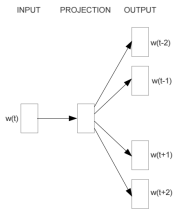
Continuous Bag Of Words



CBOW

37

Other models: skip-gram



38

word2vec

A model for learning word representations from large amounts of data

Has become a popular pre-processing step for learning a more robust feature representation

Models like word2vec have also been incorporated into other learning approaches (e.g., translation tasks)

39

word2vec resources

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

https://code.google.com/archive/p/word2vec/

https://deeplearning4j.org/word2vec

https://arxiv.org/pdf/1301.3781v3.pdf

40

Language modeling

What does natural language look like?

p(sentence)

- p("I like to eat pizza")
- p("pizza like I eat")

Often is posed as: p(word | previous words) – or some other notion of context

- p("pizza" | "I like to eat")
- p("garbage" | "I like to eat")
- p("run" | "I like to eat")

41

Ideas?

p("I like to eat pizza")

p("pizza like I eat")

p("pizza" | "I like to eat")

p("garbage" | "I like to eat")

p("run" | "I like to eat")

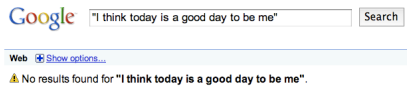
42

Look at a corpus

43

Language modeling

I think today is a good day to be me



Language modeling is about dealing with data sparsity!

44

Language modeling

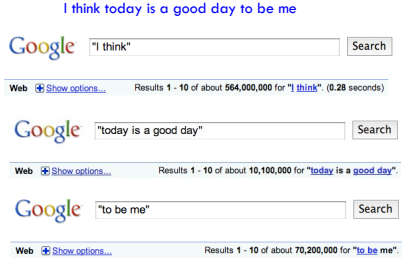
Many approaches:

- n-gram language modeling
  - Start at the beginning of the sentence
  - Generate one word at a time based on the previous words
- syntax-based language modeling
  - Construct the syntactic tree from the top down
  - e.g. context free grammar
  - eventually at the leaves, generate the words
- Neural language models
  - Predict the likelihood of the word based on the context
  - Often allows for generalization beyond the lexical strings

45

n-gram language modeling

I think today is a good day to be me



46

Our friend the chain rule

Step 1: decompose the probability

P(I think today is a good day to be me) =

P(I | <start>) x

P(think | I) x

P(today | I think) x

P(is | I think today) x

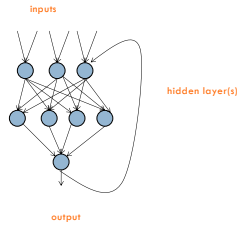
P(a | I think today is) x

P(good | I think today is a) x

…

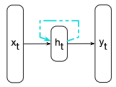
47

Recurrent neural networks



48

Recurrent neural nets



x_t = input

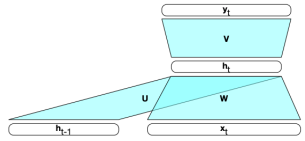
h_t = hidden layer output

y_t = output

Figure 9.1 from Jurafsky and Martin

49

Recurrent neural networks



x_t = input

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

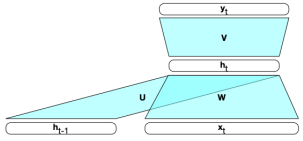
h_t = hidden layer output

y_t = output

Figure 9.2 from Jurafsky and Martin

50

Recurrent neural networks

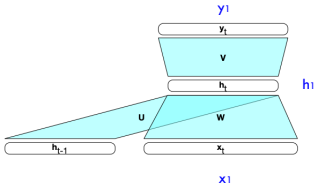


x_1

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

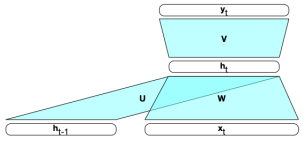
51

Recurrent neural networks



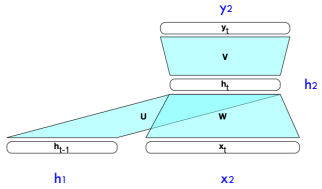
52

Recurrent neural networks



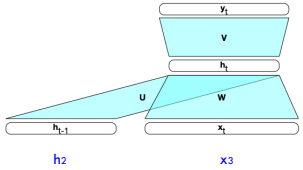
53

Recurrent neural networks



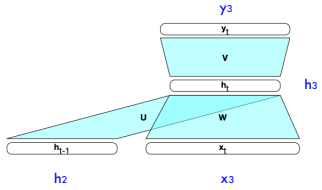
54

Recurrent neural networks



55

Recurrent neural networks



56

RNNs unrolled

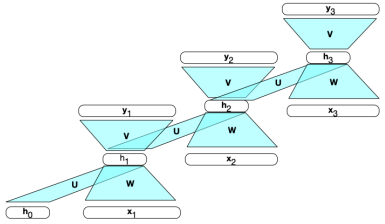
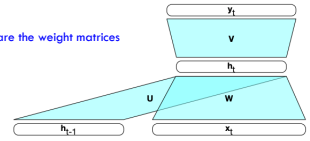


Figure 9.2 from Jurafsky and Martin

57

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

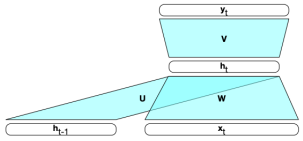
h_t = hidden layer output

y_t = output

Figure 9.2 from Jurafsky and Martin

58

RNN language models



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

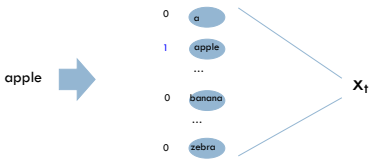
How do we input a word into a NN?

59

"One-hot" encoding

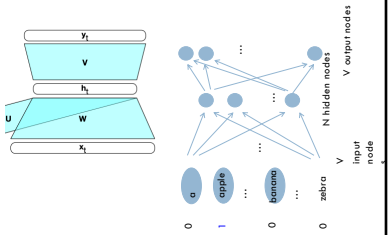
For a vocabulary of V words, have V input nodes

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



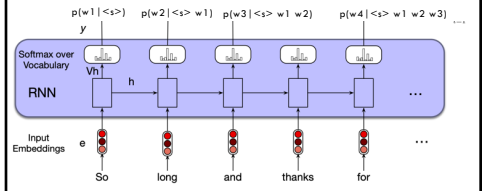
60

RNN language model



61

RNN language model

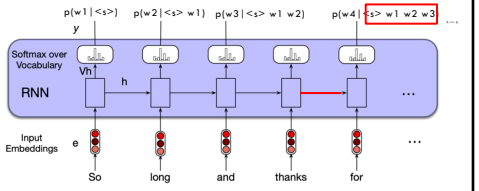


Softmax = turn into probabilities

Figure 9.6 from Jurafsky and Martin

62

RNN language model

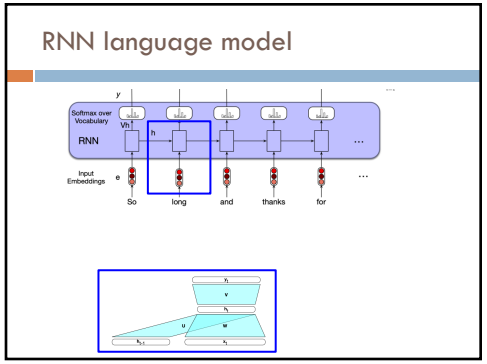


Softmax = turn into probabilities

Figure 9.6 from Jurafsky and Martin

63

RNN language model



64

Training RNN LM

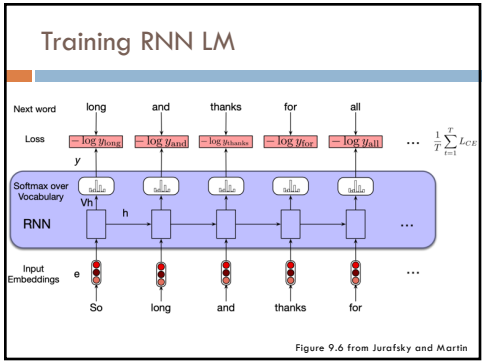


Figure 9.6 from Jurafsky and Martin

65

Generation with RNN LM

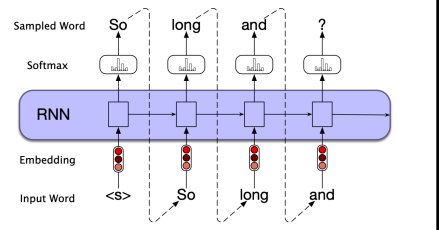


Figure 9.9 from Jurafsky and Martin

66

Stacked RNNs

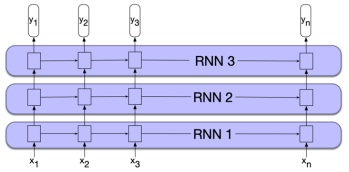
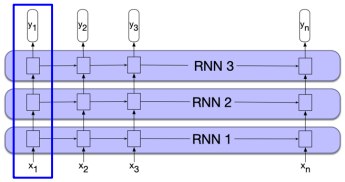


Figure 9.10 from Jurafsky and Martin

67

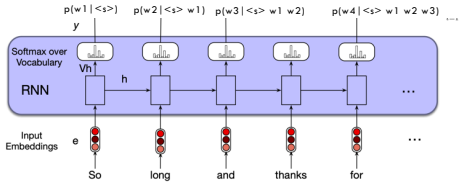
Stacked RNNs



- 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!

68

Challenges with RNN LMs

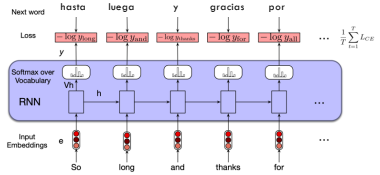


Can we use them for translation (and related tasks)?

Any challenges?

75

Challenges with RNN LMs

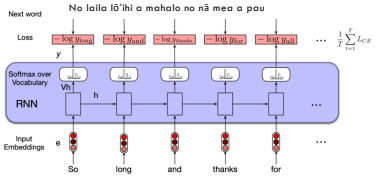


Can we use them for translation (and related tasks)?

Any challenges?

76

Challenges with RNN LMs

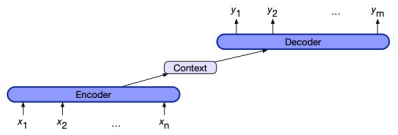


Translation isn't word-to-word

Worse for other tasks like summarization

77

Encoder-decoder models

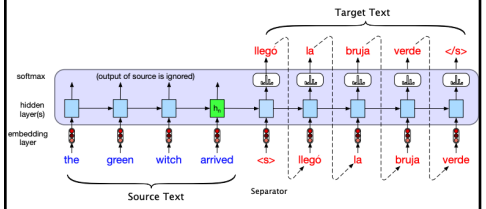


Idea:
- Process the input sentence (e.g., sentence to be translated) with a network
- Represent the sentence as some function of the hidden states (encoding)
- Use this context to generate the output

Figure 9.16 from Jurafsky and Martin

78

Encoder-decoder models: simple version

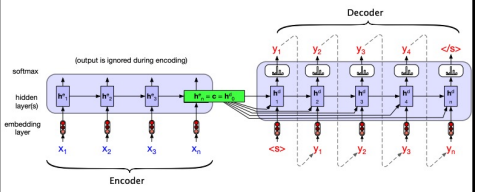


The context is the final hidden state of the encoder and is provided as input to the first step of the decoder

Figure 9.17 from Jurafsky and Martin

79

Encoder-decoder models: improved



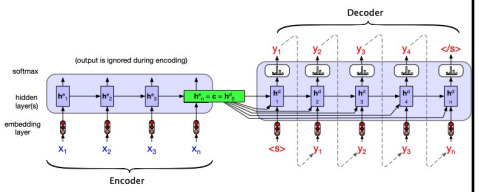
The context is some combination of **all of the** hidden states of the encoder

How is this better?

Figure 9.18 from Jurafsky and Martin

80

Encoder-decoder models: improved



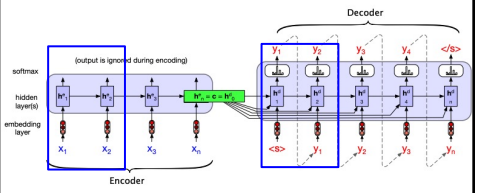
The context is some combination of **all of the** hidden states of the encoder

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

Figure 9.18 from Jurafsky and Martin

81

Encoder-decoder models: improved

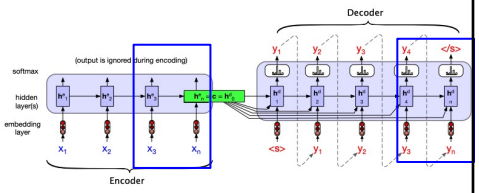


Even with this model, different encoding steps may care about different parts of the context

Figure 9.18 from Jurafsky and Martin

82

Encoder-decoder models: improved

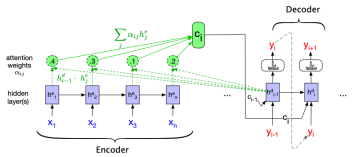


Even with this model, different encoding steps may care about different parts of the context

Figure 9.18 from Jurafsky and Martin

83

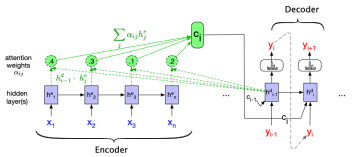
Attention



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

84

Attention

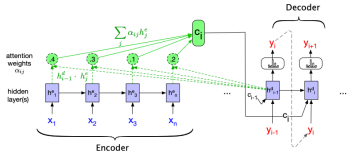


Simple version attention is static, but can learn attention mechanism (i.e., relationship between encoder and decoder hidden states)

Figure 9.23 from Jurafsky and Martin

85

Attention

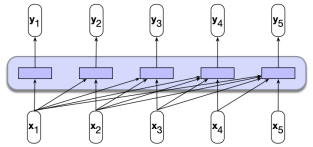


Key RNN challenge: computation is sequential
- This prevents parallelization
- Harder to model contextual dependencies

Figure 9.23 from Jurafsky and Martin

86

Another model

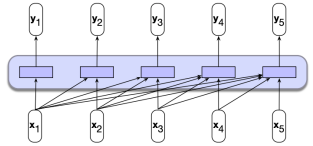


How is this setup different from the RNN?

Figure 10.1 from Jurafsky and Martin

87

Another model



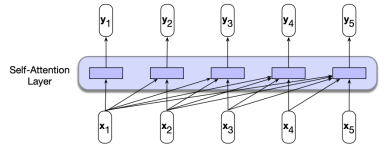
Do not rely on the hidden states for context information

Parallel: computation can all happen at once

Figure 10.1 from Jurafsky and Martin

88

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

89

Self-attention

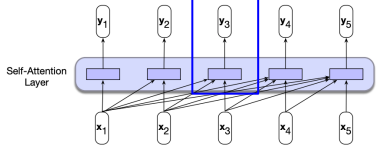


Figure 10.1 from Jurafsky and Martin

90

Transformer block

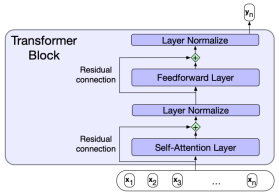
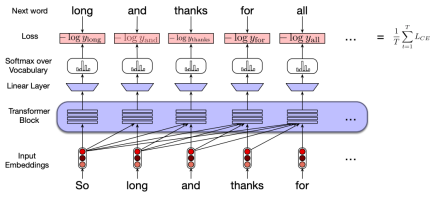


Figure 10.4 from Jurafsky and Martin

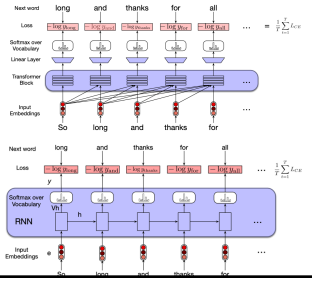
91

Transformer network



92

Transformer network vs. RNN



93

GPT

Generative: outputs things

Pre-trained: previously trained on a large corpus

Transformer: uses the transformer network

94

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

95

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]

96

Another example

https://adamharley.com/nn_vis/

97