Lecture 15:
Pre-training and Fine-tuning Transformers

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Oct 17, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning

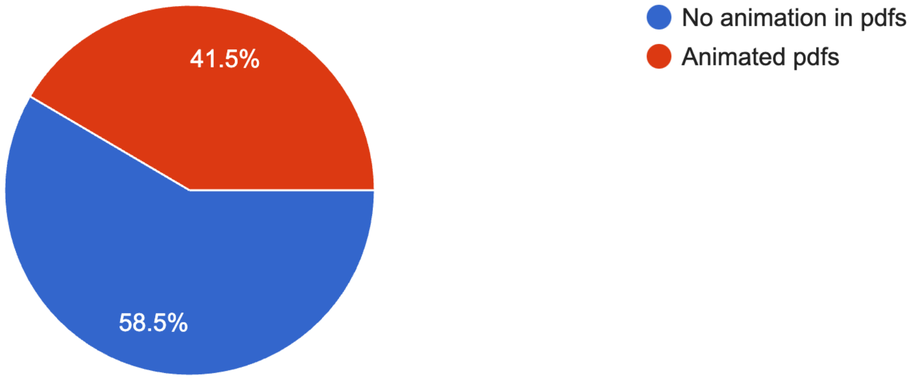
Announcements

- HW2 and midterms are being graded
  - Class standing / distributions in the next class
- Upcoming deadlines:
  - Tue, 10/22 - HW3
  - Fri, 10/25 - Project Progress Report
    - What are we expecting? See class website: https://swabhs.com/f24-csci544-appliednlp/details/project/
      - once again describe the project's goals
      - contain all details on the dataset (your dataset should mostly be collected by this time),
      - contain some initial results (think of this as a motivating results), and
      - must outline a concrete plan of what will be done before the final report.
  - Tue, 10/29 - Quiz 4
- Thanks for the feedback!
  - Overwhelmingly positive (thank you!!)
    - We heard your requests for Video lectures and made them available
  - Some of you cannot hear questions that others ask. I will try my best to remember to repeat questions in class

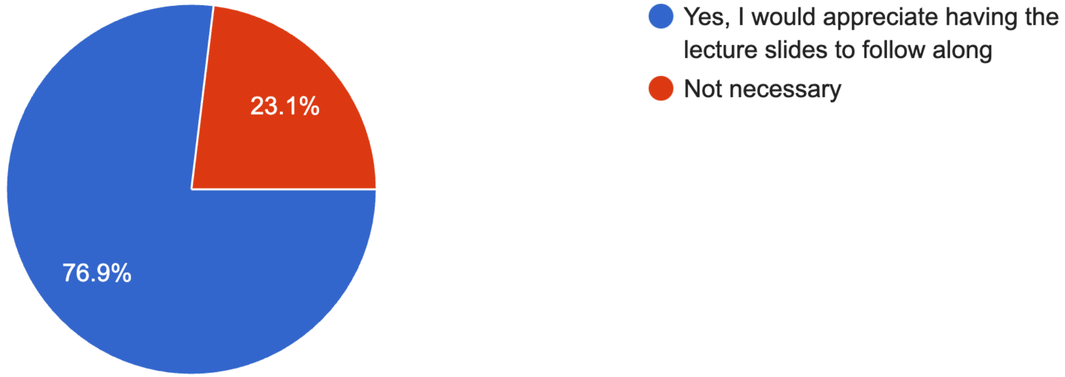
Thanks for the feedback!

Do you find the recap of the previous lecture in the beginning of each lecture useful?
65 responses

Do you like seeing animation builds in the lecture pdfs (makes the pdf much larger), or would you prefer no animation in the lecture pdfs?
65 responses



Is the availability of lecture slides before class necessary for your learning during class ?
65 responses

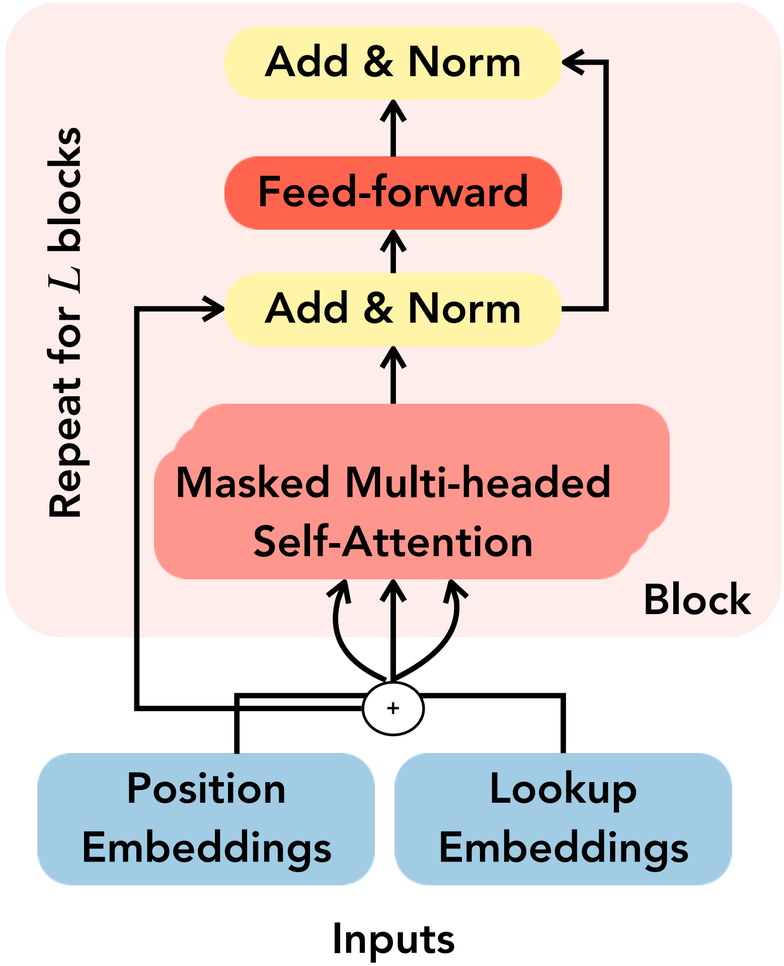


Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

Recap: Transformers as Encoders, Decoders, Encoder-Decoders

The Transformer Model



Residual Connections

Original: 

- Original Connections: $X^{(i)} = \text{Layer}(X^{(i-1)})$ where i represents the layer
- **Residual Connections** : trick to help models train better.
  - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$
    - Helps learn "the residual" from the previous layer
    - Remember: the layer contains all the non-linearities

Easier gradient flow, easier learning

Add: 

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

- 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**
  - **thus making the hidden state values more stable**
- 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

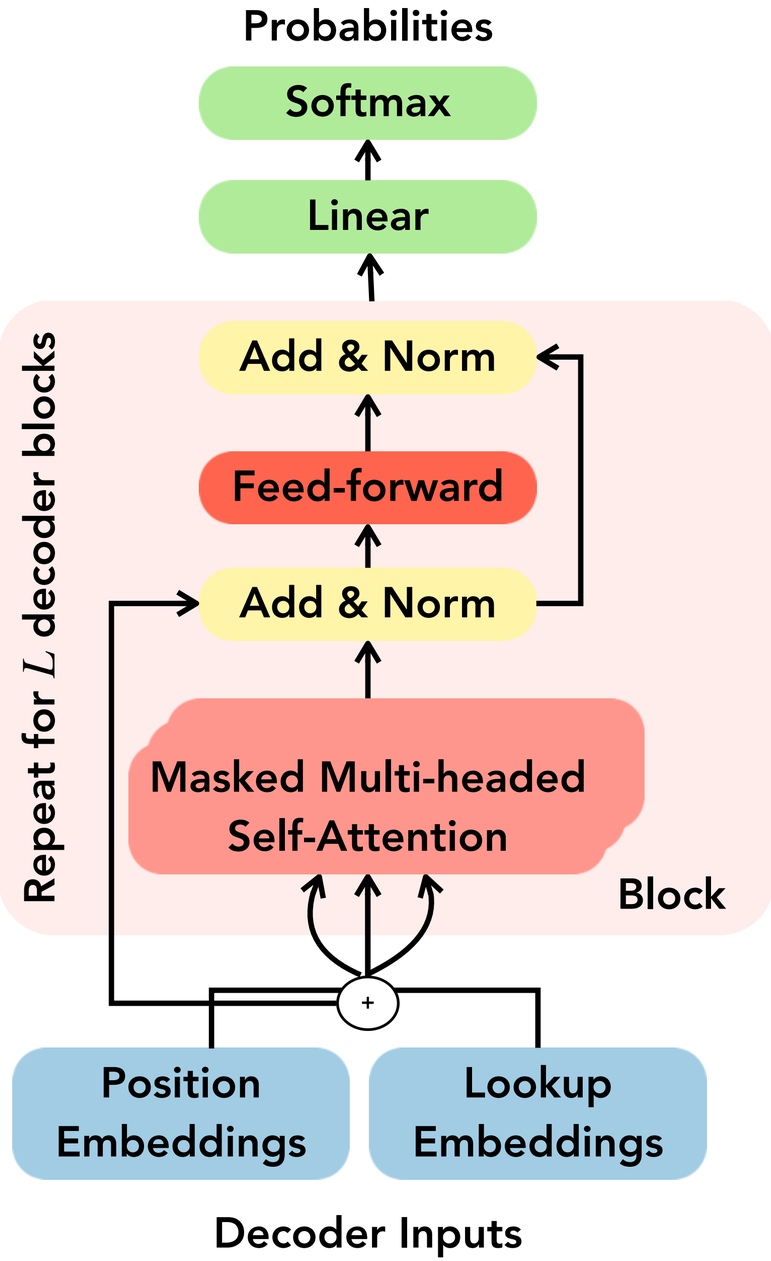
- Let $\gamma \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$ be learned "gain" and "bias" parameters. (Can omit!)

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

Xu et al., 2019

The Transformer Decoder

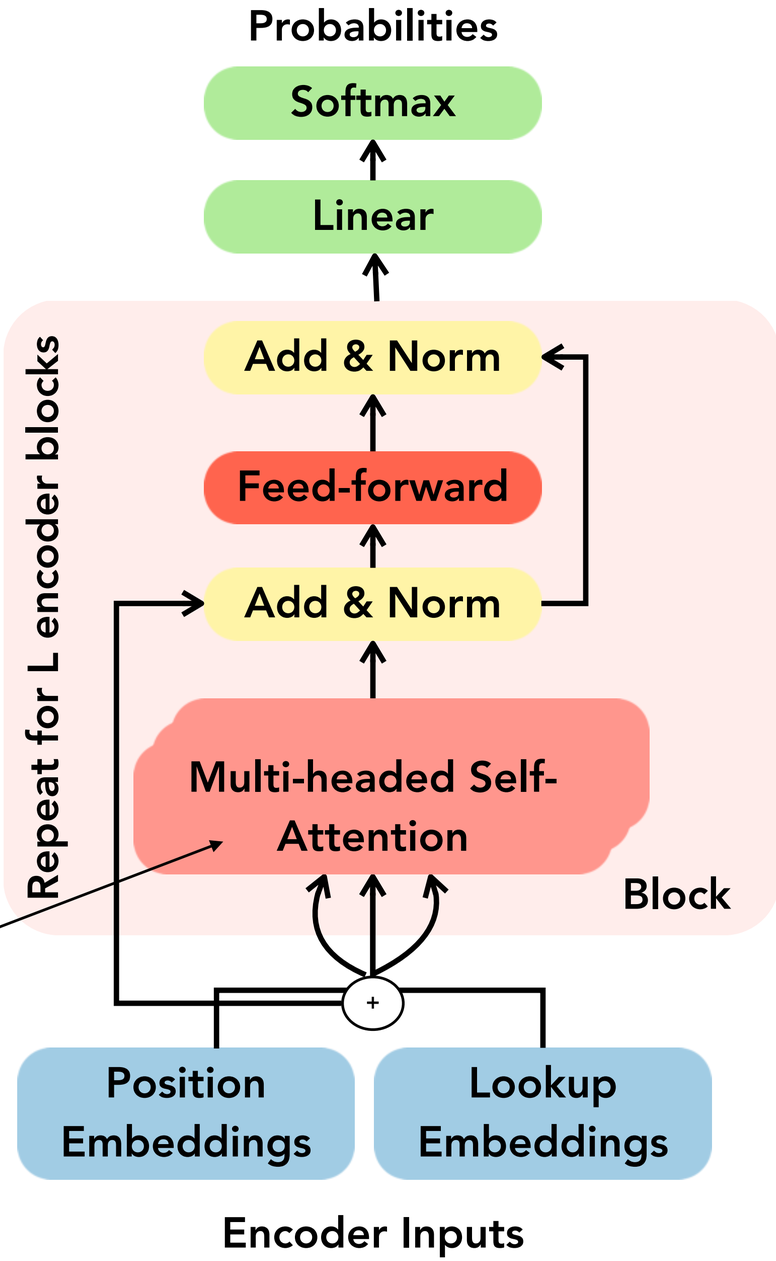
- The Transformer Decoder is a stack of Transformer Decoder Blocks.
- Each Block consists of:
  - Self-attention
  - Add & Norm
  - Feed-Forward
  - Add & Norm
- Output layer is as always a softmax layer



The Transformer Encoder

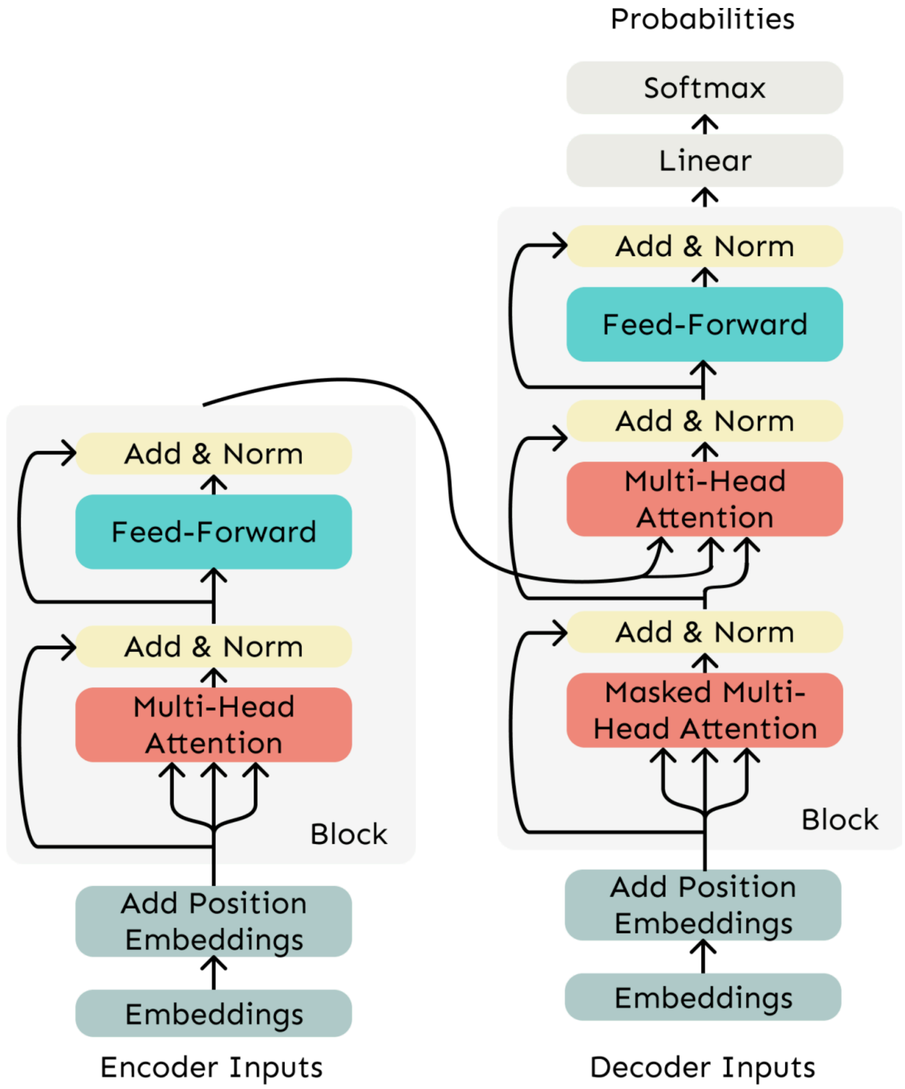
- The Transformer Decoder constrains to unidirectional context, as for language models.
- What if we want bidirectional context, i.e. both left to right as well as right to left?
- The only difference is that we remove the masking in the self-attention.
- Commonly used in sequence prediction tasks such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

No Masking!



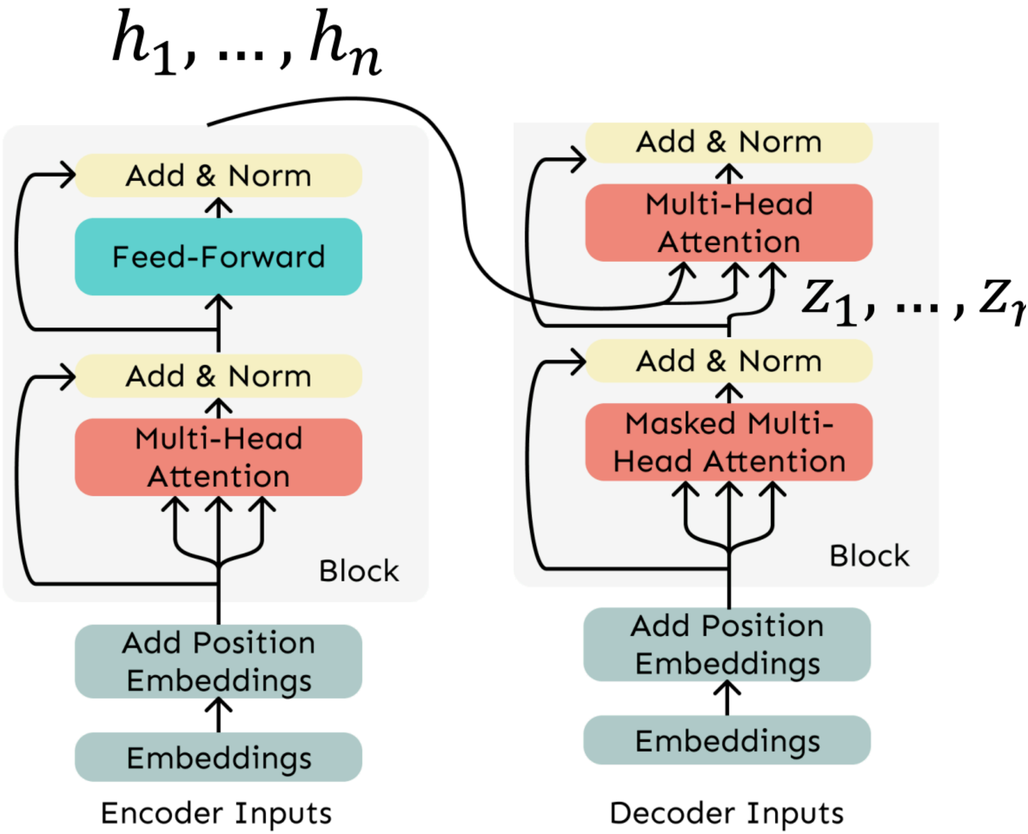
The Transformer Encoder-Decoder

- 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.

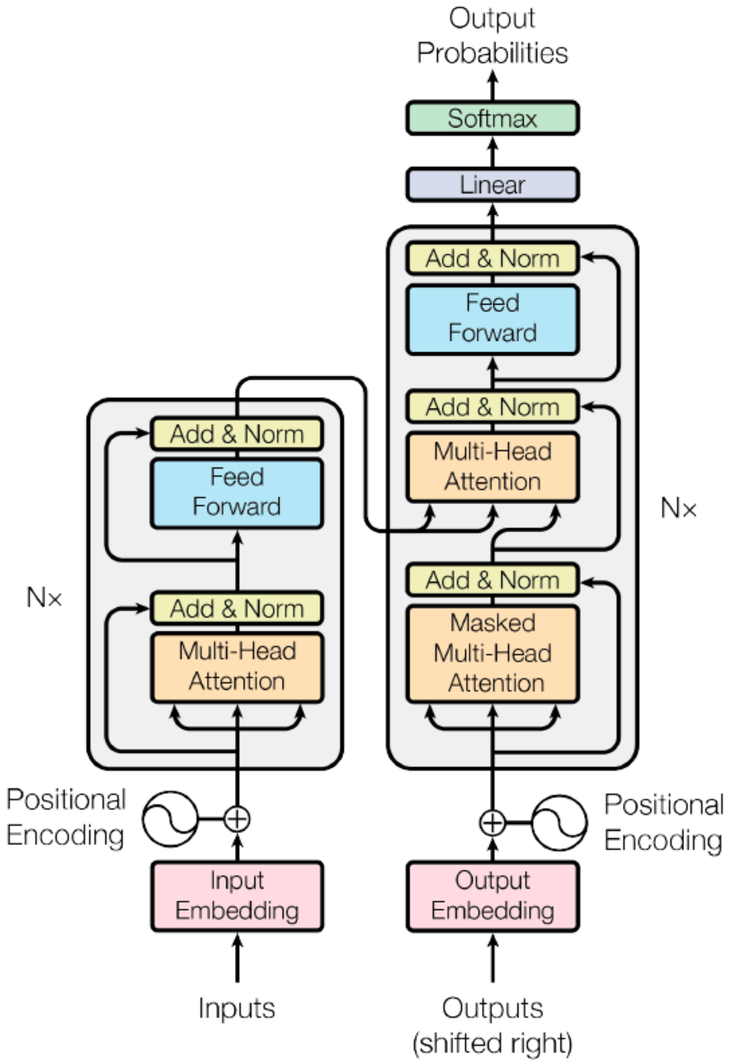


Cross Attention

- We saw that self -attention is when keys, queries, and values come from the same source.
- In the decoder, we have attention that looks more like what we saw last week.
- 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, $\mathbf{h}_i \in \mathbb{R}^d$
- 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)

Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

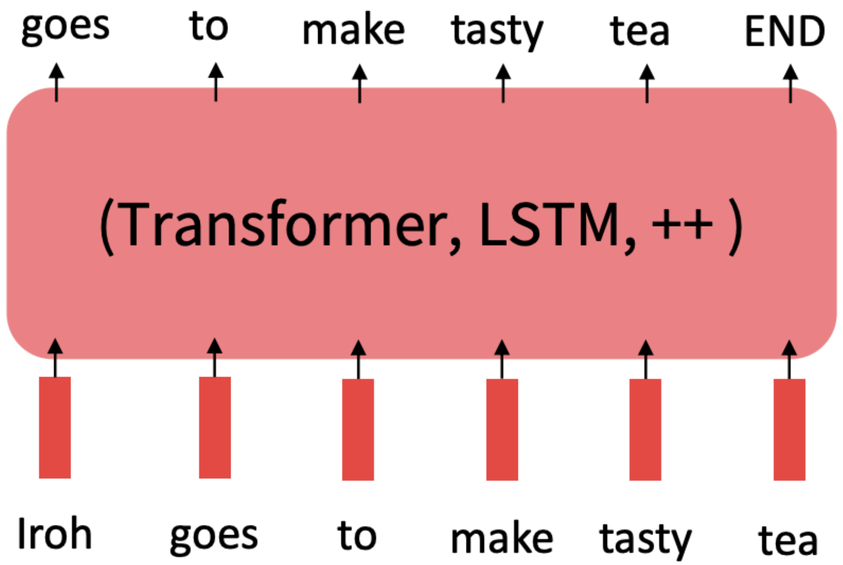
The Pre-training and Fine-tuning Paradigm

The Pretraining / Finetuning Paradigm

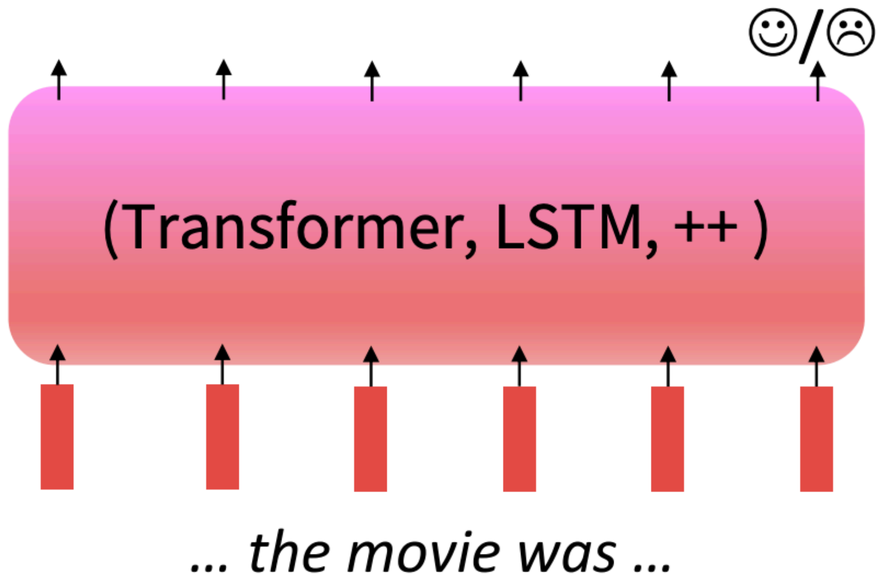
- Pretraining can improve NLP applications by serving as parameter initialization.

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!

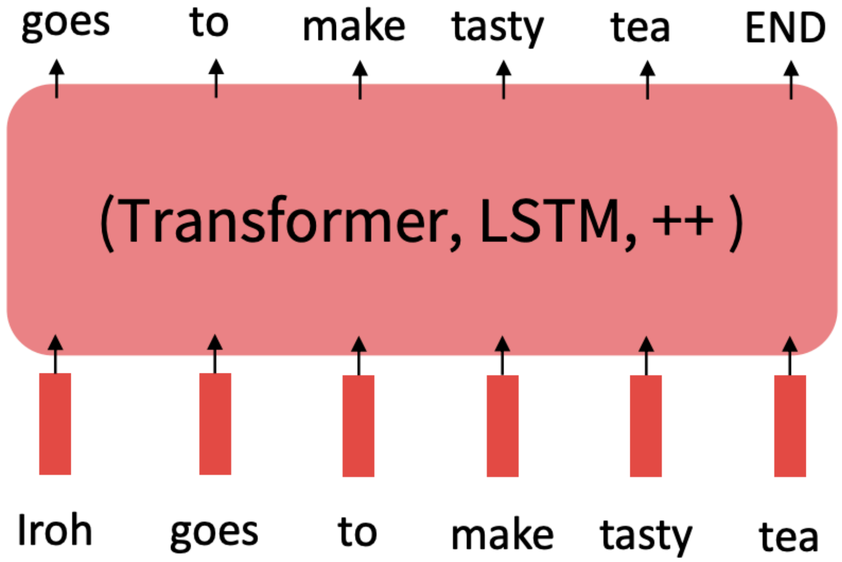


Pretraining

- 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)

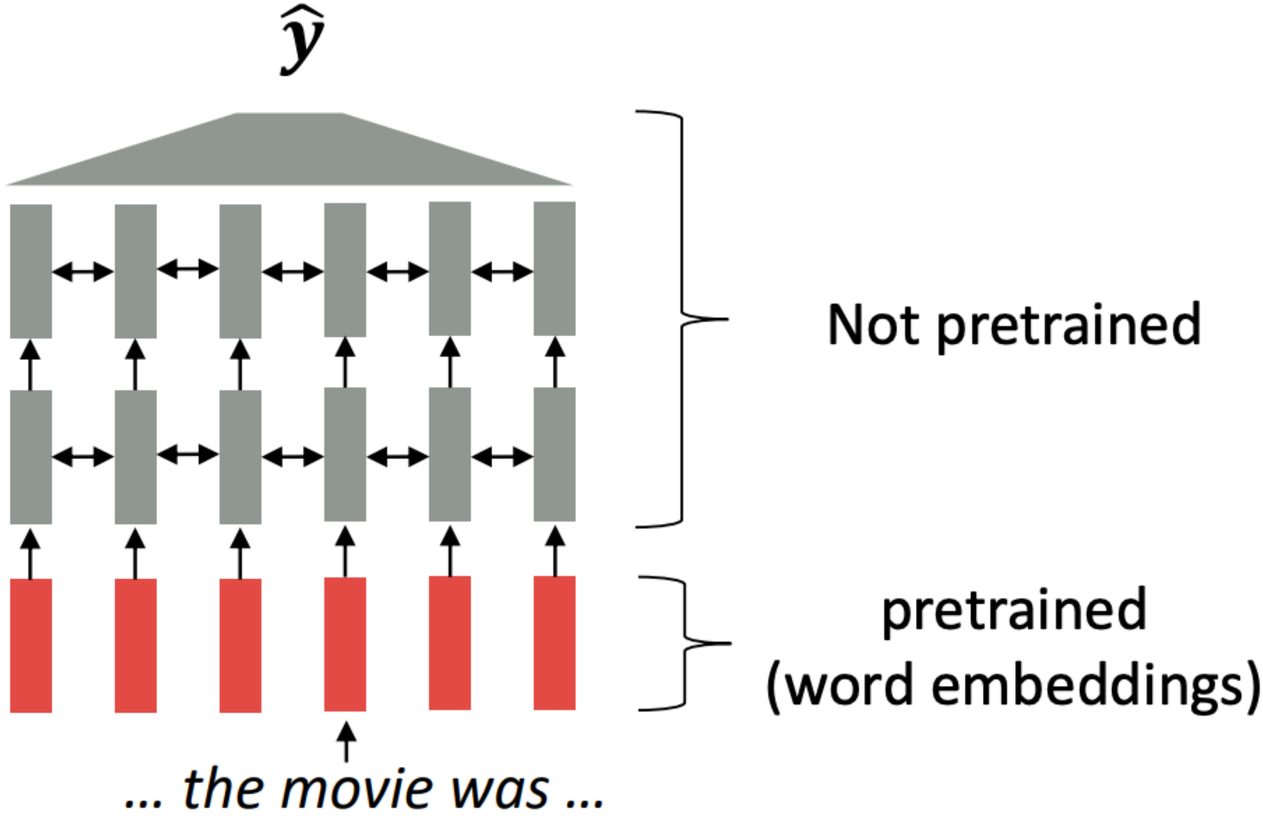
Lots of text; learn general things!



Word embeddings were pretrained too!

Previously:

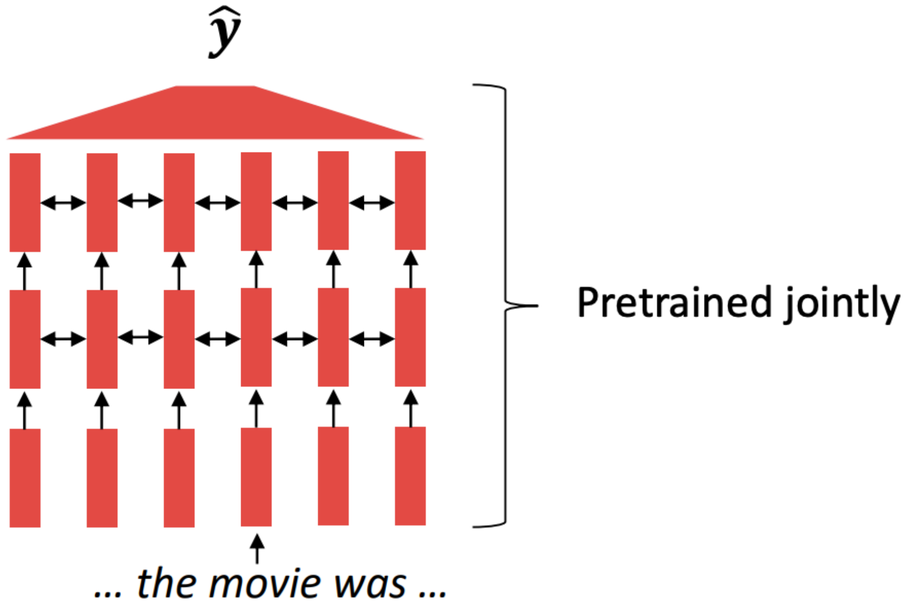
- Start with pretrained word embeddings
  - word2vec
  - GloVe
  - Trained with limited context (windows)
- Learn how to incorporate context in an LSTM or Transformer while training on the task (e.g. sentiment classification)
- Paradigm till 2017



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]

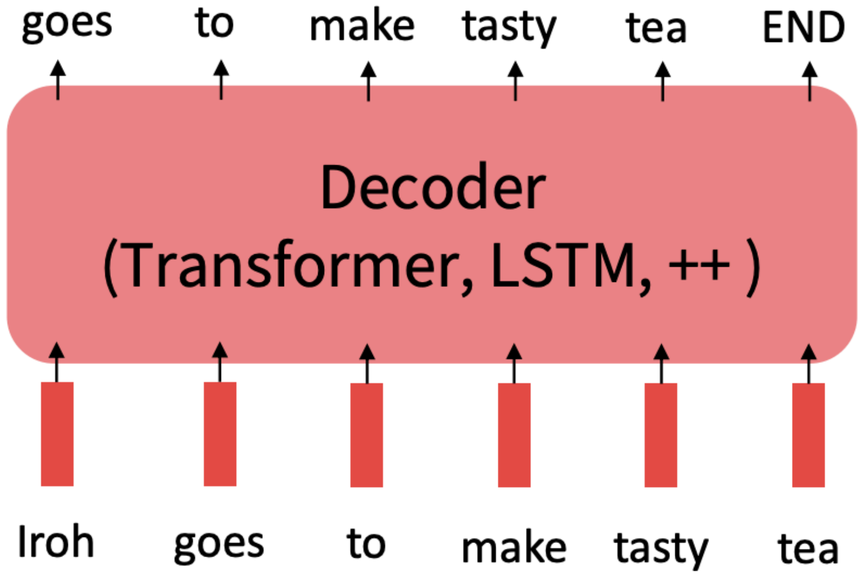
Pretraining: Intuition from SGD

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.



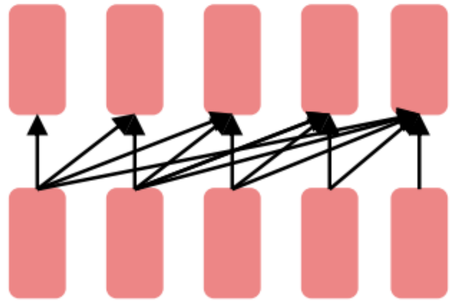
Semi-supervised Sequence Learning

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

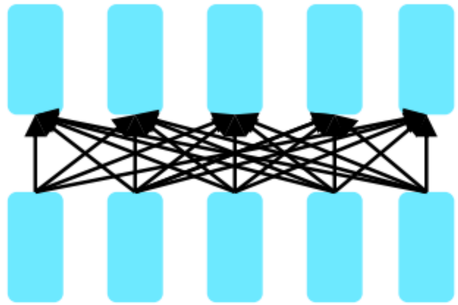
Quoc V. Le
Google Inc.
qvl@google.com

Pretraining

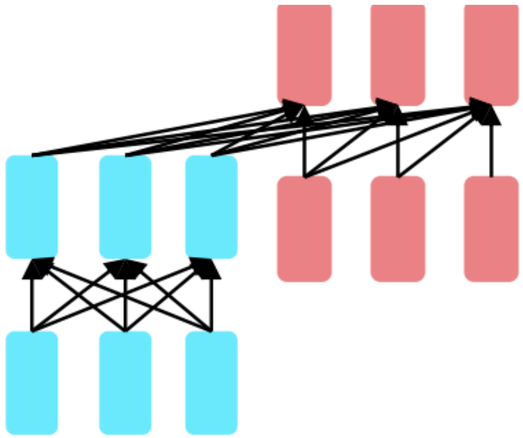
- Not restricted to language modeling! Can be any task
- But most successful if the task definition is very general. Hence, language modeling is a great pretraining option
- Three options!



Decoders

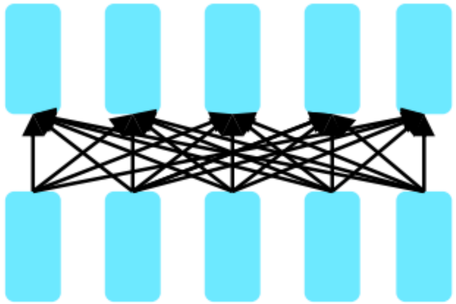


Encoders



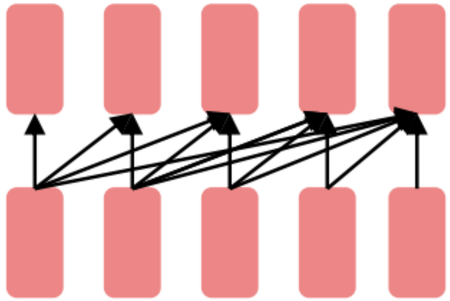
Encoder-Decoders

Pretraining for three types of architectures



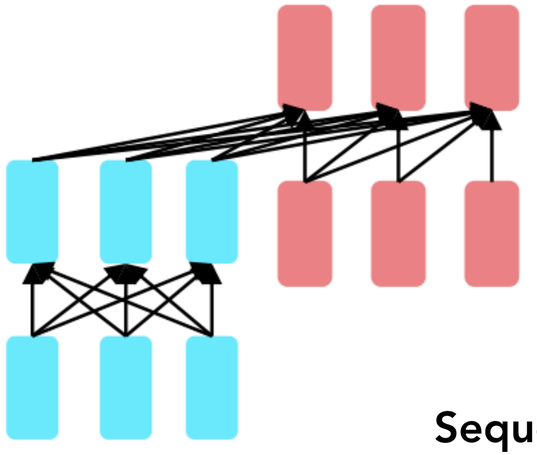
Encoders

Bidirectional Context



Decoders

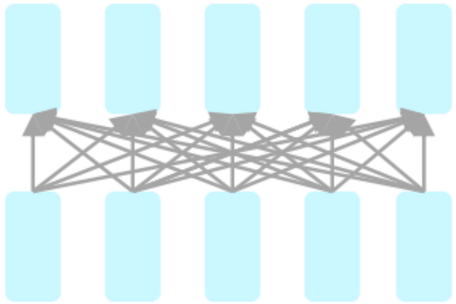
Language Models



Encoder-Decoders

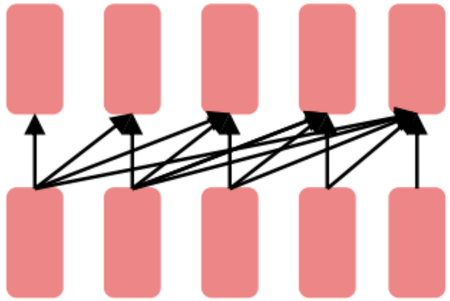
Sequence-to-sequence

Pretraining for three types of architectures



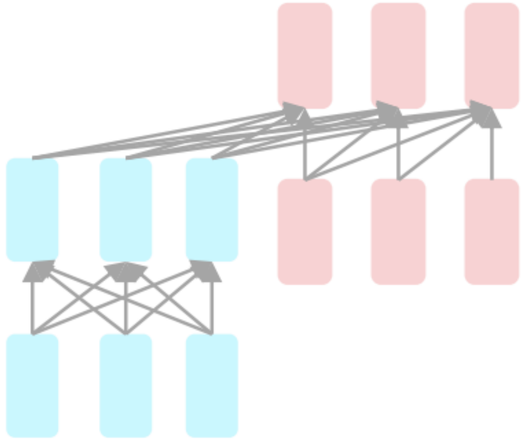
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

Sequence-to-sequence

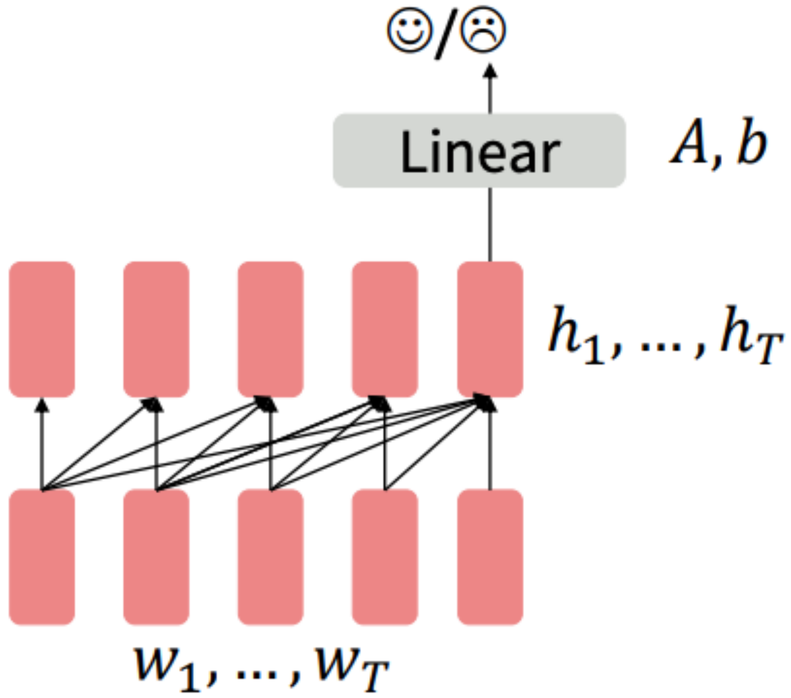
Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

Pre-training Decoder-Only Models

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

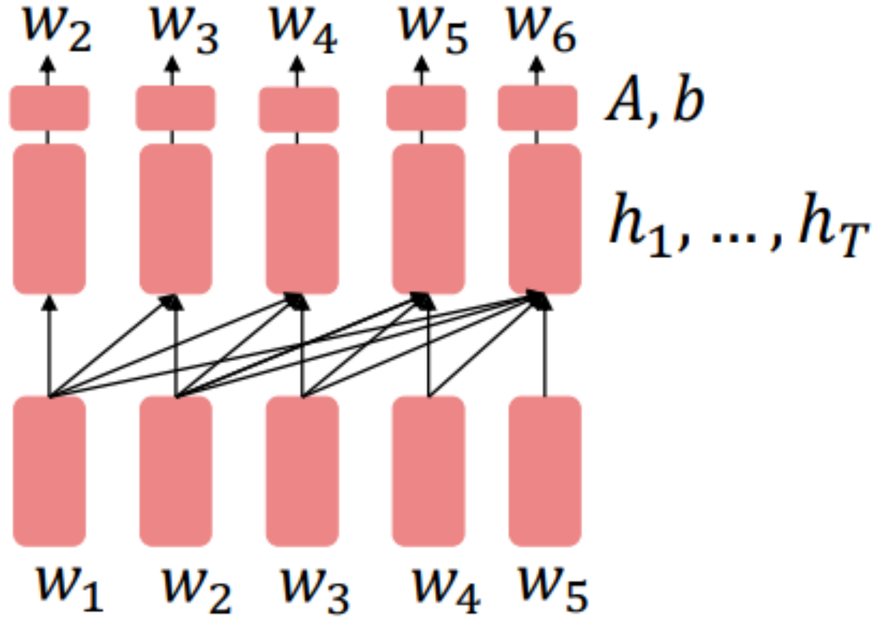
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned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

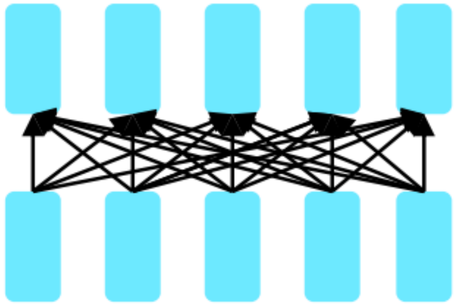
More in future weeks!

Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

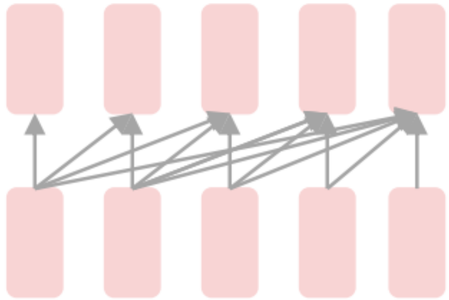
Pre-training Encoder-Only Models

Pretraining for three types of architectures



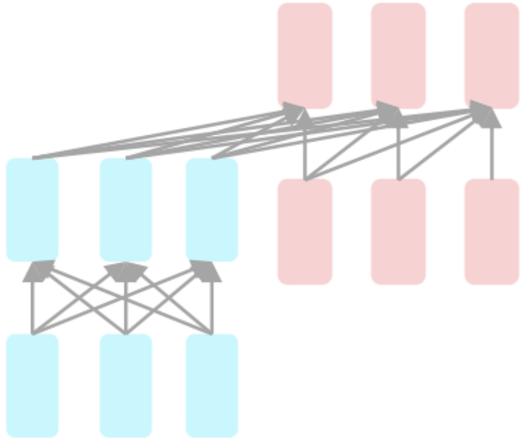
Encoders

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

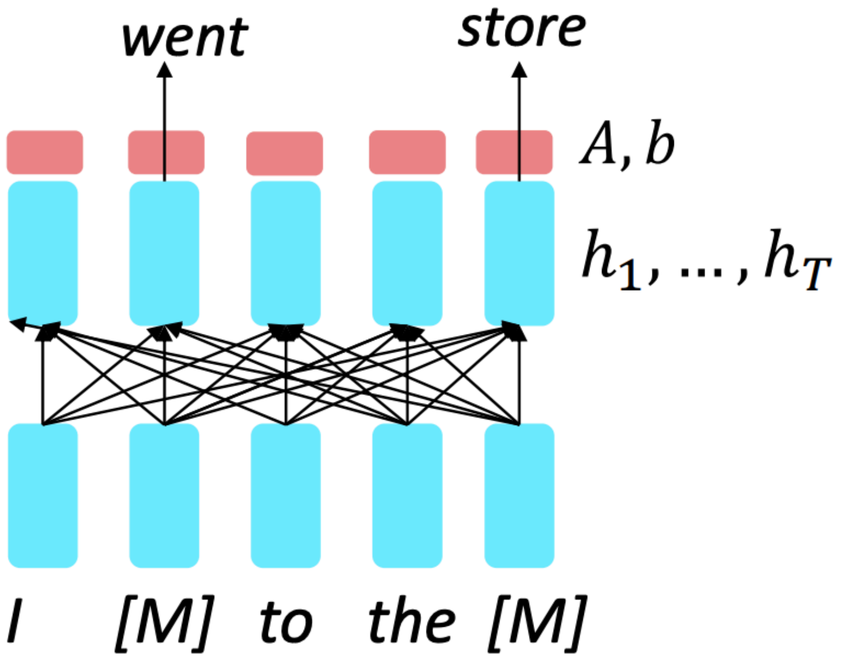
'Cause darling i'm a _____ dressed like a daydream

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

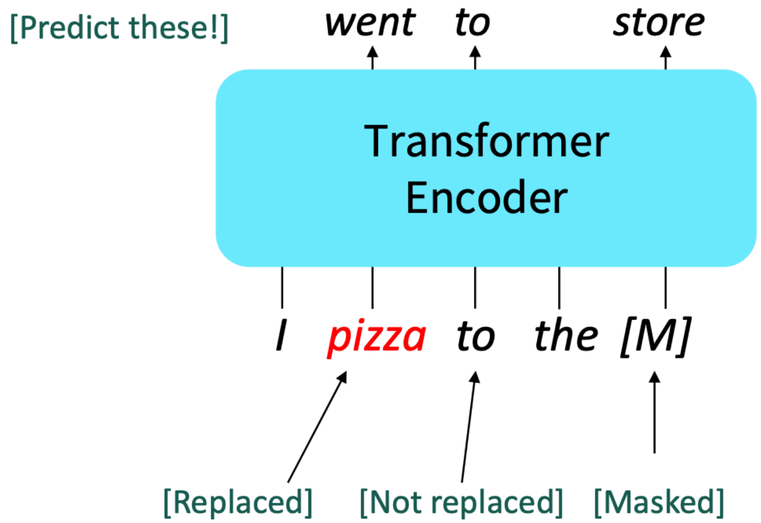


Masked Language Modeling

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

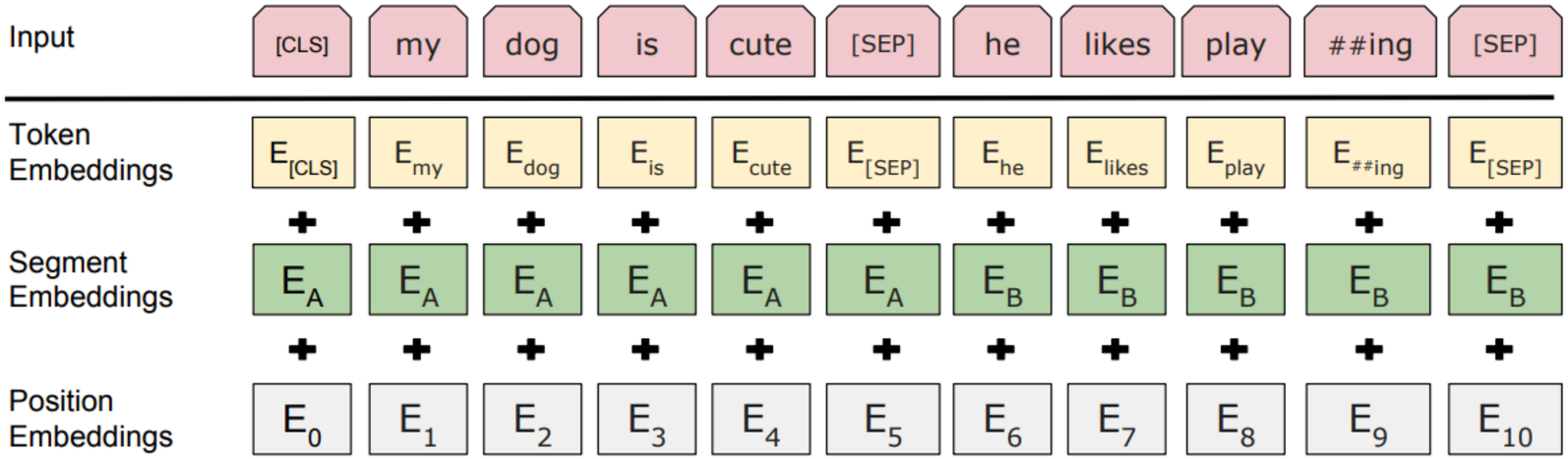


Why?

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

BERT: Bidirectional Encoder Representations from Transformers

- The pretraining input to BERT was two separate contiguous chunks of text:

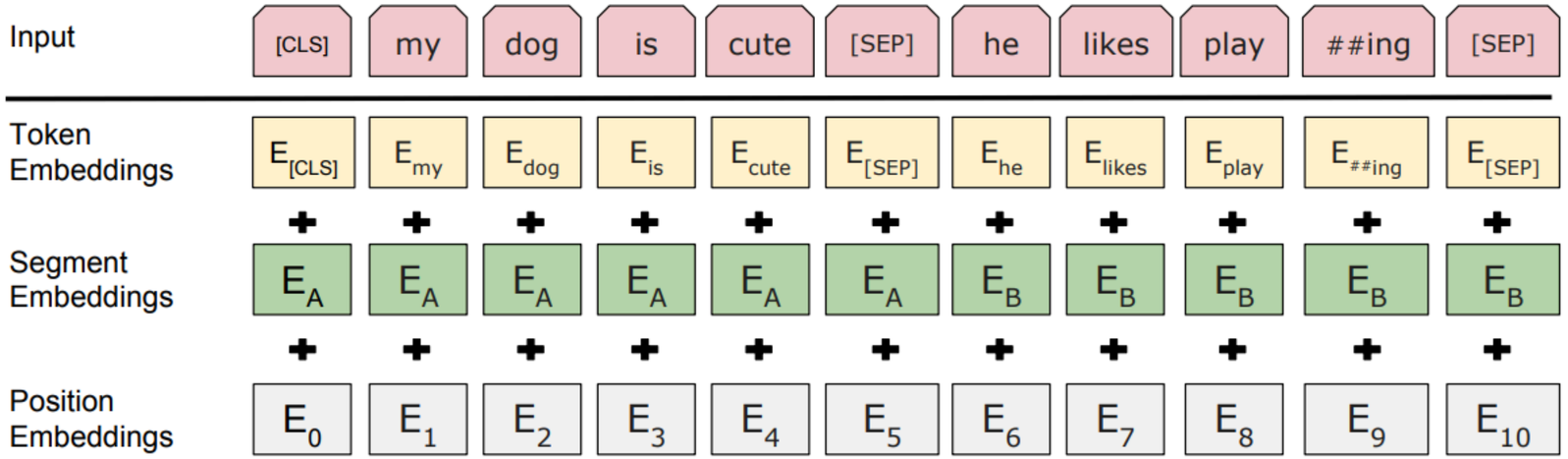


- BERT was trained to predict whether one chunk follows the other or is randomly sampled.
  - [CLS] and [SEP] tokens
  - [SEP] is used for next sentence prediction - do these sentences follow each other?
  - [CLS] for text classification / connection to fine-tuning

BERT: Training Details

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
    - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."

BERT: Contextual Embeddings



- BERT results in contextual embeddings
  - Embeddings for tokens in context, not just type embeddings like word2vec, GloVe
  - Can be used for measuring the semantic similarity of two words in context
  - Useful in linguistic tasks that require *precise* models of word meaning

BERT: Results

- BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Various Text Classification tasks like sentiment classification

BERT: Extensions

- Some generally accepted improvements to the BERT pretraining formula:
  - **RoBERTa**: mainly just train BERT for longer and remove next sentence prediction!
  - **SpanBERT**: masking contiguous spans of words makes a harder, more useful pretraining task
- A lot of BERT variants that used the BERT formula
  - ALBERT: BERT with parameter-reduction techniques
  - DistilBERT:
  - DeBERTa: Decoding-enhanced BERT with disentangled attention
  - FlauBERT: BERT for French
  - XLNet: Multilingual BERT
  - Etc.
- **BERTology**: How and why BERT worked so well

BERT: Overview

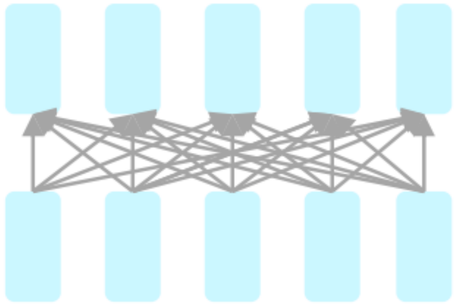
- [SEP]: Later work has argued this "next sentence prediction" is not necessary
- In general, more compute, more data can improve pretraining even when not changing the underlying Transformer encoder
- Results in contextual embeddings
- Key Limitation:
  - Cannot be used for generation
  - No pretraining encoders can be used for autoregressive generation very naturally
    - There are clunky ways in which you could try…but not a natural fit
    - For this, we need to have a decoder!

Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

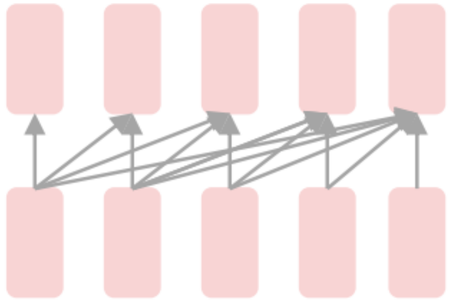
Pre-training Encoder-Decoder Models

Pretraining for three types of architectures



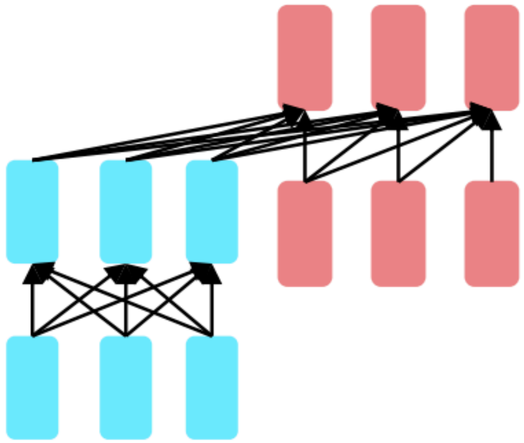
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

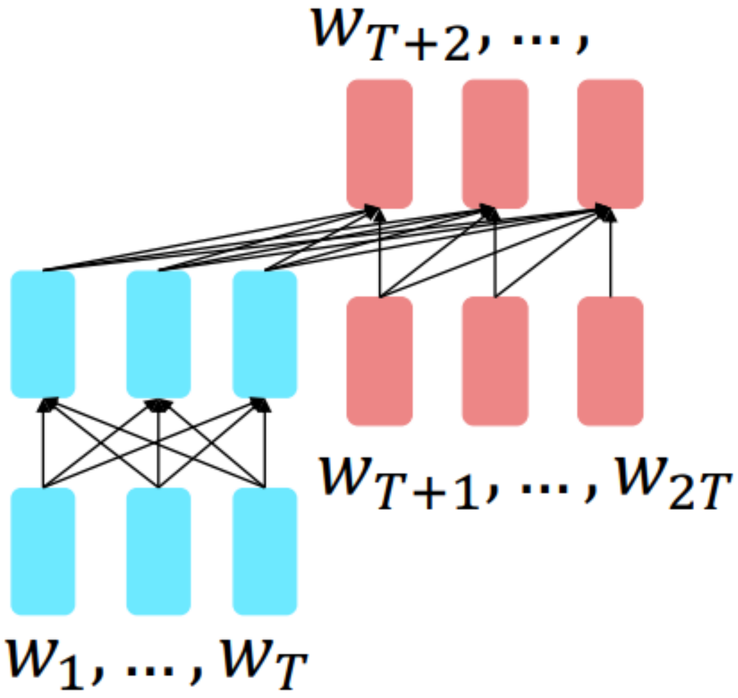
Sequence-to-sequence

Pretraining Encoder-Decoder Models

- For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$
$$h_{T+1}, \ldots, h_2 = Decoder(w_1, \ldots, w_T, h_1, \ldots, h_T)$$
$$y_i \sim Ah_i + b, i > T$$

The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling.



T5: A Pretrained Encoder-Decoder Model

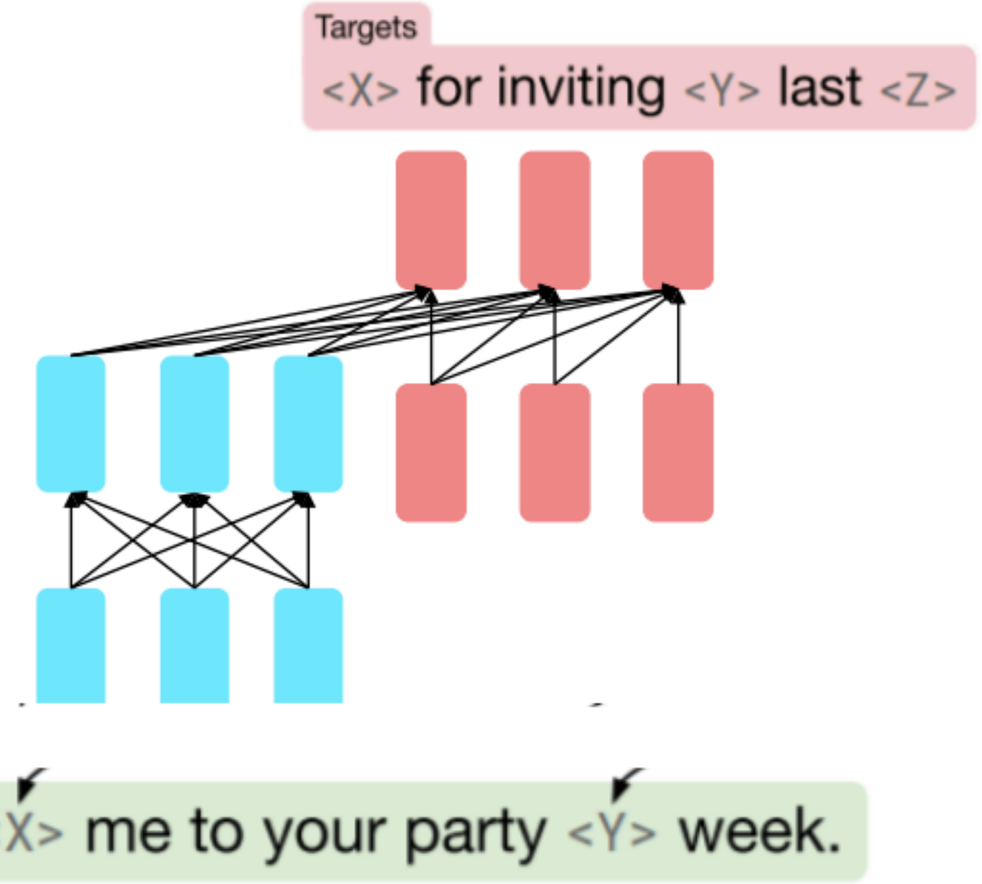
- Raffel et al., 2018 built T5, which uses as a span corruption pretraining objective

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Original text

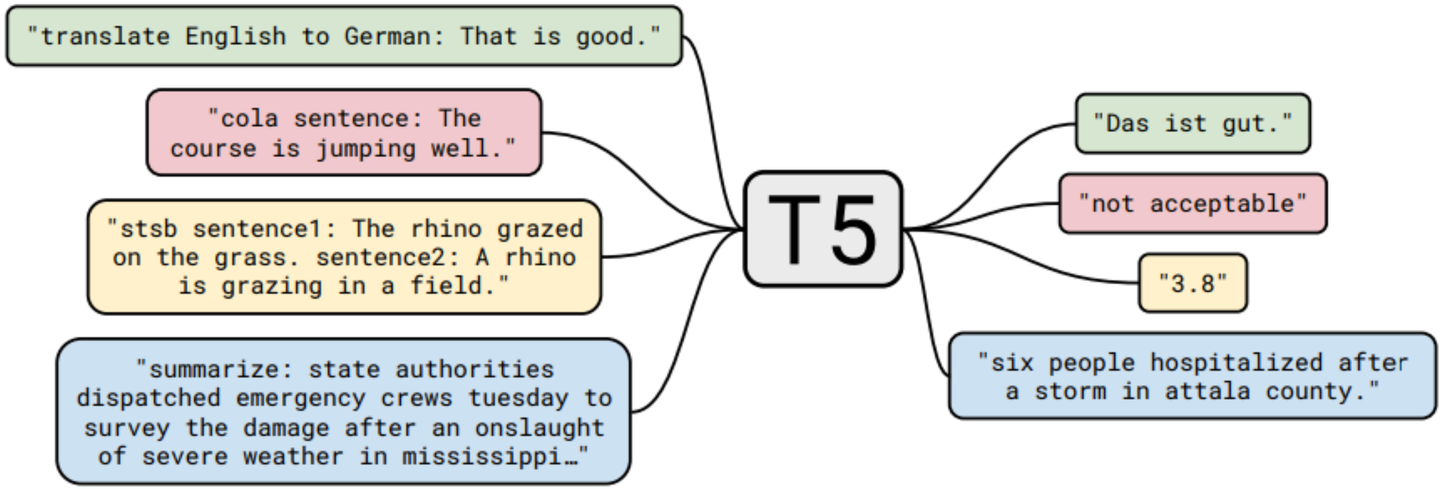
Thank you for inviting me to your party last week.

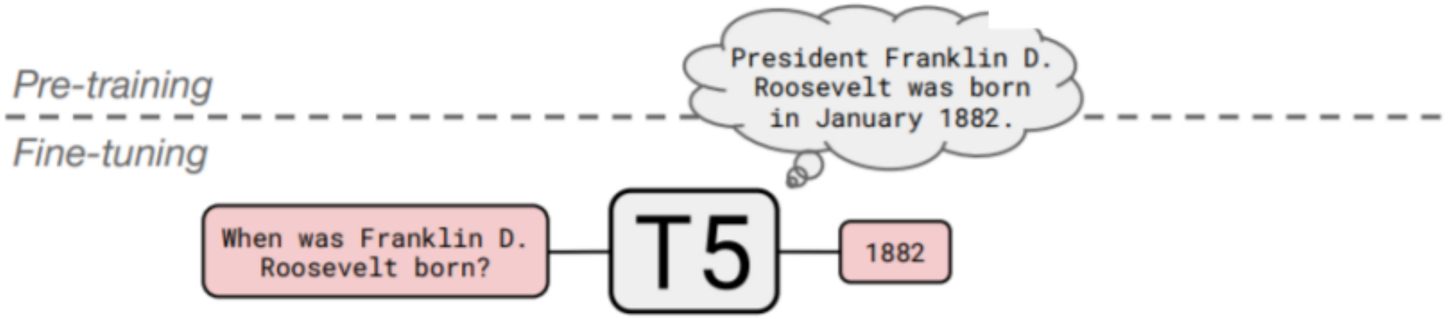
This is implemented in text preprocessing: it's still an objective that looks like language modeling at the decoder side.



T5: Task Preparation

A fascinating property of T5: it can be finetuned to answer a wide range of questions, retrieving knowledge from its parameters.





T5 Results

- Raffel et al., 2018 found encoder-decoders to work better than decoders for their tasks, and span corruption (denoising) to work better than language modeling.

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

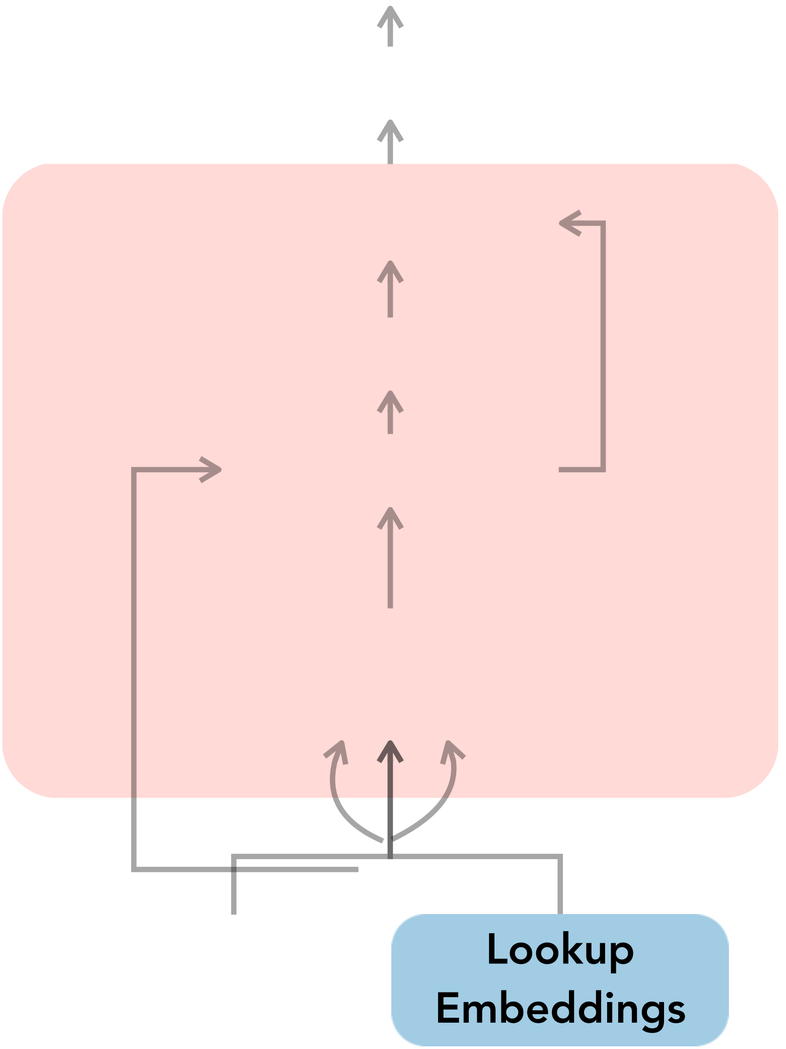
Lecture Outline

- Announcements
- Recap: Transformers as Encoders, Decoders, Encoder-Decoders
- The pre-training and fine-tuning paradigm
  - Pre-training Decoder-Only Models
  - Pre-training Encoder-Only Models
  - Pre-training Encoder-Decoder Models
- Tokenization

Tokenization in Transformers

The Input Layer

- So far, we have made some assumptions about a language's vocabulary
- Our approach so far: use a known, fixed vocabulary
  - Built from training data, with tens of thousands of components
  - However, even with the largest vocabulary, we may encounter out-of-vocabulary words at test time
  - Our approach so far: map novel words seen at test time (OOV) to a single UNK



How to get the words?

Or, more accurately, the tokens?

- Problem: break the text into a sequence of discrete tokens
- For alphabetic languages such as English, deterministic scripts usually suffice to achieve accurate tokenization
- However, in languages such as Chinese and Swahili, words are typically composed of a small number of characters, without intervening whitespace

Word Structure in Language

- Finite vocabulary assumptions make even less sense in many languages.
  - Many languages exhibit complex morphology, or word structure.
  - The effect is more word types, each occurring fewer times.

-ambia = to tell

Example: Swahili verbs can have hundreds of conjugations, each encoding a wide variety of information. (Tense, mood, definiteness, negation, information about the object, ++)

Conjugation of *-ambia*

Non-finite forms		
Form	Positive	Negative
Infinitive	kuambia	kutoambia

Simple finite forms		
Positive form	Singular	Plural
Imperative	ambia	ambieni
Habitual	huambia	

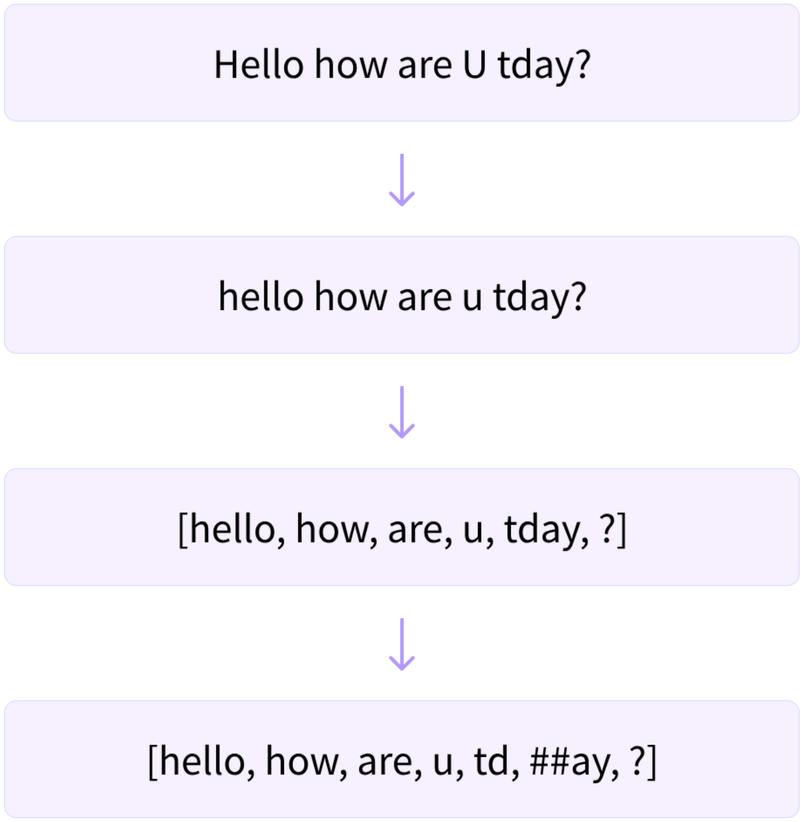
Complex finite forms

Polarity	Persons				Persons / Classes		Classes											
	1st		2nd		3rd / M-wa		M-mi		Ma		Ki-vi		N		U	Ku	Pa	Mu
	Sg.	Pl.	Sg.	Pl.	Sg. / 1	Pl. / 2	3	4	5	6	7	8	9	10	11 / 14	15 / 17	16	18
Past [less ▲]																		
Positive	niliambia naliambia	tuliambia twaliambia	uliambia waliambia	mliambia mwaliambia	aliambia	waliambia	uliambia	iliambia	liliambia	yaliambia	kiliambia	viliambia	iliambia	ziliambia	uliambia	kuliambia	paliambia	muliambia
Negative	sikuambia	hatukuambia	hukuambia	hamkuambia	hakuambia	hawakuambia	haukuambia	haikuambia	halikuambia	hayakuambia	hakikuambia	havikuambia	haikuambia	hazikuambia	haukuambia	hakukuambia	hapakuambia	hamukuambia
Present [less ▲]																		
Positive	ninaambia naambia	tunaambia	unaambia	mnaambia	anaambia	wanaambia	unaambia	inaambia	linaambia	yanaambia	kinaambia	vinaambia	inaambia	zinaambia	unaambia	kunaambia	panaambia	munaambia
Negative	siambii	hatuambii	huambii	hamambii	haambii	hawaambii	hauambii	haiambii	haliambii	hayaambii	hakiambii	haviambii	haiambii	haziambii	hauambii	hakuambii	hapaambii	hamuambii
Future [less ▲]																		
Positive	nitaambia	tutaambia	utaambia	mtaambia	ataambia	wataambia	utaambia	itaambia	litaambia	yataambia	kitaambia	vitaambia	itaambia	zitaambia	utaambia	kutaambia	pataambia	mutaambia
Negative	sitaambia	hatutaambia	hutaambia	hamtaambia	hataambia	hawataambia	hautaambia	haitaambia	halitaambia	hayataambia	hakitaambia	havitaambia	haitaambia	hazitaambia	hautaambia	hakutaambia	hapataambia	hamutaambia
Subjunctive [less ▲]																		
Positive	niambie	tuambie	uambie	mambie	aambie	waambie	uambie	iambie	liambie	yaambie	kiambie	viambie	iambie	ziambie	uambie	kuambie	paambie	muambie
Negative	nisiambie	tusiambie	usiambie	msiambie	asiambie	wasiambie	usiambie	isiambie	lisiambie	yasiambie	kisiambie	visiambie	isiambie	zisiambie	usiambie	kusiambie	pasiambie	musiambie
Present Conditional [less ▲]																		
Positive	ningeambia	tungeambia	ungeambia	mngeambia	angeambia	wangeambia	ungeambia	ingeambia	lingeambia	yangeambia	kingeambia	vingeambia	ingeambia	zingeambia	ungeambia	kungeambia	pangeambia	mungeambia
Negative	nisingeambia singeambia	tusingeambia hatungeambia	usingeambia hungeambia	msingeambia hamngeambia	asingeambia hangeambia	wasingeambia hawangeambia	usingeambia haungeambia	isingeambia haingeambia	lisingeambia halingeambia	yasingeambia hayangeambia	kisingeambia hakingeambia	visingeambia havingeambia	isingeambia haingeambia	zisingeambia hazingeambia	usingeambia haungeambia	kusingeambia hakungeambia	pasingeambia hapangeambia	musingeambia hamungeambia
Past Conditional [less ▲]																		
Positive	ningaliambia	tungaliambia	ungaliambia	mngaliambia	angaliambia	wangaliambia	ungaliambia	ingaliambia	lingaliambia	yangaliambia	kingaliambia	vingaliambia	ingaliambia	zingaliambia	ungaliambia	kungaliambia	pangaliambia	mungaliambia
Negative	nisingaliambia singaliambia	tusingaliambia hatungaliambia	usingaliambia hungaliambia	msingaliambia hamngaliambia	asingaliambia hangaliambia	wasingaliambia hawangaliambia	usingaliambia haungaliambia	isingaliambia haingaliambia	lisingaliambia halingaliambia	yasingaliambia hayangaliambia	kisingaliambia hakingaliambia	visingaliambia havingaliambia	isingaliambia haingaliambia	zisingaliambia hazingaliambia	usingaliambia haungaliambia	kusingaliambia hakungaliambia	pasingaliambia hapangaliambia	musingaliambia hamungaliambia
Conditional Contrary to Fact [less ▲]																		
Positive	ningeliambia	tungeliambia	ungeliambia	mngeliambia	angeliambia	wangeliambia	ungeliambia	ingeliambia	lingeliambia	yangeliambia	kingeliambia	vingeliambia	ingeliambia	zingeliambia	ungeliambia	kungeliambia	pangeliambia	mungeliambia
Gnomic [less ▲]																		
Positive	naambia	twaambia	waambia	mwaambia	aambia	waambia	waambia	yaambia	laambia	yaambia	chaambia	vyaambia	yaambia	zaambia	waambia	kwaambia	paambia	mwaambia
Perfect [less ▲]																		

Source: Wiktionary

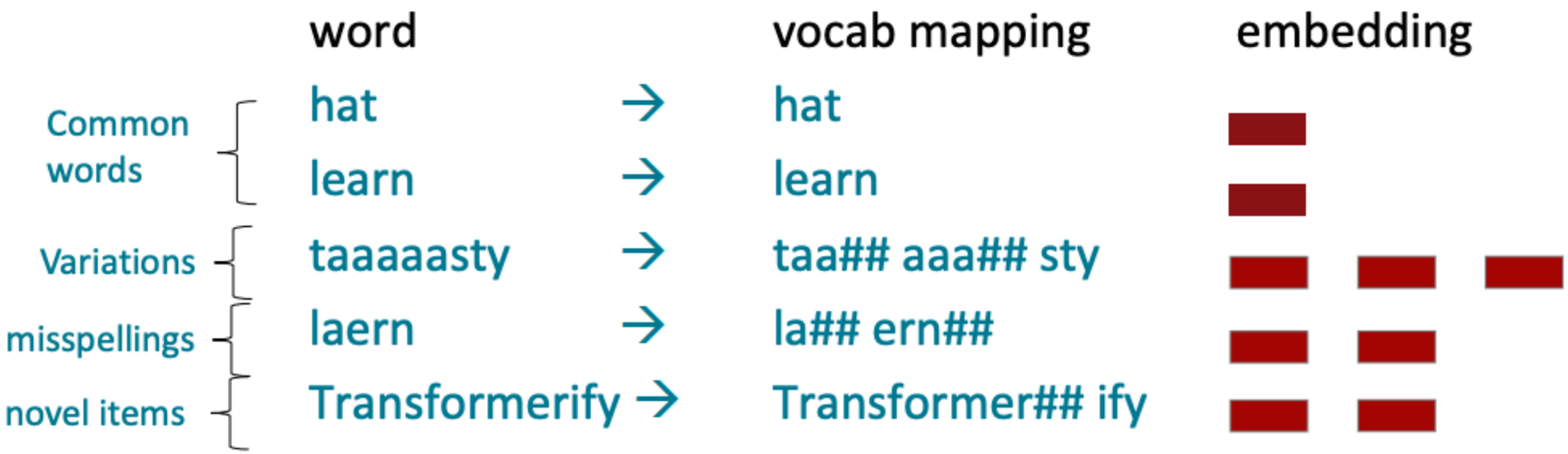
Subword Modeling

- Solution: look at subwords!
- Subword modeling encompasses a wide range of methods for reasoning about structure below the word level
  - Subwords may be words, parts of words, characters, bytes
- The dominant modern paradigm is to learn a vocabulary of parts of words (subword tokens)
- At training and testing time, each word is split into a sequence of known subwords
- Different algorithms:
  - Byte-Pair Encoding
  - WordPiece Modeling
  - Follow different strategies. Often contain prepending / appending special tokens (##, </w>)



Word structure and subword models

- Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components.
- In the worst case, words are split into as many subwords as they have characters.



Byte-pair encoding

- Byte-pair encoding is a simple, effective strategy for defining a subword vocabulary
- Adapted for word segmentation from data compression technique (Gage, 1994)
  - Instead of merging frequent pairs of bytes, we merge characters or character sequences
- Algorithm:
  1. Start with a vocabulary containing only characters and an "end-of-word" symbol.
  2. Using a corpus of text, find *the most common adjacent characters* "a,b"; add "ab" as a subword
     - This is a learned operation! However, not a parametric function
     - Only combine pairs (hence the name!)
  3. Replace instances of the character pair with the new subword; repeat until desired vocabulary size.
- At test time, first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols
- Originally used in NLP for machine translation; now a similar method (WordPiece) is used in pretrained models.

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	l o w e r </w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>
l o w </w>	w i d e s t</w>	n e w e s t </w>
l o w </w>	w i d e s t </w>	n e w e s t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es								

Frequency

d-e (3)	l-o (7)	t-</w> (8)
e-r (2)	n-e (5)	w-</w> (5)
e-s (8)	o-w (7)	w-e (7)
e-w (5)	r-</w> (2)	w-i (3)
i-d (3)	s-t (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	l o w e r </w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>
l o w </w>	w i d es t</w>	n e w es t </w>
l o w </w>	w i d es t </w>	n e w es t </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

Frequency

d-es (3)	l-o (7)	w-</w> (5)
e-r (2)	n-e (5)	w-es (5)
e-w (5)	o-w (7)	w-e (2)
es-t (8)	r-</w> (2)	w-i (3)
i-d (3)	t-</w> (8)	

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

BPE in action

Corpus

low	lower	newest
low	lower	newest
low	widest	newest
low	widest	newest
low	widest	newest

Corpus

low</w>	lower</w>	newest</w>
low</w>	lower</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>
low</w>	widest</w>	newest</w>

Corpus

l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	l o w e r </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>
l o w </w>	w i d est </w>	n e w est </w>

Vocabulary

d	e	i	l	n	o	s	t	w
es	est	est</w>	lo	low	low</w>	ne	new	newest</w>

After 10 merges

Source: https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/

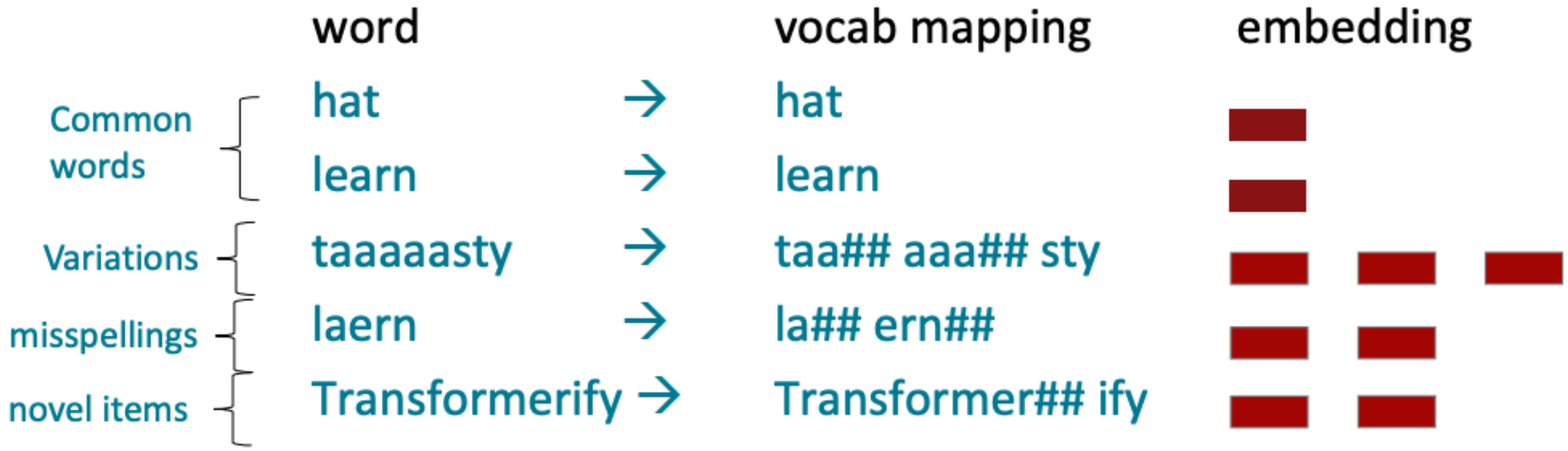
WordPiece Modeling

- Algorithm from Google, similar to BPE
- Identifies subwords by adding a prefix (##)
  - Each word is initially split by adding ## to all the characters inside a word
  - So, for instance, “word” gets split like this: w ##o ##r ##d
  - For this vocabulary:
    - ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
  - Splits may look like:
    - ("h" "##u" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("h" "##u" "##g" "##s", 5)
- On merging, ## ***between*** the two tokens is removed
  - This explains the presence of the token “##ing”

WordPiece Modeling Outcome

- Different stopping criteria: number of merges or size of resulting vocabulary
- In the worst case, at test time, words are split into as many subwords as they have characters
- Common words end up being a part of the subword vocabulary, while rarer words are split into (sometimes intuitive, sometimes not) components

WordPiece Outcome



Tokenization: Questions

- Where does the token "##ing" come from?
  - In WordPiece tokenization, all non-starting characters are initialized as ##x.
    - Like: h, ##e, ##l, ##l, ##o.
  - Upon merging, only the first segment keeps its ##.
- How is tokenization done in Chinese?
  - Follows the same broad overall algorithm, but the initial split into characters involve language-specific rules
  - e.g. stroke-level tokenization [Si et al., 2023]

Source: https://huggingface.co/learn/nlp-course/chapter6/6?fw=pt