Lecture 13: Pretraining Transformers

Instructor: Swabha Swayamdipta

USC CSCI 499 LMs in NLP

Mar 18, Spring 2024

Slides adapted from Dan Jurafsky, Chris Manning, John Hewitt, Anna Goldie

Logistics / Announcements

- Project Progress Reports have been graded!
- HW3 due Wednesday
- Graded Quiz 4 sheets available today
- Final Presentation Schedule
- Coming Soon: Mid-semester Feedback

	A	B	C	D	E
1	**Date**	**Day**	**Team 1**	**Team 2**	**Team 3**
2	Apr 17	Wed		WallESense	CuringBot
3	Apr 22	Mon	Pseudocoder	AutoRate	LLMBots
4	Apr 24	Wed	MixRx	SephoraShopper	MagicRecipe

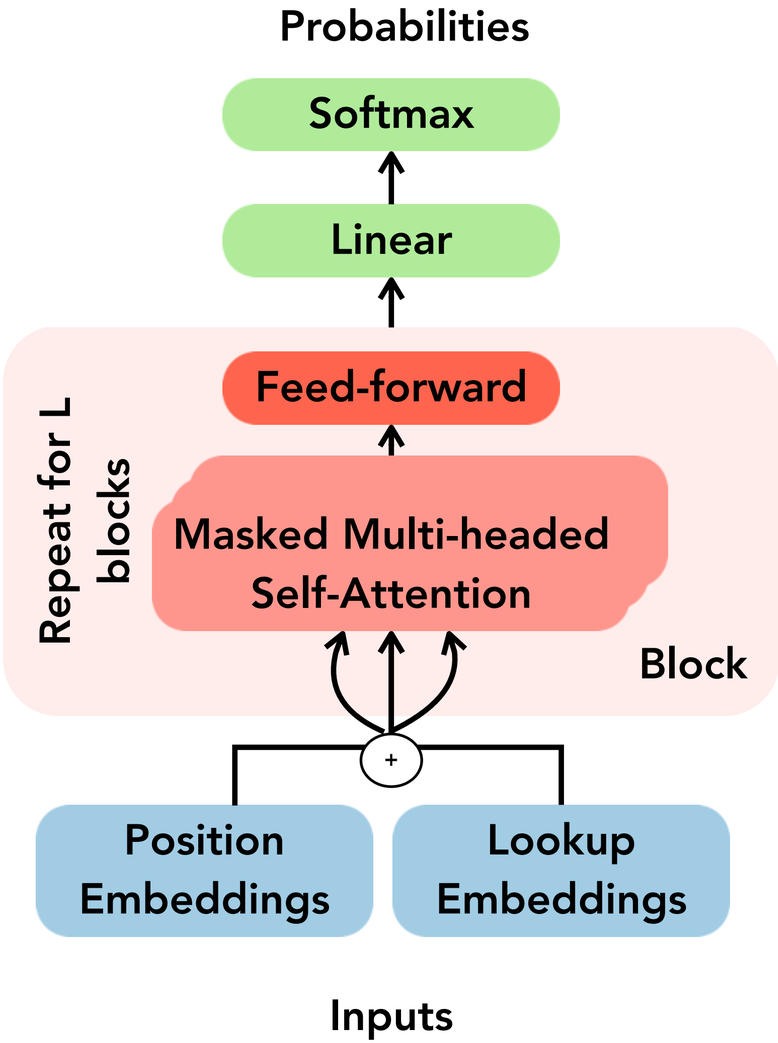
Lecture Outline

- Recap: Transformers as LMs
- Quiz 4 Answers
- The Pretraining and Finetuning Paradigm
  - Decoder-only
    - GPT-2 and Language Models
  - Encoder-only
    - BERT and Masked Language Modeling
  - Encoder-Decoder
    - T5 and sequence-to-sequence modeling
- Tokenization

Recap: Transformers as Language Models

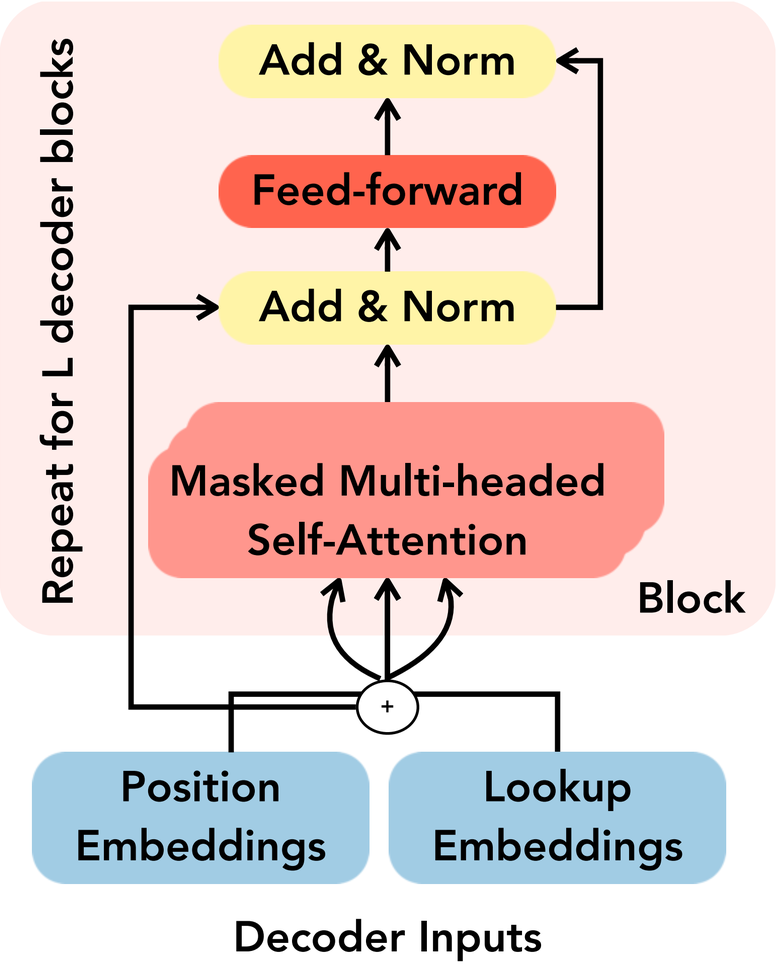
Self-Attention Transformer Building Block

- Self-attention:
  - the basis of the method; with multiple heads
- Position representations:
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- Nonlinearities:
  - At the output of the self-attention block
  - Frequently implemented as a simple feedforward network.
- Masking:
  - In order to parallelize operations while not looking at the future.
  - Keeps information about the future from "leaking" to the past.



The Transformer Model

- Transformers are made up of stacks of transformer blocks, each of which is a multilayer network made by combining feedforward networks and **self-attention layers**, the key innovation of self-attention transformers
- The Transformer Decoder-only model corresponds to
  - a Transformer language model
- Lookup embeddings can be randomly initialized (more common) or taken from existing resources such as word2vec
  - We will look at tokenization (later)

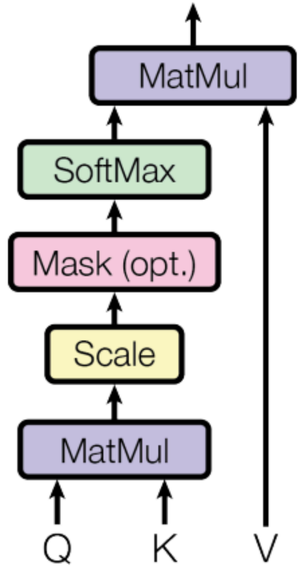


Transformer Decoder

Scaled Dot Product Attention

$$\textbf{output}_\ell = \textbf{softmax}(XQ_\ell K_\ell^T X^T) * XV_\ell$$

Scaled Dot-Product Attention



- So far: Dot product self-attention
- When dimensionality d becomes large, dot products between vectors tend to become large
- Because of this, inputs to the softmax function can be large, making the gradients small
- Now: Scaled Dot product self-attention to aid in training

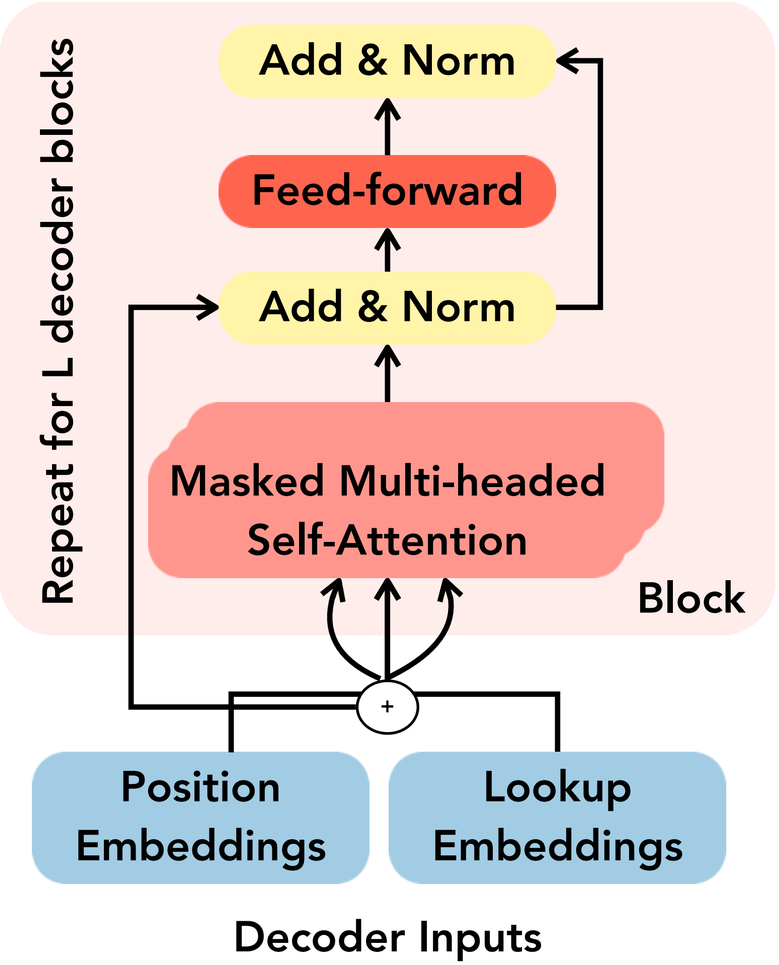
$$\textbf{output}_\ell = \textbf{softmax}\left(\frac{XQ_\ell K_\ell^T X^T}{\sqrt{d/h}}\right) * XV_\ell$$

- We divide the attention scores by d/h, to stop the scores from becoming large just as a function of d/h, where h is the number of heads

Attention is all you need (Vaswani et al., 2017)

The Transformer Decoder

- Two optimization tricks that help training:
  - Residual Connections
  - Layer Normalization
- In most Transformer diagrams, these are often written together as "Add & Norm"
  - Add: Residual Connections
  - Norm: Layer Normalization



Transformer Decoder

Residual Connections

- Original Connections: $X^{(i)} = \text{Layer}(X^{(i-1)})$ where i represents the layer
- **Residual Connections** : trick to help models train better.
  - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$
    - so we only have to learn "the residual" from the previous layer

Allowing information to skip a layer improves learning and gives higher level layers **direct access to information** from lower layers (He et al., 2016).

Layer Normalization

- Layer normalization is another trick to help models train faster
- Idea: cut down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation **within each layer**
- Let $x \in \mathbb{R}^d$ be an individual (word) vector in the model.

$$\mu = \frac{1}{d}\sum_{j=1}^{d} x_j; \quad \mu \in \mathbb{R} \qquad \sigma = \sqrt{\frac{1}{d}\sum_{j=1}^{d}(x_j - \mu)^2}; \quad \sigma \in \mathbb{R}$$

Result: New vector with zero mean and a standard deviation of one

$$\hat{x} = \frac{x - \mu}{\sigma}$$

Component-wise subtraction

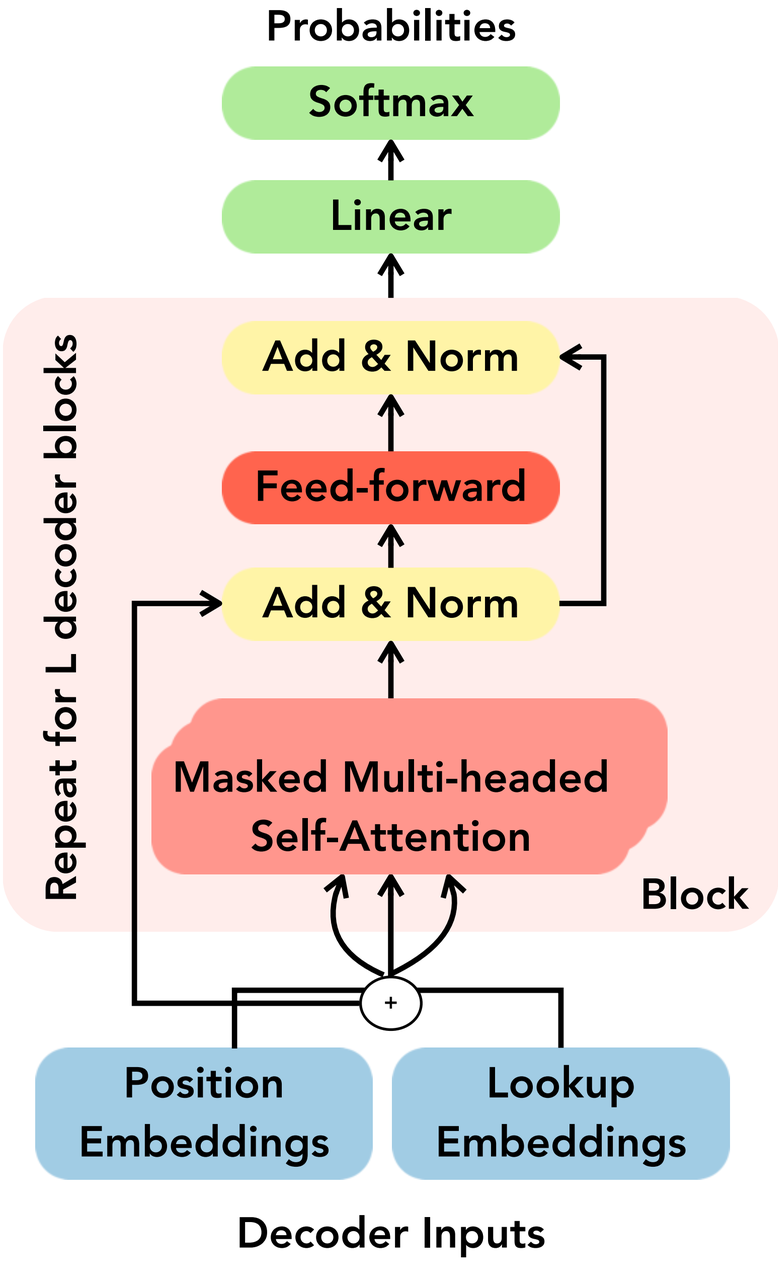
- Let $\gamma \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)

$$\textbf{LayerNorm} = \gamma\hat{x} + \beta$$

Xu et al., 2019

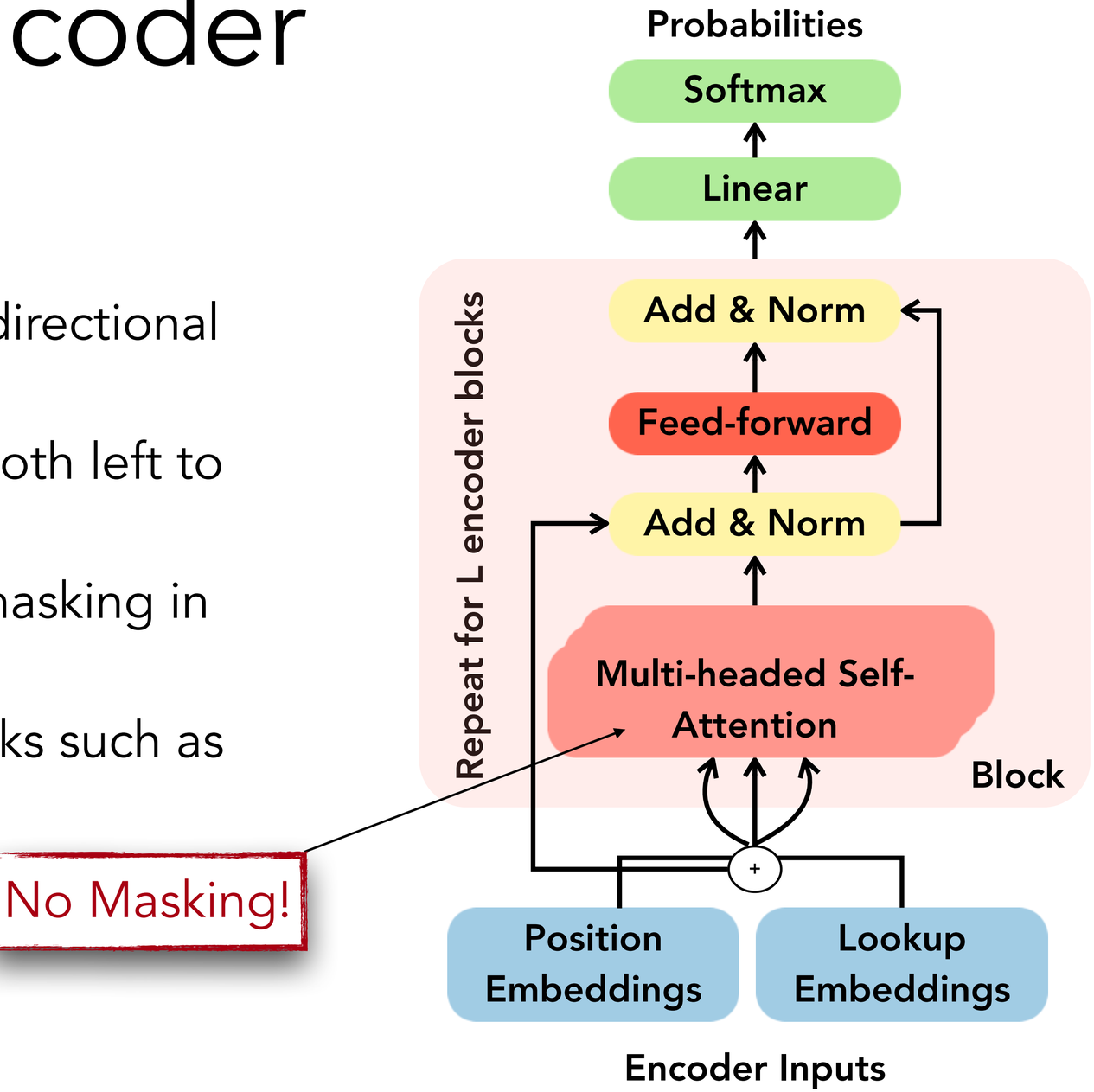
The Transformer Decoder

- The Transformer Decoder is a stack of Transformer Decoder Blocks.
- Each Block consists of:
  - Self-attention
  - Add & Norm
  - Feed-Forward
  - Add & Norm
- Output layer is as always a softmax layer



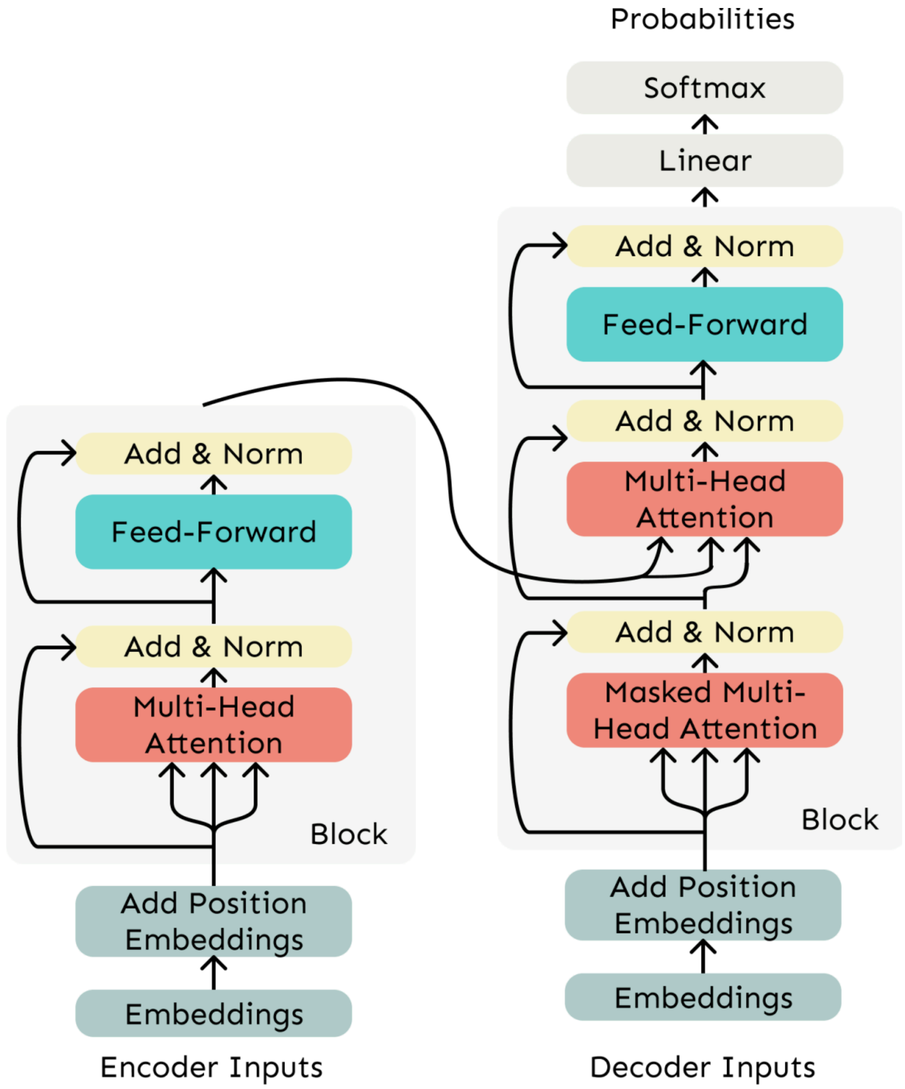
The Transformer Encoder

- The Transformer Decoder constrains to unidirectional context, as for language models.
- What if we want bidirectional context, i.e. both left to right as well as right to left?
- The only difference is that we remove the masking in the self-attention.
- Commonly used in sequence prediction tasks such as POS tagging
  - One output token y per input token x



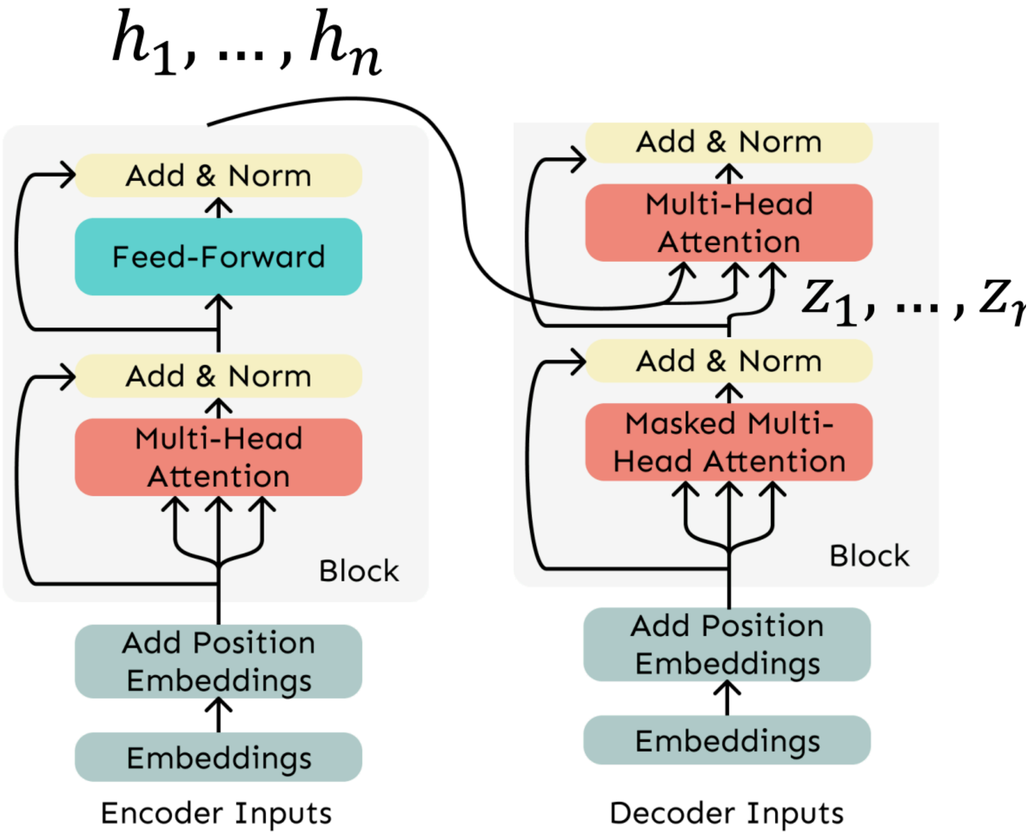
The Transformer Encoder-Decoder

- Recall that in machine translation, we processed the source sentence with a bidirectional model and generated the target with a unidirectional model.
- For this kind of seq2seq format, we often use a Transformer Encoder-Decoder.
- We use a normal Transformer Encoder.
- Our Transformer Decoder is modified to perform **cross-attention** to the output of the Encoder.

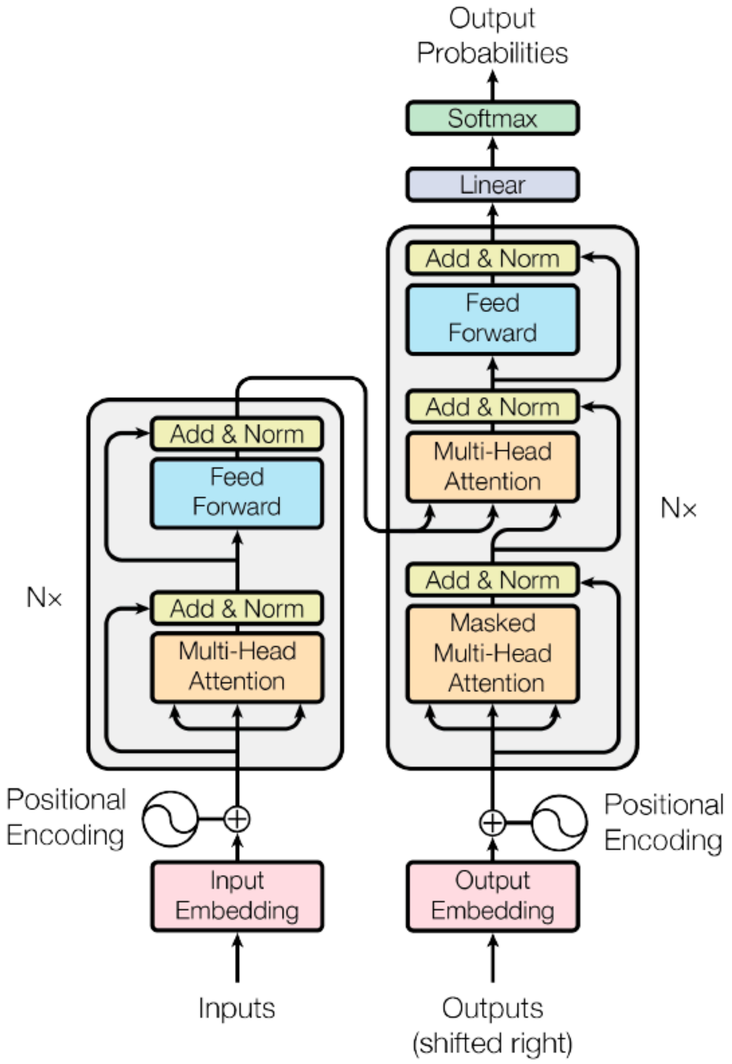


Cross Attention

- Self-attention is when keys, queries, and values come from the same source.
- In the decoder, attention looks more like the attention mechanism in encoder-decoder RNNs
- Let $\mathbf{h}_1, \ldots, \mathbf{h}_n$ be output vectors from the Transformer encoder; $\mathbf{h}_i \in \mathbb{R}^d$
- Let $\mathbf{z}_1, \ldots, \mathbf{z}_n$ be input vectors from the Transformer decoder, $\mathbf{z}_i \in \mathbb{R}^d$
- Then keys and values are drawn from the encoder (like a memory):
  - $\mathbf{k}_i = \mathbf{K}\mathbf{h}_i, \mathbf{v}_i = \mathbf{V}\mathbf{h}_i$
- And the queries are drawn from the decoder, $\mathbf{q}_i = \mathbf{Q}\mathbf{z}_i$



Transformer Diagram



Attention is all you need (Vaswani et al., 2017)

Transformers: Performance

Machine Translation

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Language Modeling

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

The real power of Transformers comes from pretraining language models which are then adapted for different tasks

Quiz 4 Answers
Redacted

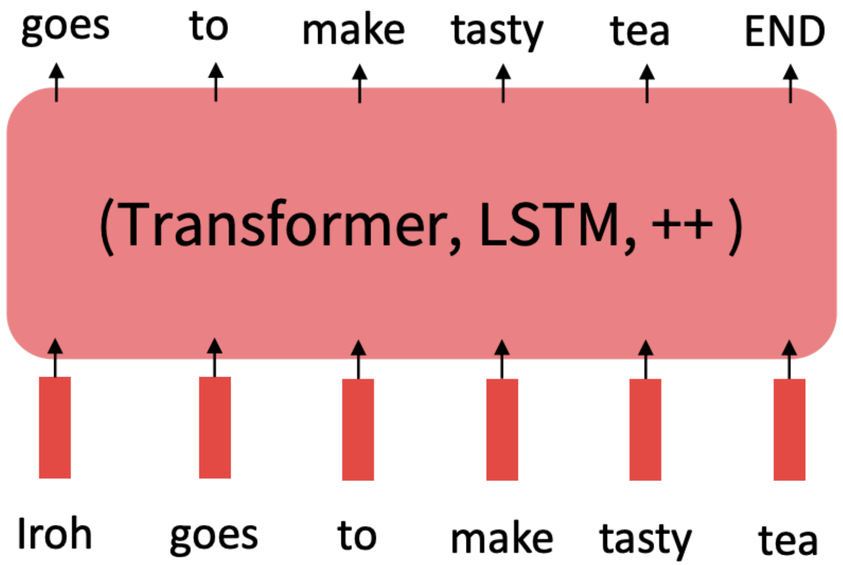
The Pre-training and Fine-tuning Paradigm

The Pretraining / Finetuning Paradigm

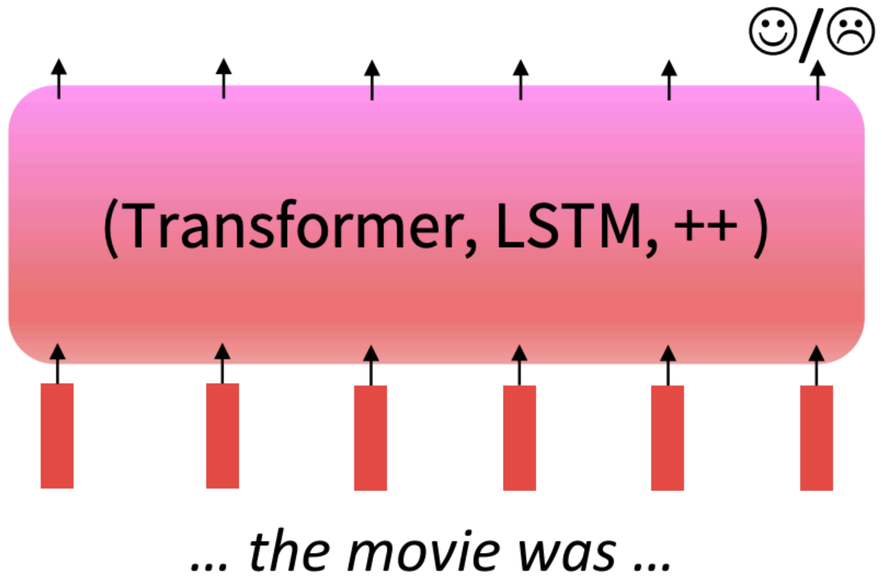
- Pretraining can improve NLP applications by serving as parameter initialization.

Key idea: "Pretrain once, finetune many times."

Step 1: Pretrain (on language corpora)
Lots of text; learn general things!



Step 2: Finetune (on your task data)
Not many labels; adapt to the task!

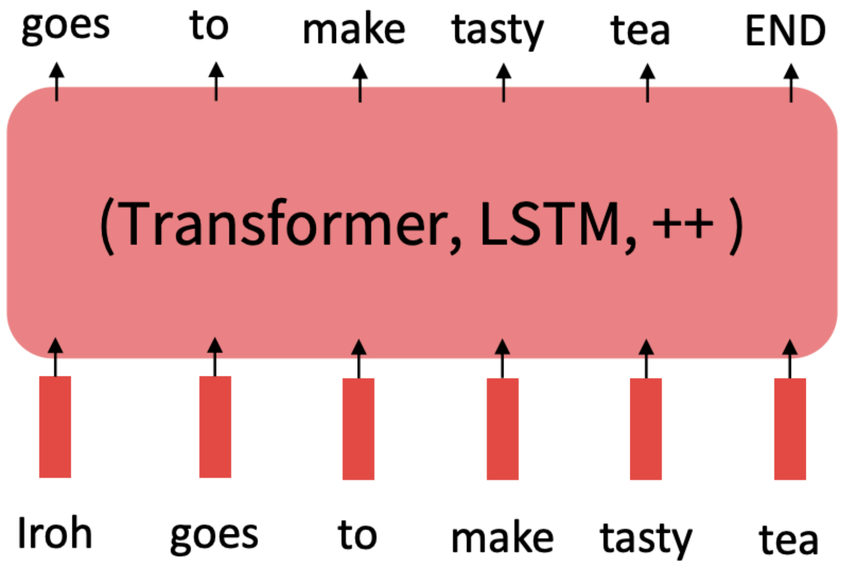


Pretraining

- Central Approach: Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.
- Used for parameter initialization
  - Part of network
  - Full network
- Abstracts away from the task of "learning the language"

Step 1: Pretrain (on language corpora)

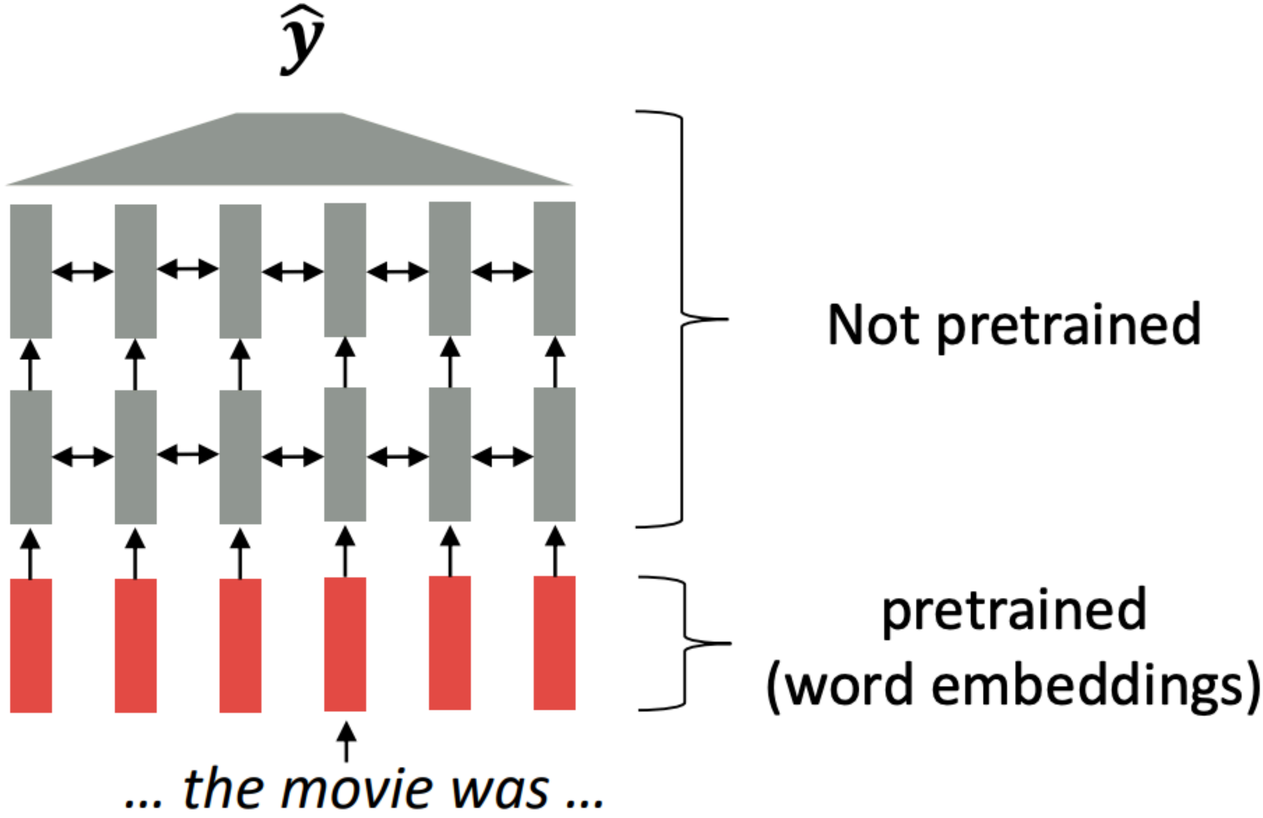
Lots of text; learn general things!



Word embeddings were pretrained too!

Previously:

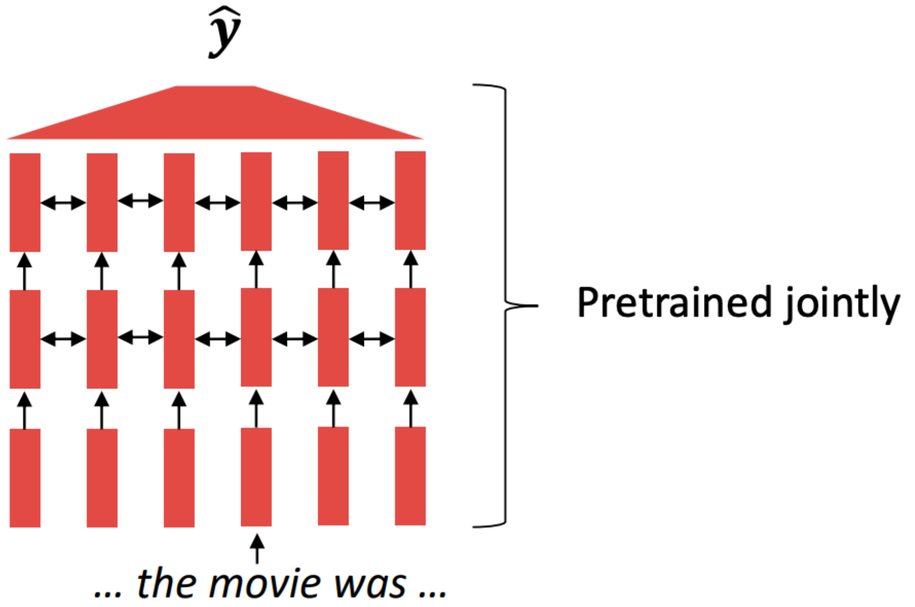
- Start with pretrained word embeddings
  - word2vec
  - GloVe
  - Trained with limited context (windows)
- Learn how to incorporate context in an LSTM or Transformer while training on the task (e.g. sentiment classification)
- Paradigm till 2017



However, the word “movie” gets the same word embedding, no matter what sentence it shows up in!

Pretraining Entire Models

- In modern NLP:
  - All (or almost all) parameters in NLP networks are initialized via pretraining.
  - This has been exceptionally effective at building strong:
    - representations of language
    - parameter initializations for strong NLP models.
    - probability distributions over language that we can sample from



[This model has learned how to represent entire sentences through pretraining]

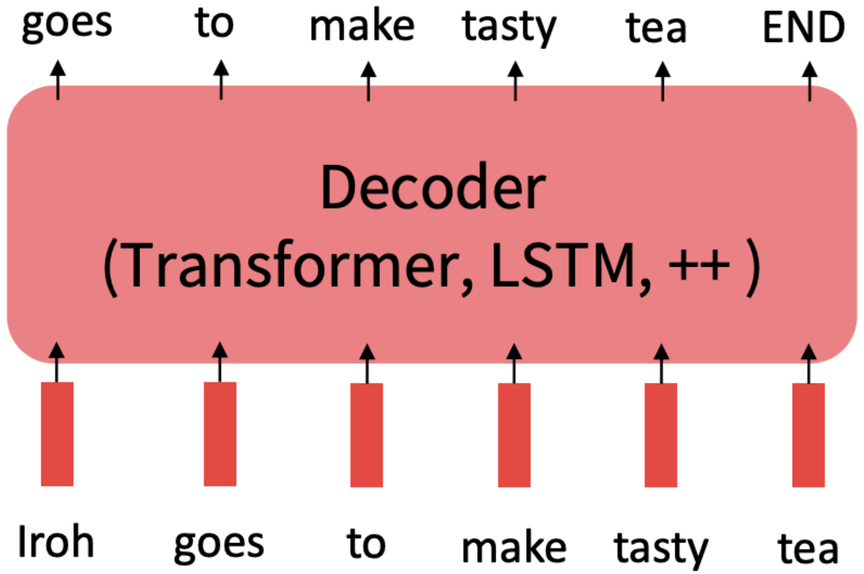
Pretraining: Intuition from SGD

Why should pretraining and finetuning help, from a "training neural nets" perspective?

- Pretraining provides parameters $\hat{\theta}$ by approximating $\min_{\theta} \mathscr{L}_{\text{pretrain}}(\theta)$
  - $\mathscr{L}_{\text{pretrain}}(\theta)$ is the pretraining loss
- Then, finetuning approximates $\min_{\theta} \mathscr{L}_{\text{finetune}}(\theta)$, **but starting at** $\hat{\theta}$.
  - $\mathscr{L}_{\text{finetune}}(\theta)$ is the finetuning loss
- The pretraining may matter because stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during finetuning
  - It is possible that the finetuning local minima near $\hat{\theta}$ tends to generalize well!
  - And/or, maybe the gradients of finetuning loss near $\hat{\theta}$ propagate nicely!

Pretraining: Language Models

- Recall the language modeling task:
  - Model $p_\theta(w_t \mid w_{1:t-1})$, the probability distribution over words given their past contexts.
  - There's lots of data for this! (In English.)
- Pretraining through language modeling:
  - Train a neural network to perform language modeling on a large amount of text.
  - Save the network parameters.



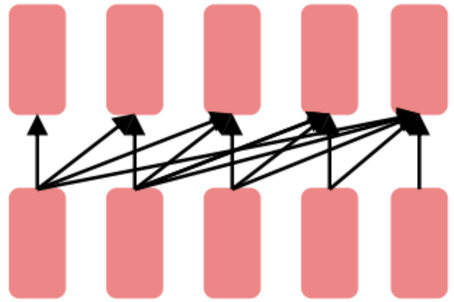
Semi-supervised Sequence Learning

Andrew M. Dai
Google Inc.
adai@google.com

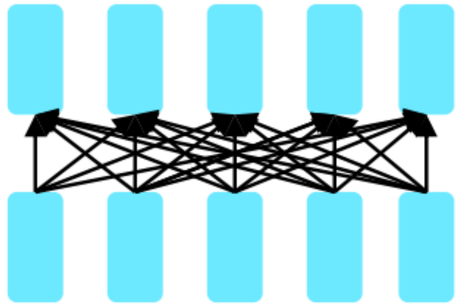
Quoc V. Le
Google Inc.
qvl@google.com

Pretraining

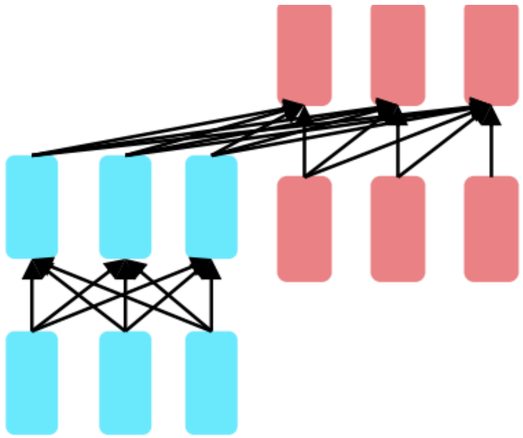
- Not restricted to language modeling! Can be any task
- But most successful if the task definition is very general. Hence, language modeling is a great pretraining option
- Three options!



Decoders

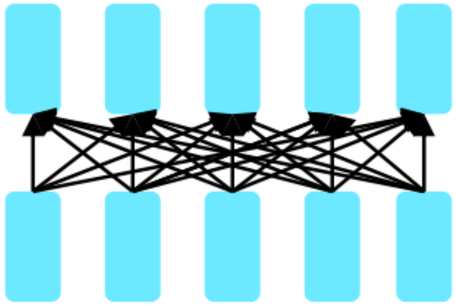


Encoders



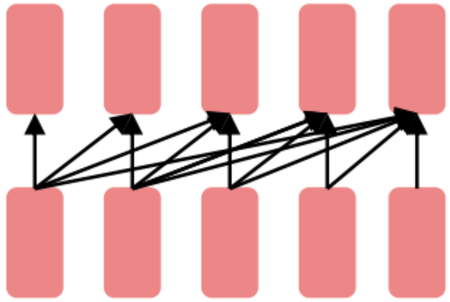
Encoder-Decoders

Pretraining for three types of architectures



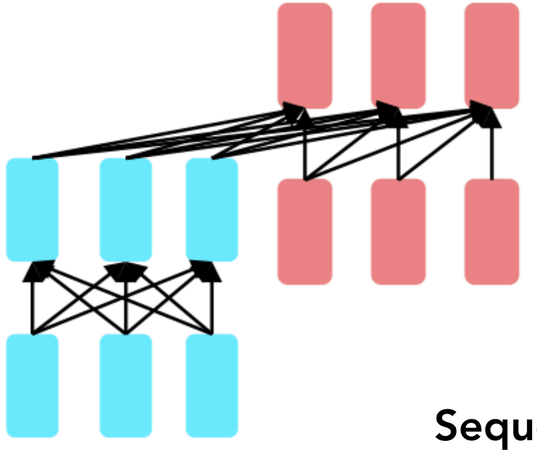
Encoders

Bidirectional Context



Decoders

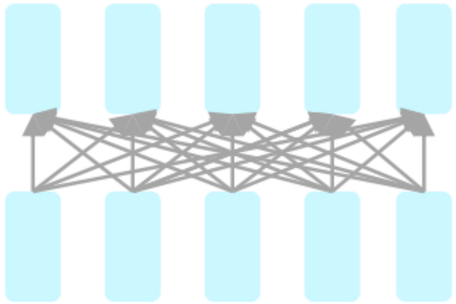
Language Models



Encoder-Decoders

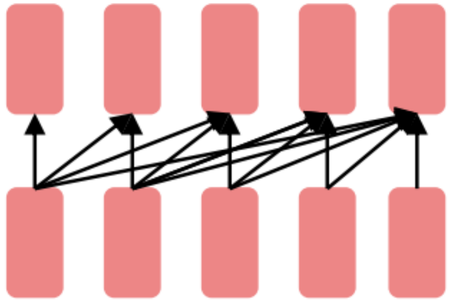
Sequence-to-sequence

Pretraining for three types of architectures



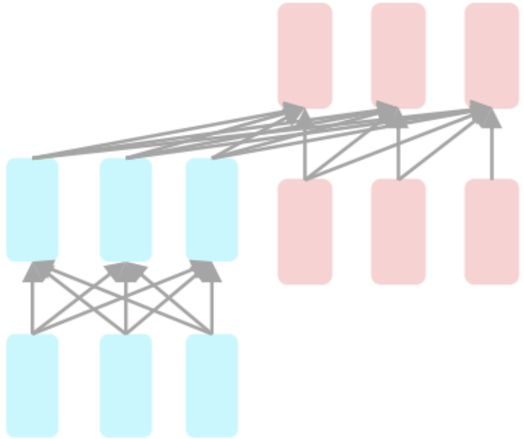
Encoders

Bidirectional Context



Decoders

Language Models

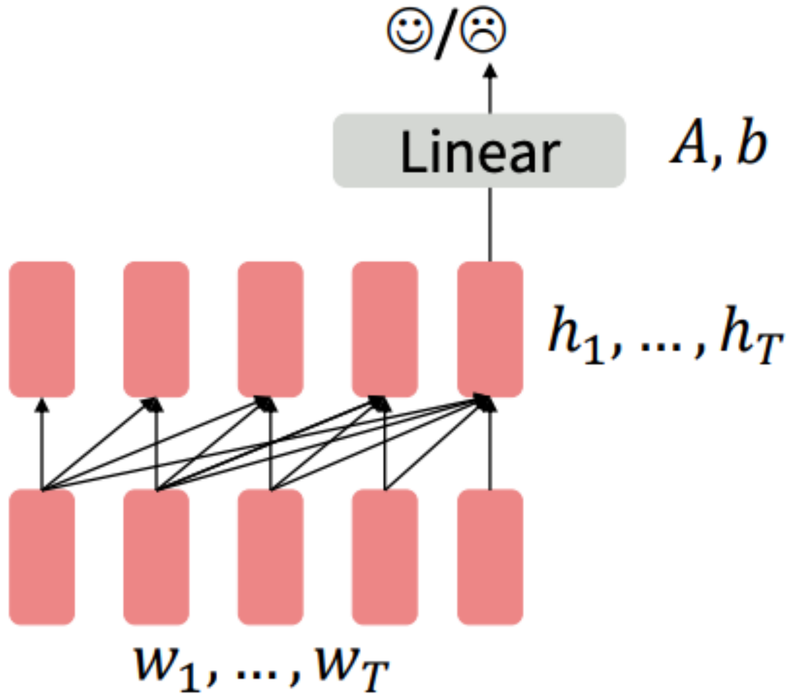


Encoder-Decoders

Sequence-to-sequence

Pretraining Decoders: Classifiers

- When using language model pretrained decoders, we can ignore that they were trained to model $p_\theta(w_t \mid w_{1:t-1})$
- We can finetune them by training a classifier on the last word's hidden state
  - $h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$
  - $y \approx A h_T + b$
  - Where A and b are randomly initialized and specified by the downstream task.
- Gradients backpropagate through the whole network.



The linear layer hasn't been pretrained and must be learned from scratch.

Generative Pretrained Transformer (GPT)

- 2018's GPT was a big success in pretraining a decoder!
  - Transformer decoder with 12 layers, 117M parameters.
  - 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
  - Byte-pair encoding with 40,000 merges
  - Trained on BooksCorpus: over 7000 unique books.
    - Contains long spans of contiguous text, for learning long-distance dependencies.
  - The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"

OpenAI

[Radford et al., 2018]

Adapting GPT

- How do we format inputs to our decoder for finetuning tasks?
- Natural Language Inference: Label pairs of sentences as entailing/contradictory/neutral

  - Premise: The man is in the doorway
  - Hypothesis: The person is near the door

Entailment

- Radford et al., 2018 evaluate on natural language inference by formatting the input as a sequence of tokens for the decoder
  - [START] The man is in the doorway [DELIM] The person is near the door [EXTRACT]
  - The linear classifier is applied to the representation of the [EXTRACT] token.

[Radford et al., 2018]

GPT: Results on Classification

- Outperforms Recurrent Neural Nets

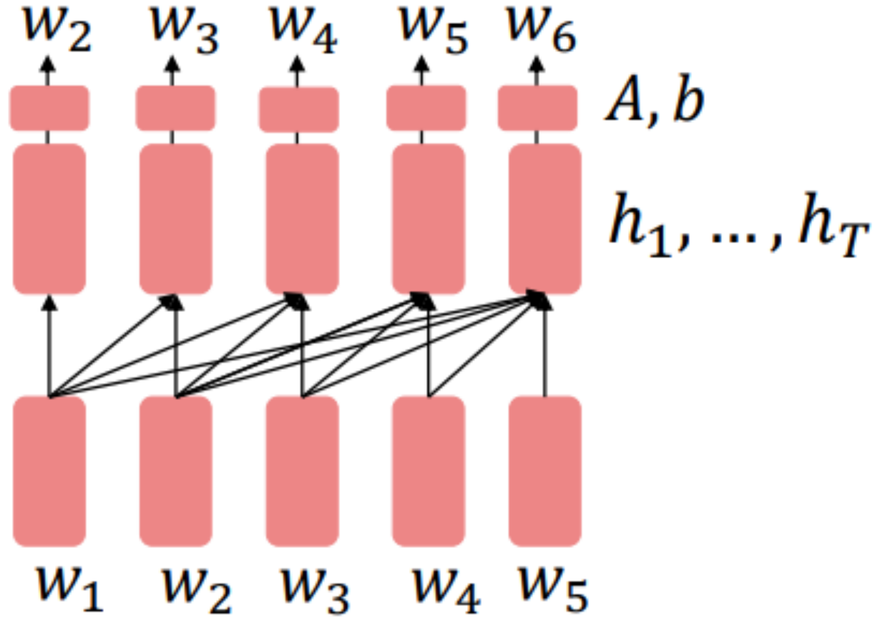
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

[Radford et al., 2018]

Pretraining Decoders: Generators

- More natural: pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t \mid w_{1:t-1})$
  - $h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$
- $w_t \approx Ah_{t-1} + b$
- Where A, b were pretrained in the language model!

- This is helpful in tasks where the output is a sequence with a vocabulary like that at pretraining time!
  - Dialogue (context=dialogue history)
  - Summarization (context=document)



The linear layer has been pretrained

GPT-2

- GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.
- Moved away from classification, only generation

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

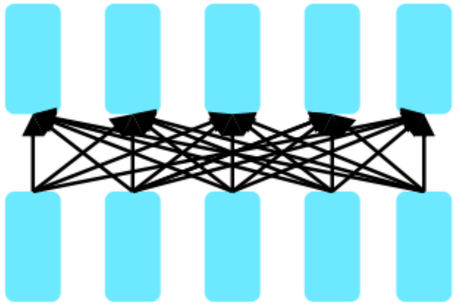
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT-3 and beyond

- Markedly improved generation capabilities by greatly increasing data and model scale
  - Other tricks in GPT-3.5 and above: RLHF (Reinforcement Learning with Human Feedback)
- Solving all tasks through generation, even obviating the need to fine-tune!
  - Instruction tuning

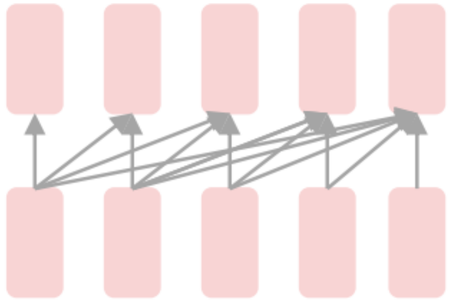
More in future weeks!

Pretraining for three types of architectures



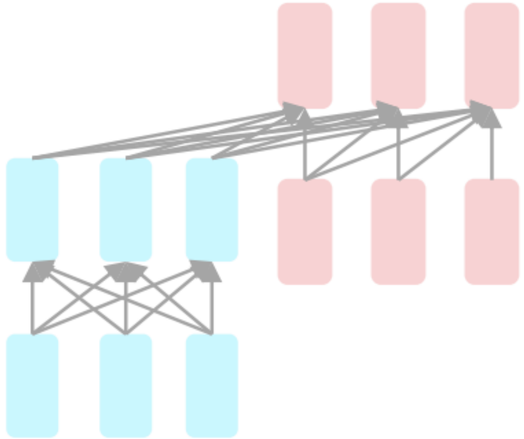
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

Sequence-to-sequence

Pretraining Encoders: Bidirectional Context

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____

Universal Studios Theme Park is located in _____, California

Problem: Input Reconstruction

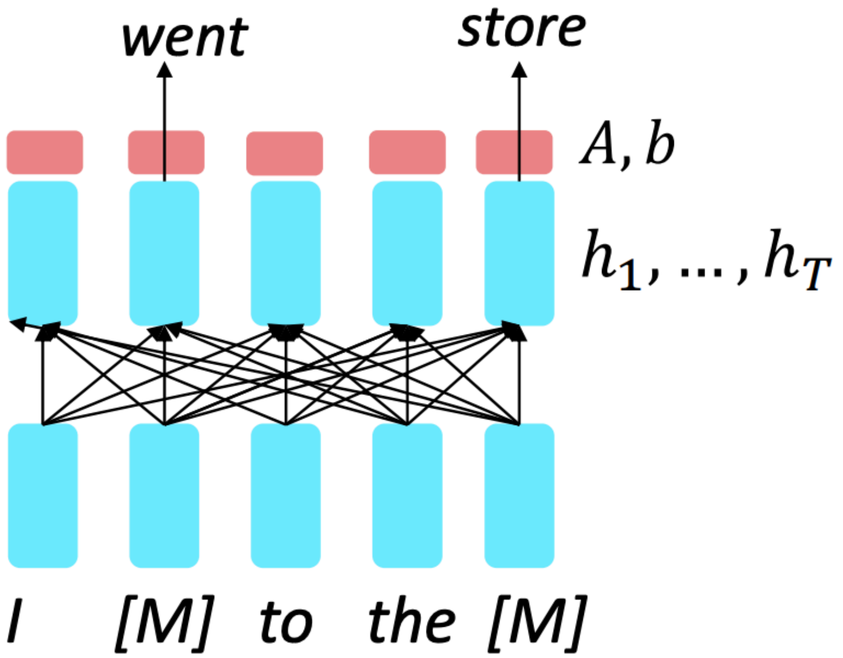
'Cause darling i'm a _____ dressed like a daydream

Bidirectional context is important to reconstruct the input!

Pretraining Encoders: Objective

- Encoders get bidirectional context, so we can't do language modeling!
- Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.
  - $h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$
  - $y_i \approx Ah_i + b$

- Only add loss terms from words that are "masked out."
- If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(\tilde{x} \mid x)$.
- Called Masked LM
- Special type of language modeling

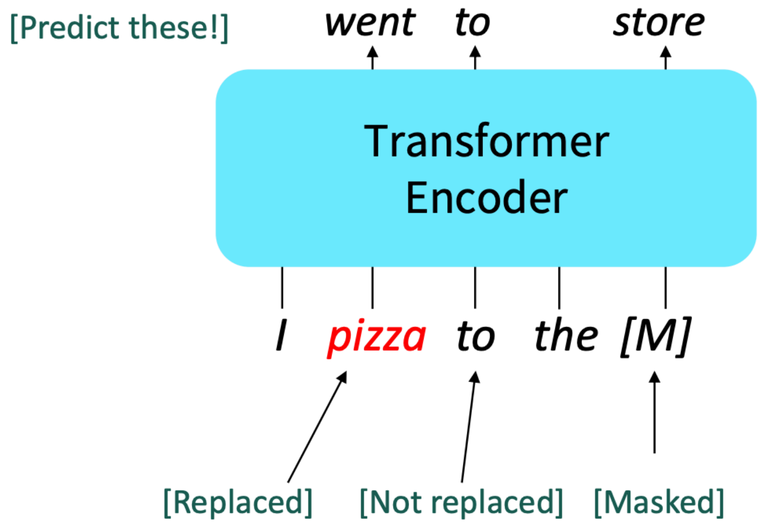


Masked Language Modeling

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and released BERT, a Transformer, pretrained to:

- 15% of the input tokens in a training sequence are sampled for learning, these are to be predicted by the model
- Of these
  - 80% are replaced with [MASK]
  - 10% are replaced with randomly selected tokens,
  - Remaining 10% are left unchanged



Why?

Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)