Lecture 14: Finetuning Transformers

Instructor: Swabha Swayamdipta

USC CSCI 499 LMs in NLP

Mar 20, Spring 2024

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

Logistics / Announcements

- HW3 due tonight
- HW4 out today, due Fri, 4/3
- Next class: Guest Lecture by Prof. Jieyu Zhao
  - Harms of LLMs
- Next week:
  - Instructor OH cancelled
  - Email me if you need anything and I'll set something up

Large Language Models (LLMs)

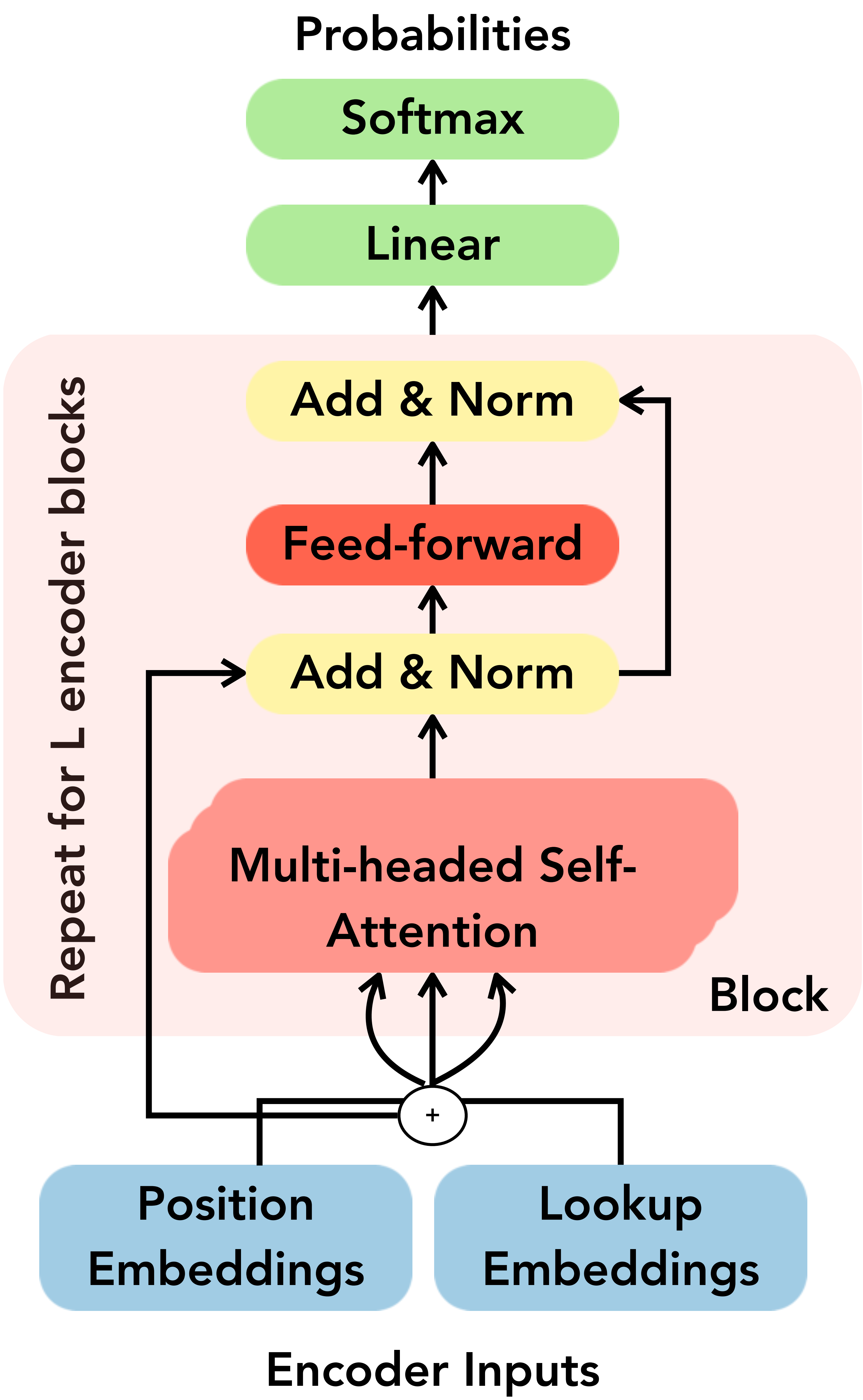
Mar 18:	Pre-training Transformers I SLIDES	
Mar 20:	**Pre-training Transformers II**	HW3 Due; HW4 Released
Mar 25:	Guest Lecture: Limitations and Harms of LLMs	
Mar 27:	Generating from Language Models I	
Apr 1:	Generating from Language Models II	
Apr 3:	**Prompting LLMs**	HW4 Due
Apr 8:	PROJECT DISCUSSIONS	
Apr 10:	Aligning LLMs	

Outro and Project Presentations

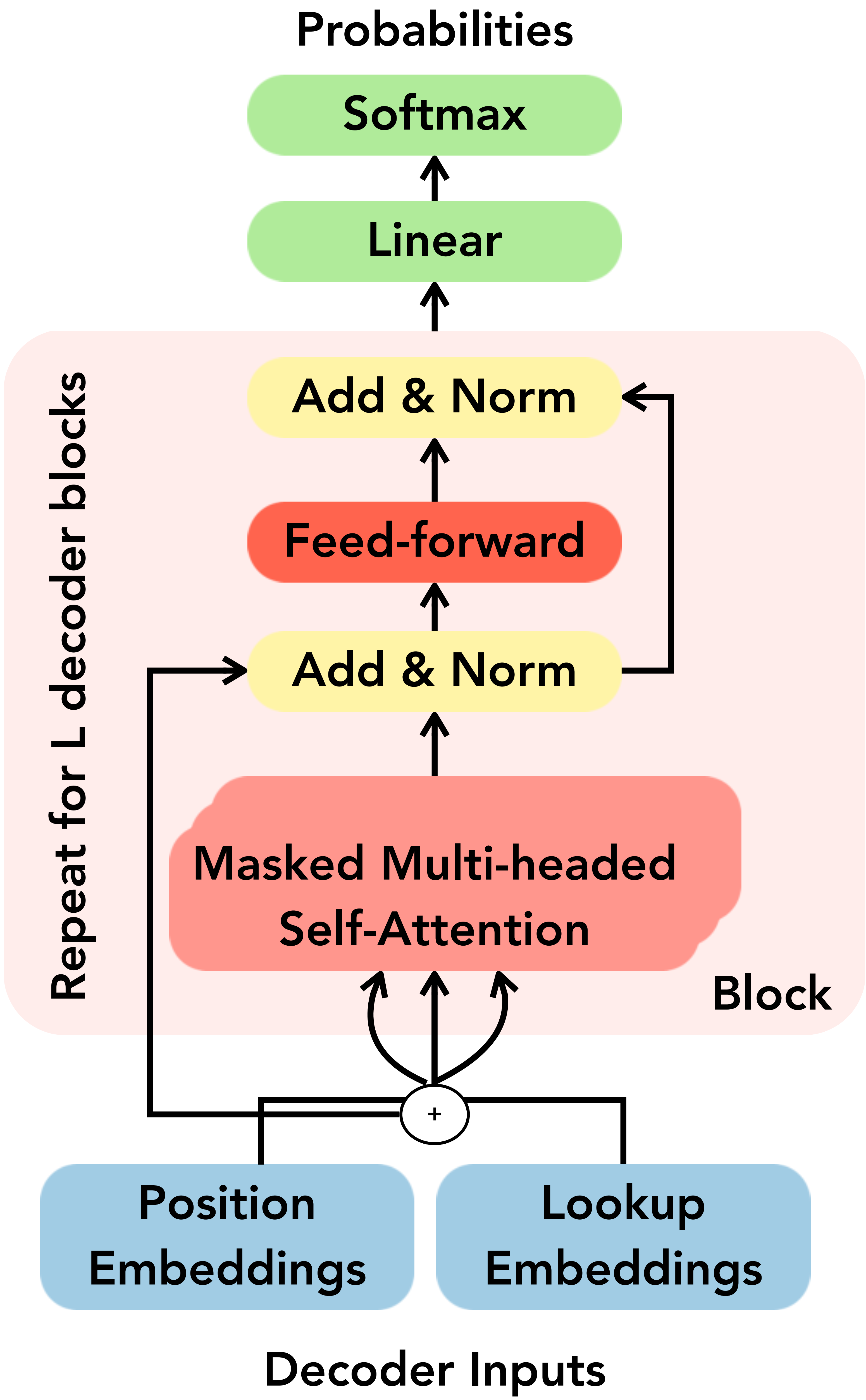
Apr 15:	**Putting it all together**	No Additional Readings
Apr 17:	PROJECT PRESENTATIONS	
Apr 22:	PROJECT PRESENTATIONS	
Apr 24:	PROJECT PRESENTATIONS	

Recap:
Transformer Encoder-Decoders

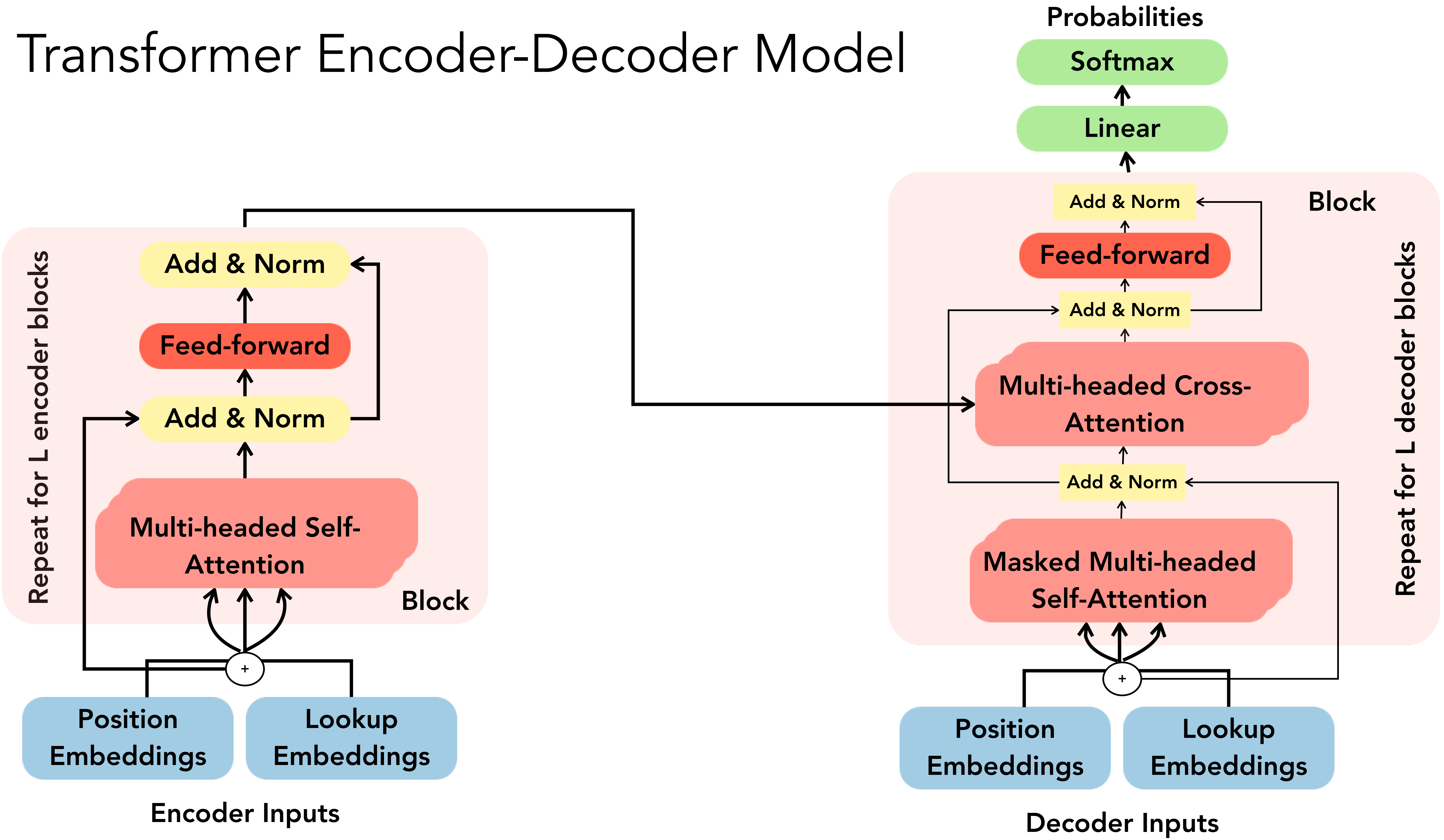
The Transformer Encoder-Only Model



The Transformer Decoder-Only Model



The Transformer Encoder-Decoder Model



Teacher Forcing

- Strategy in **training** decoders
  - Decoder-only models
  - Encoder-decoder models
- Force the decoder to consider ground truth, regardless of the probability assigned to a predicted token
- At each time step t in decoding we force the system to use the gold target token from training as the next input x_{t+1}, rather than allowing it to rely on the (possibly erroneous) decoder output $\hat{y}_t$


Residual Connections
Positional Embeddings
$L_0(\theta)$ $L_1(\theta)$ $L_2(\theta)$ $L_3(\theta)$ $L_4(\theta)$
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$
W
Layer Norm
FF
Multi-headed Self-Attention
Multi-headed Cross-Attention
Masked Multi-headed Self-Attention
× L Layers
Source Sentence x
The
green
witch
arrived
<s>
llegó
la
bruja
verde
Target y


Residual Connections
Positional Embeddings
$\hat{y}_0$ arg max
$\hat{y}_1$ arg max
$\hat{y}_2$ arg max
W
W
W
Layer Norm
FF
Layer Norm
Multi-headed Cross-Attention
Layer Norm
Masked Multi-headed Self-Attention
Layer Norm
FF
Layer Norm
Multi-headed Self-Attention
$\times L$ Layers
$\times L$ Layers
Source Sentence x
The
green
witch
arrived
<s>
la
verde
bruja
Predictions y

Recap: The Pre-training and Fine-tuning Paradigm

“Pretrain once, finetune many times.”

– Unknown, yet common practice in natural language processing

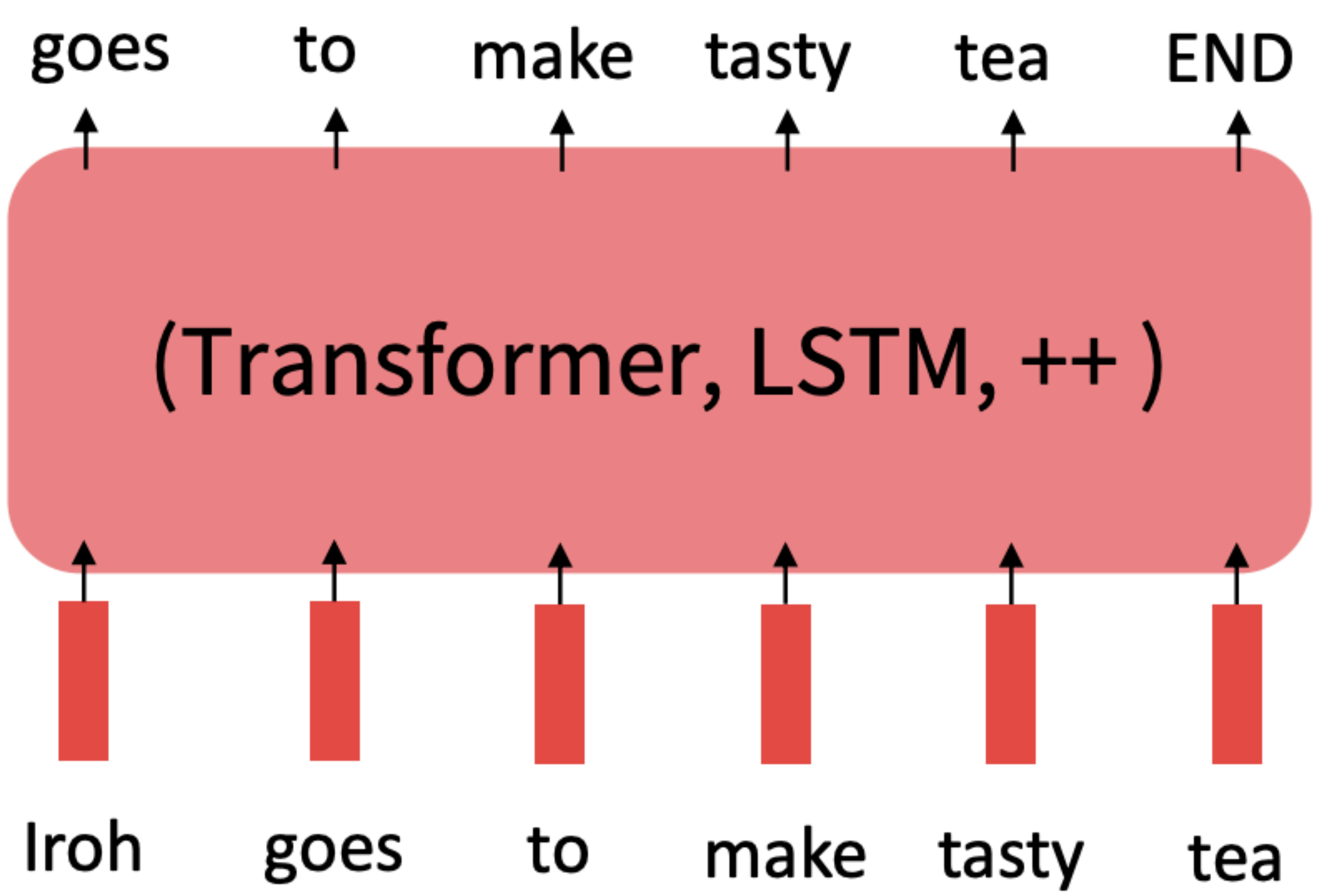
USC Viterbi

The Pretraining / Finetuning Paradigm

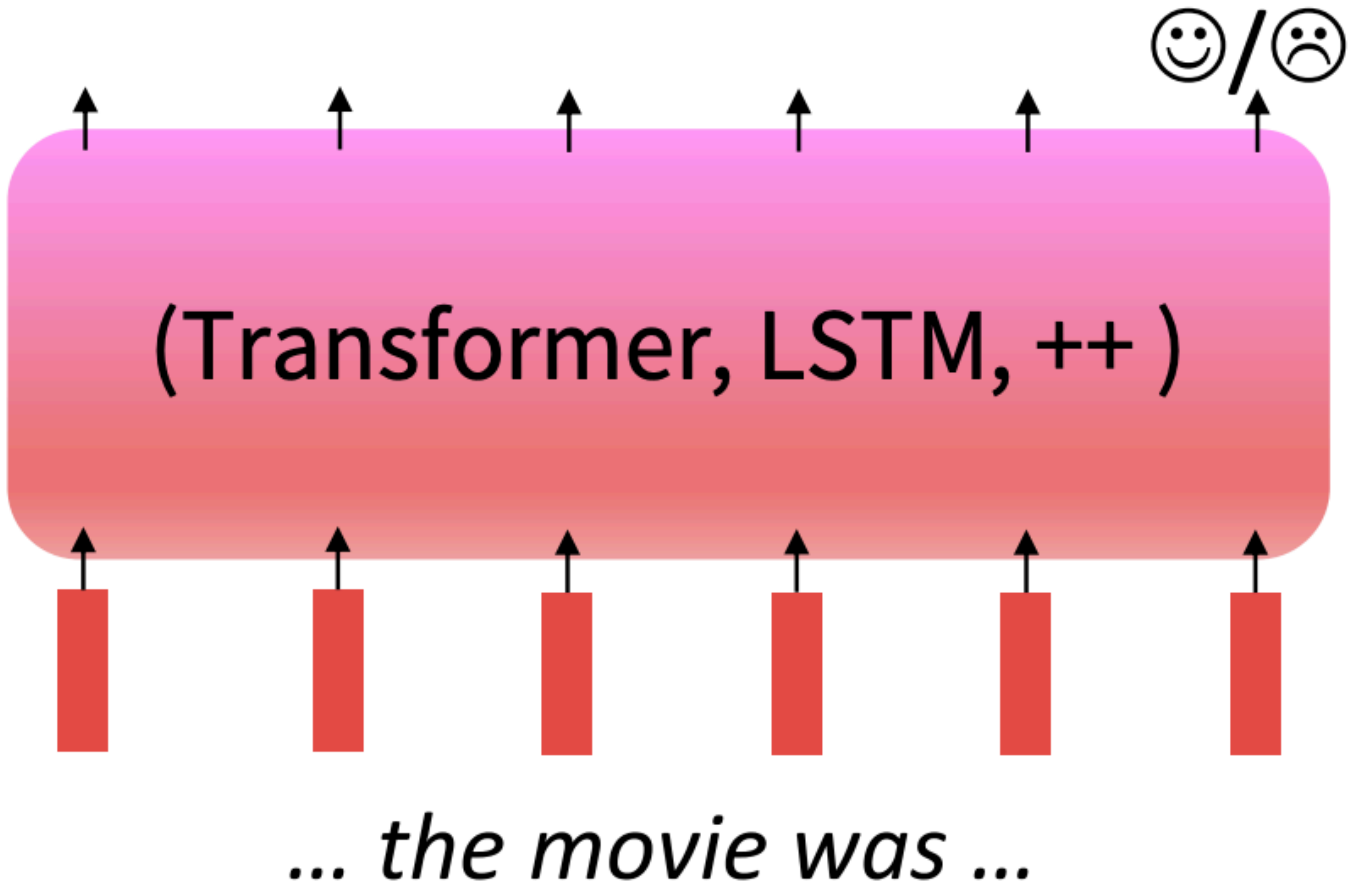
- Pretraining can improve NLP applications by serving as parameter initialization.

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!

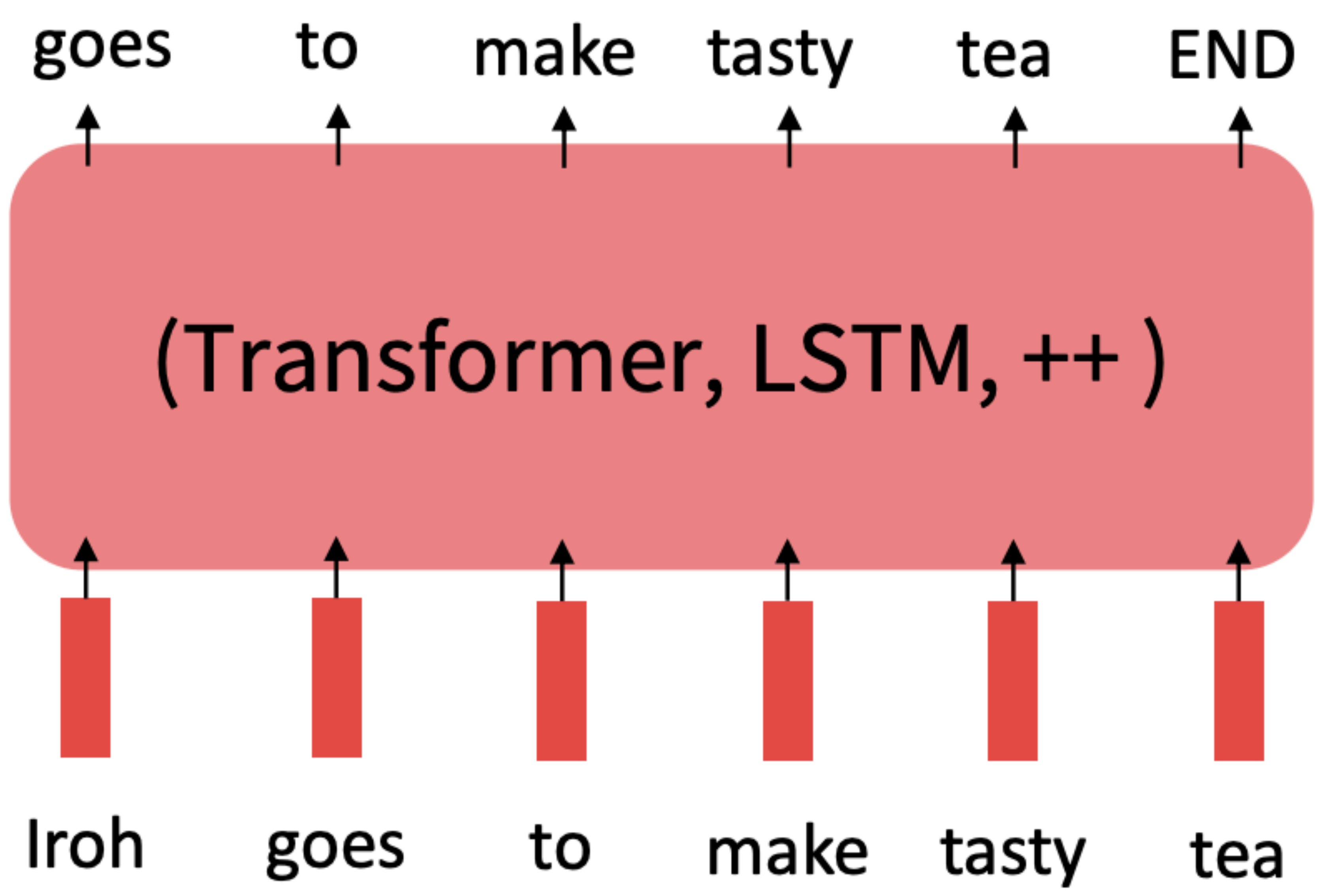


Pretraining

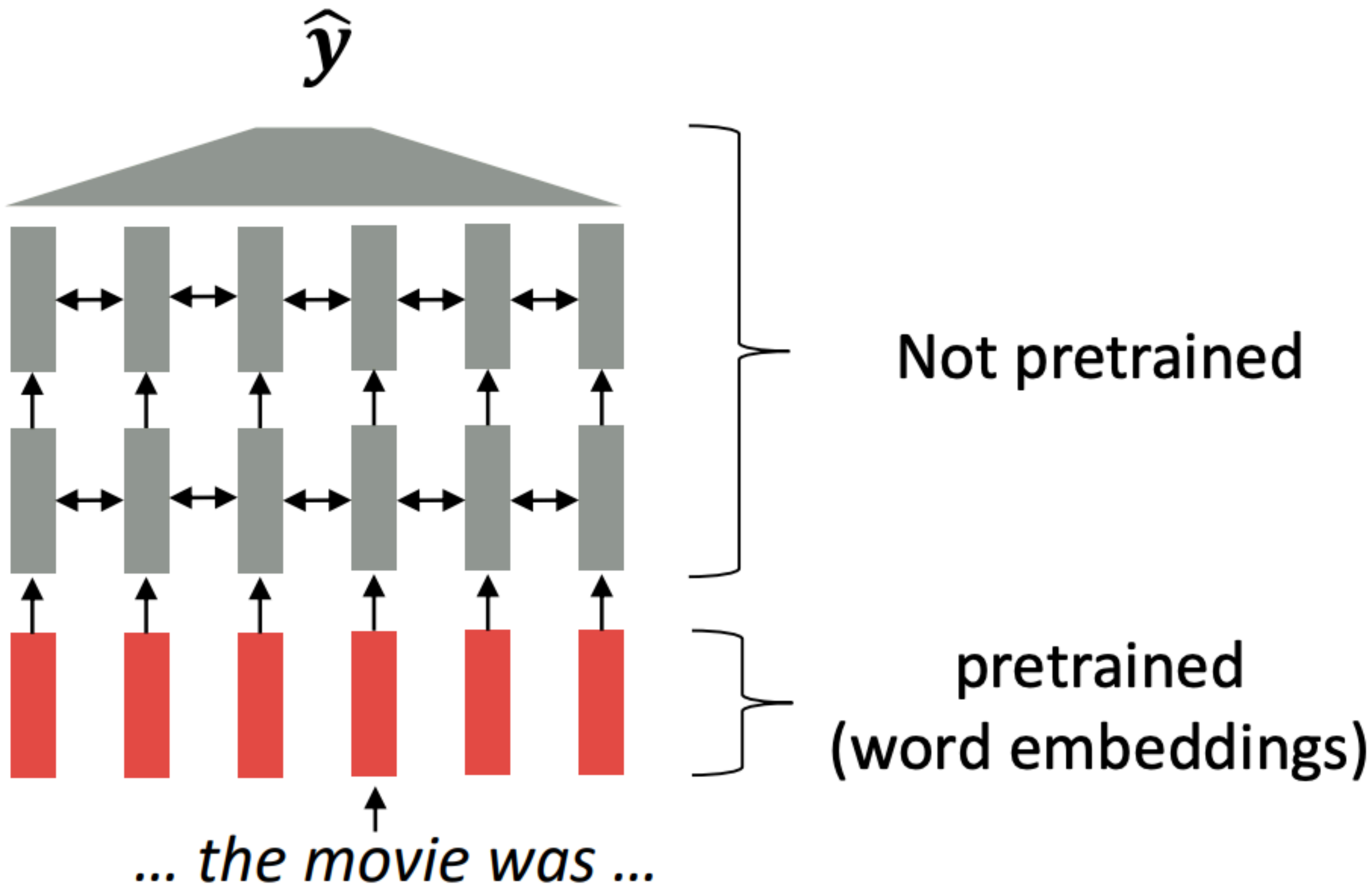
- Central Approach: Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts.
- Abstracts away from the task of "learning the language"

Step 1: Pretrain (on language corpora)

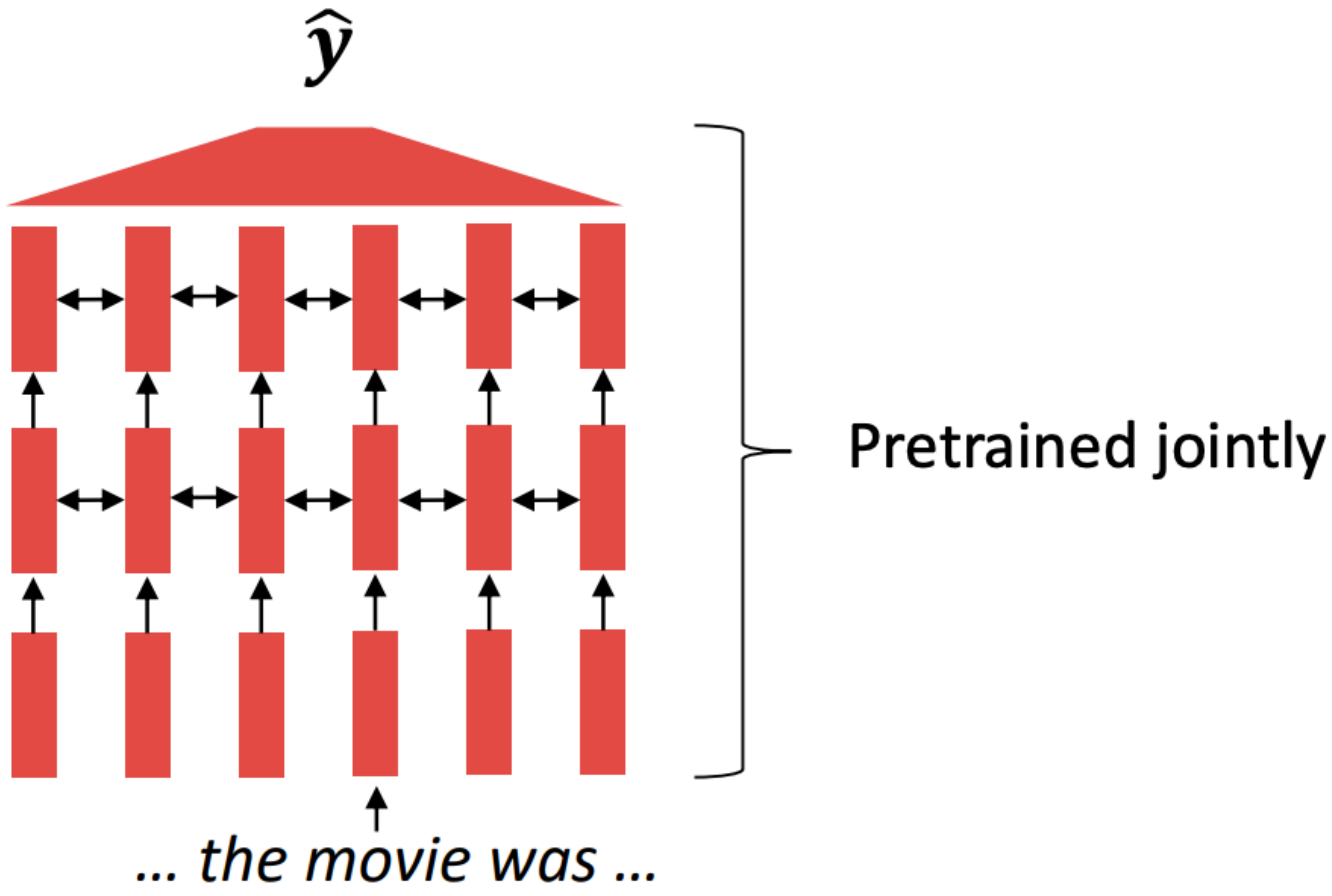
Lots of text; learn general things!



Pretraining: Parameter Initialization



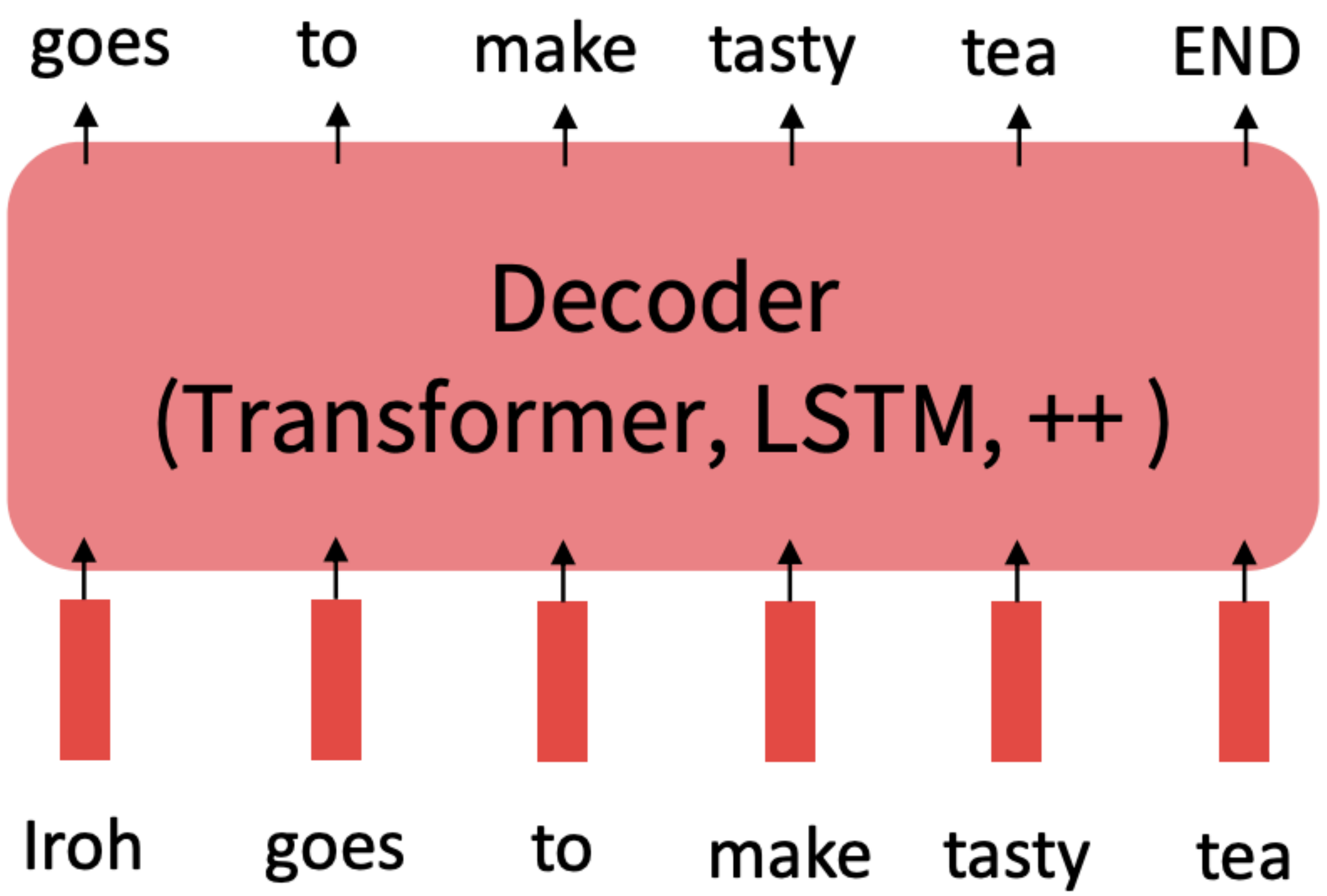
Partial network



Full network

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.
- Pretraining is 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

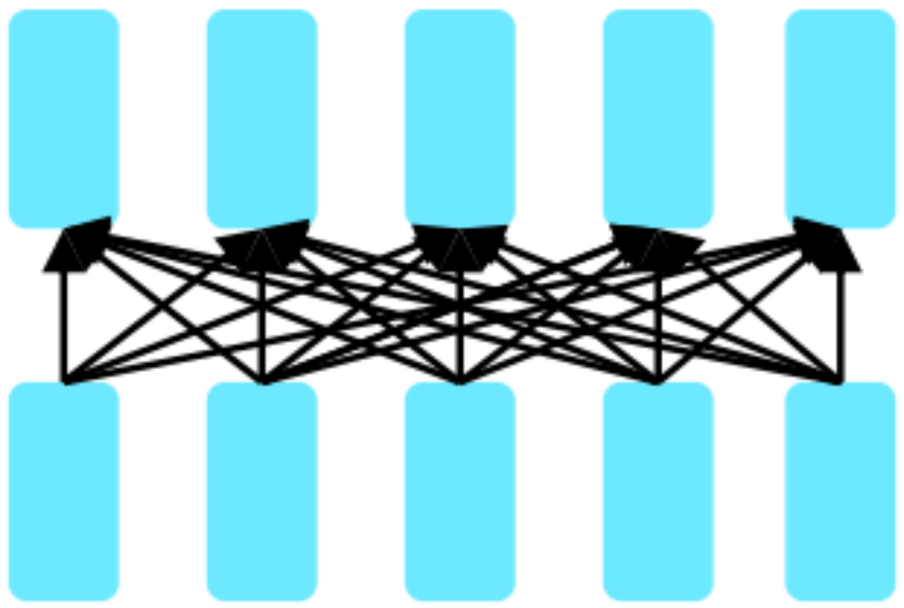


Semi-supervised Sequence Learning

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

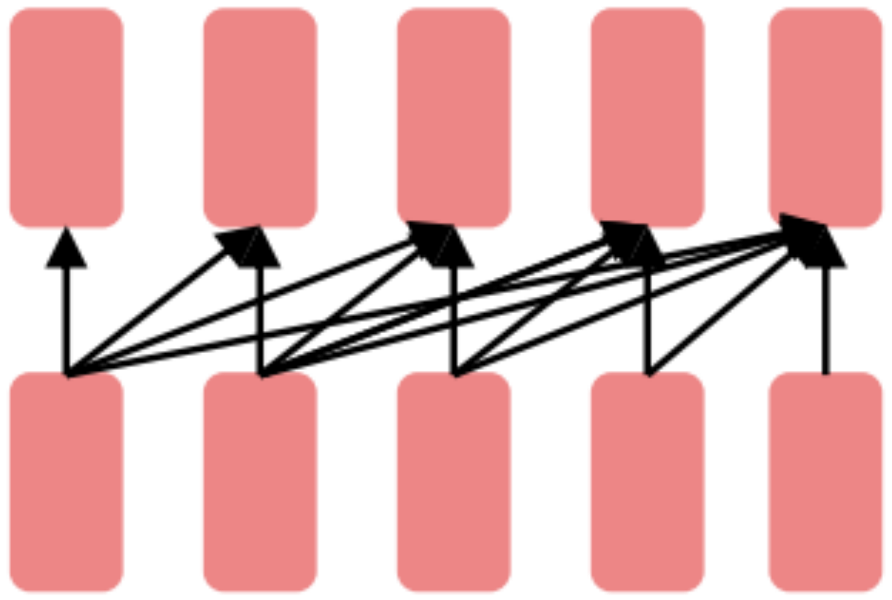
Quoc V. Le
Google Inc.
qvl@google.com

Pretraining for three types of architectures



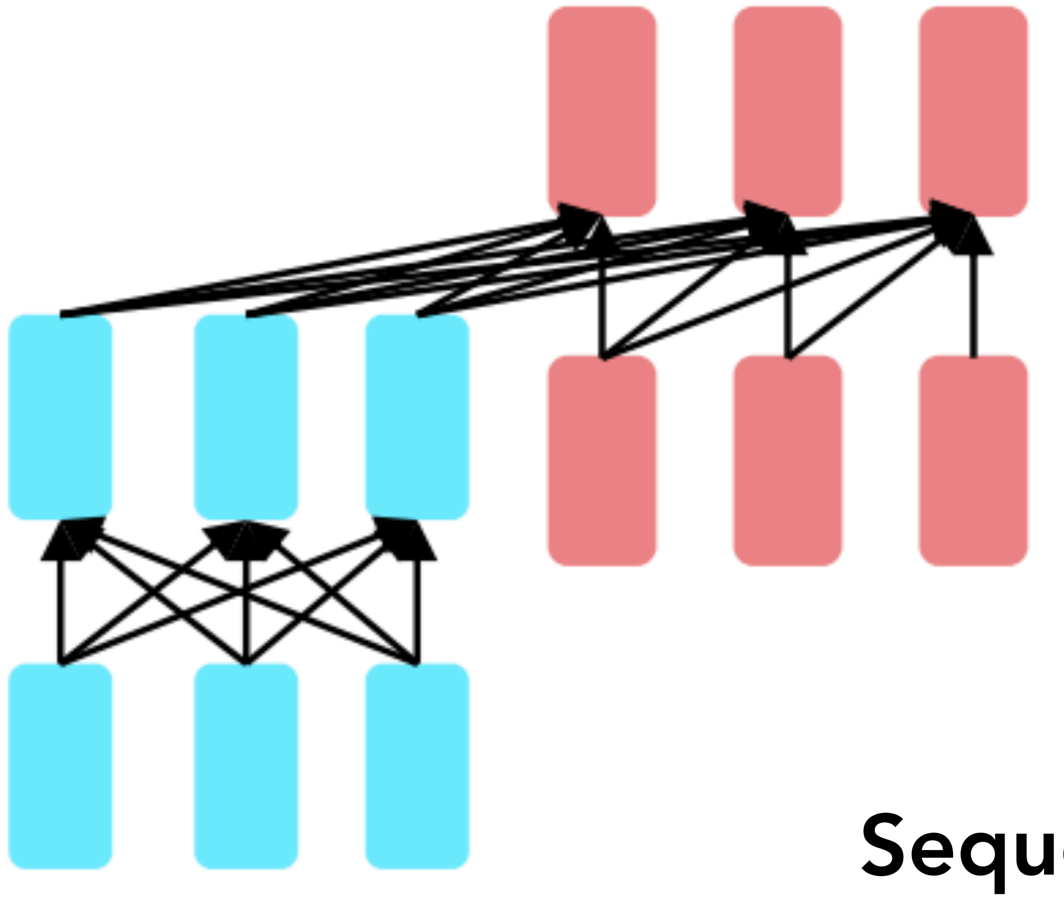
Encoders

Bidirectional Context



Decoders

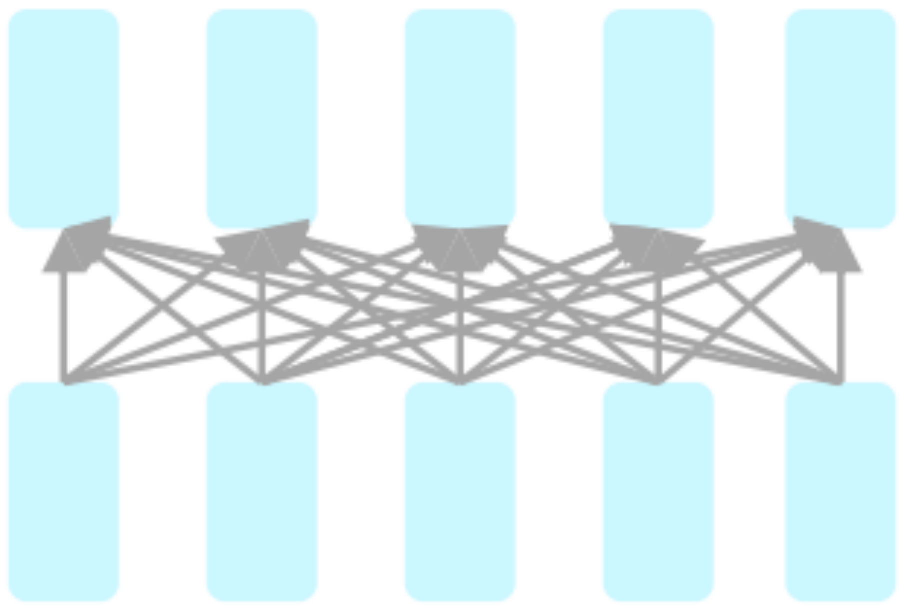
Language Models



Encoder-Decoders

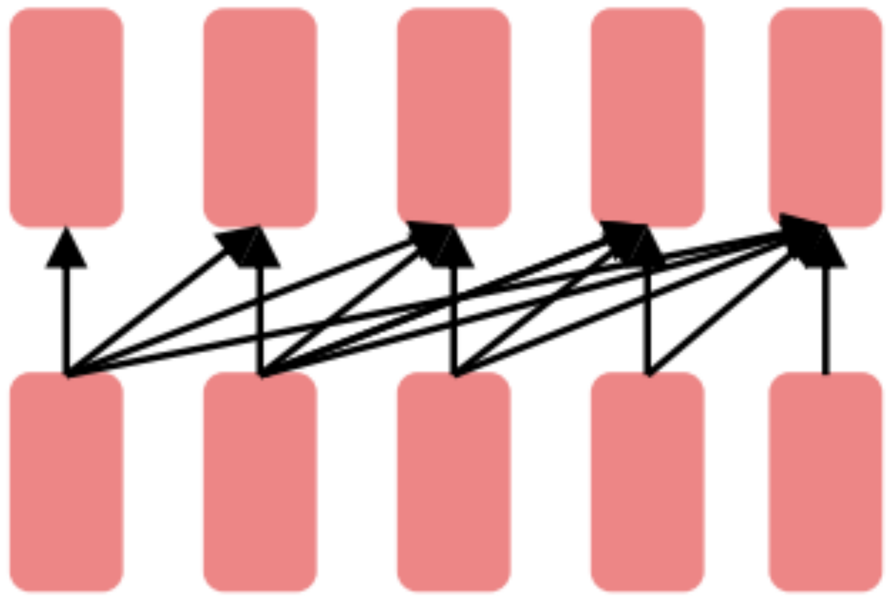
Sequence-to-sequence

Pretraining for three types of architectures



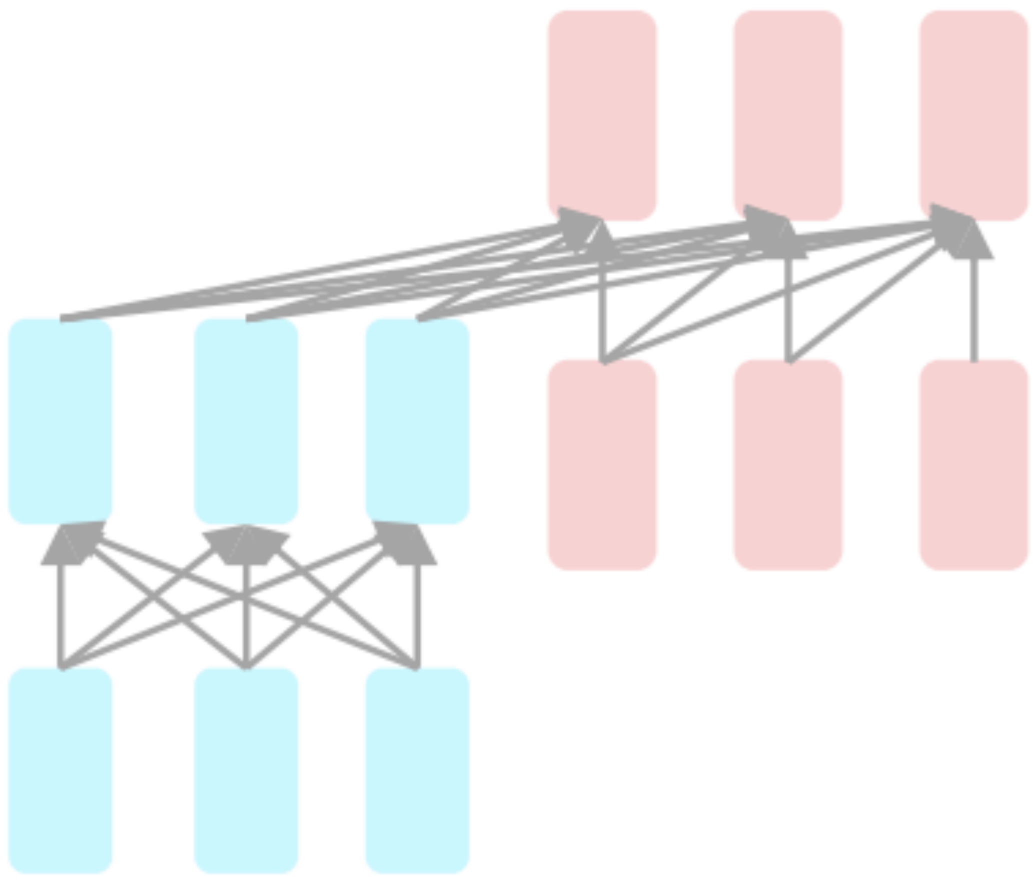
Encoders

Bidirectional Context



Decoders

Language Models

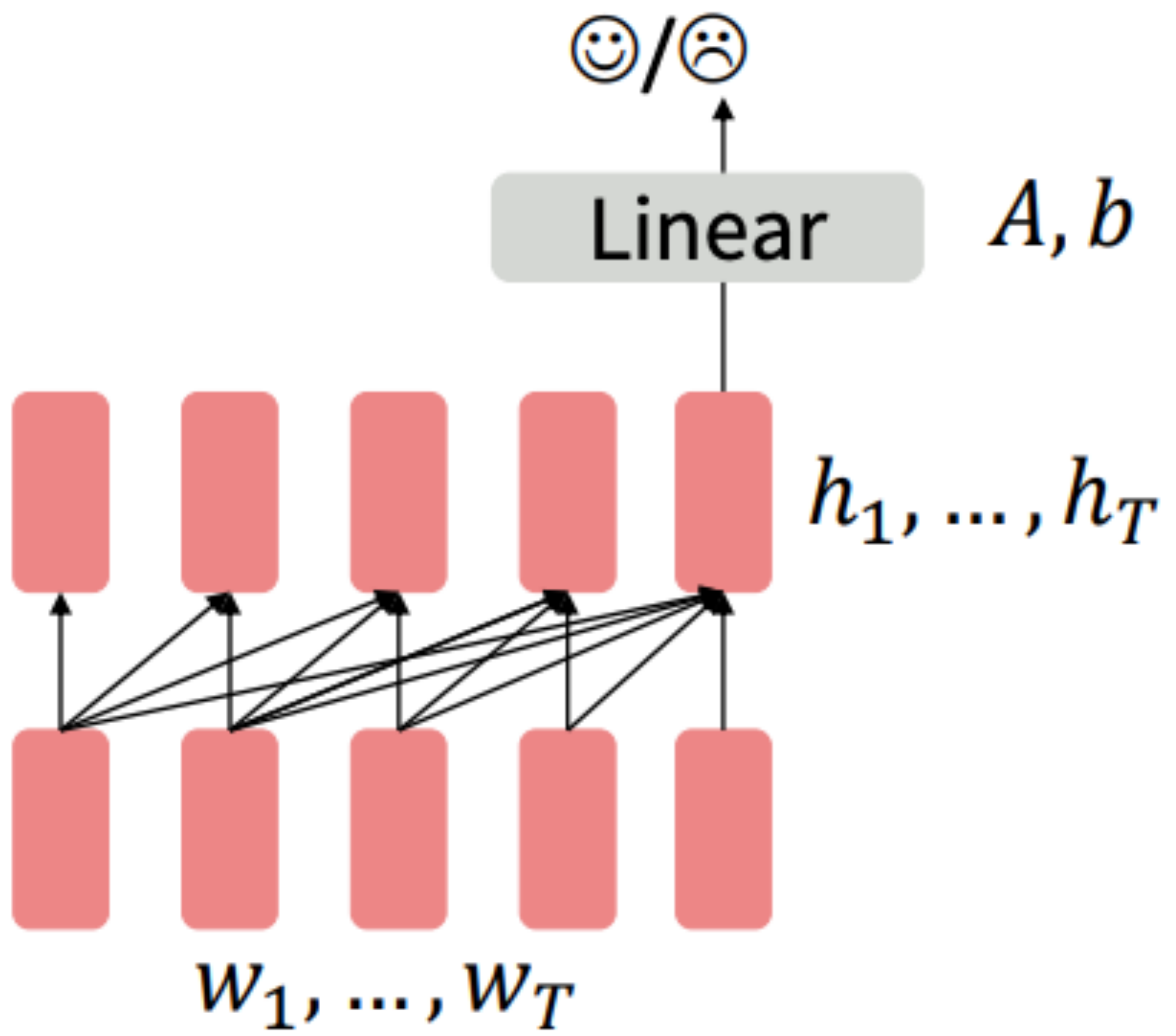


Encoder-Decoders

Sequence-to-sequence

Pretraining Decoders: Classifiers

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

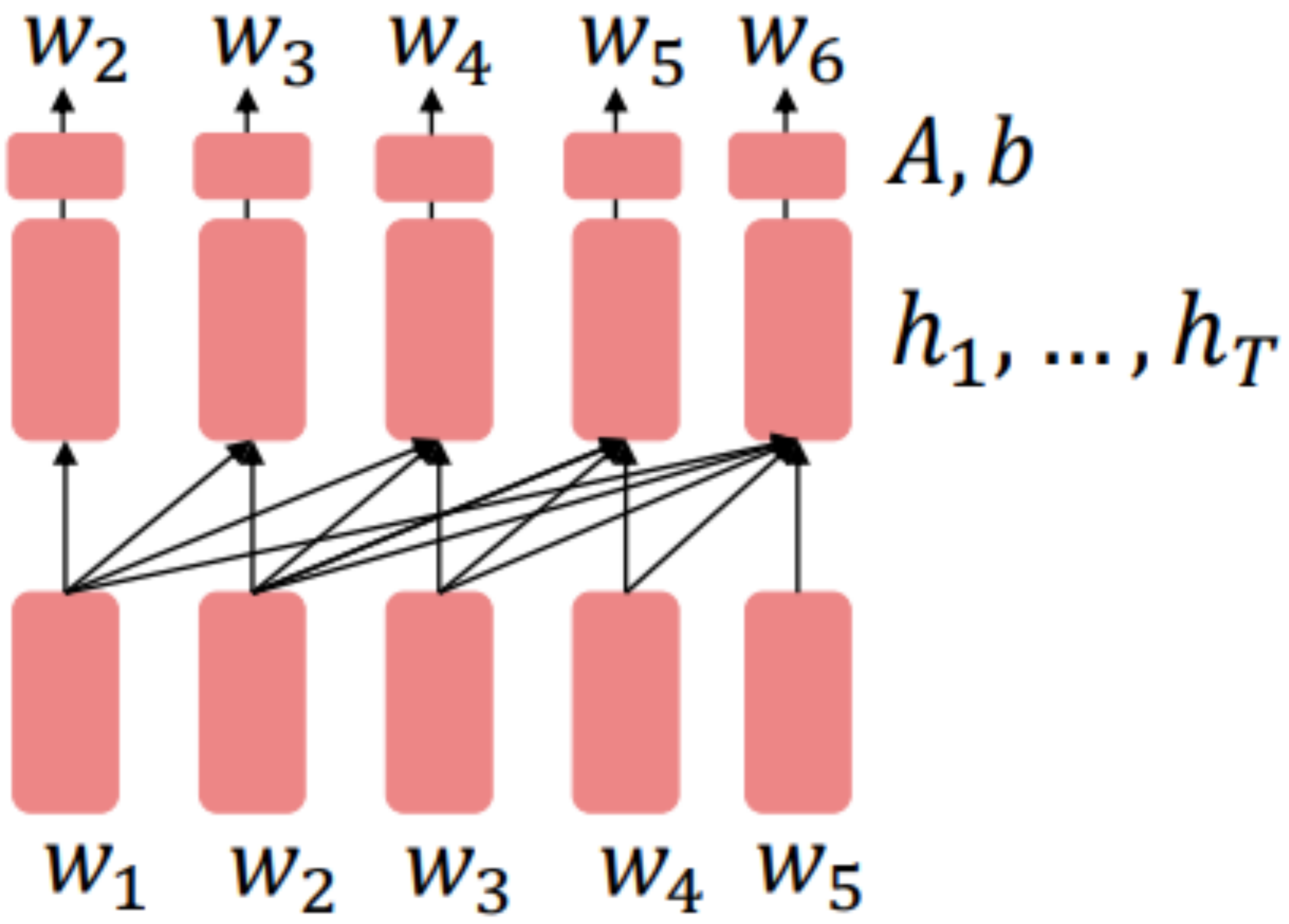


The linear layer hasn't been pretrained and must be learned from scratch.

Pretraining Decoders: Generators

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

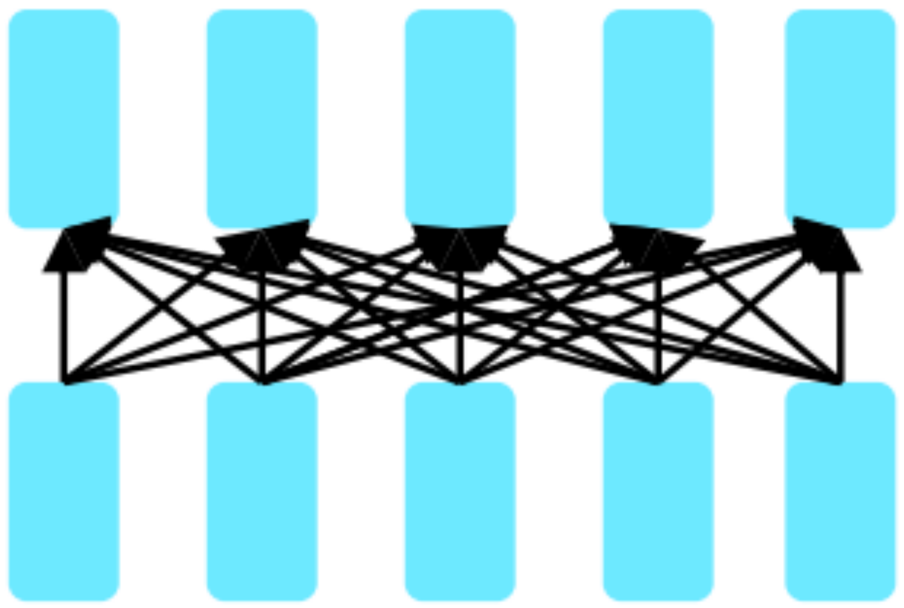
- This is helpful in tasks where the output is a sequence with a vocabulary like that at pretraining time!
  - Dialogue (context=dialogue history)
  - Summarization (context=document)



The linear layer has been pretrained

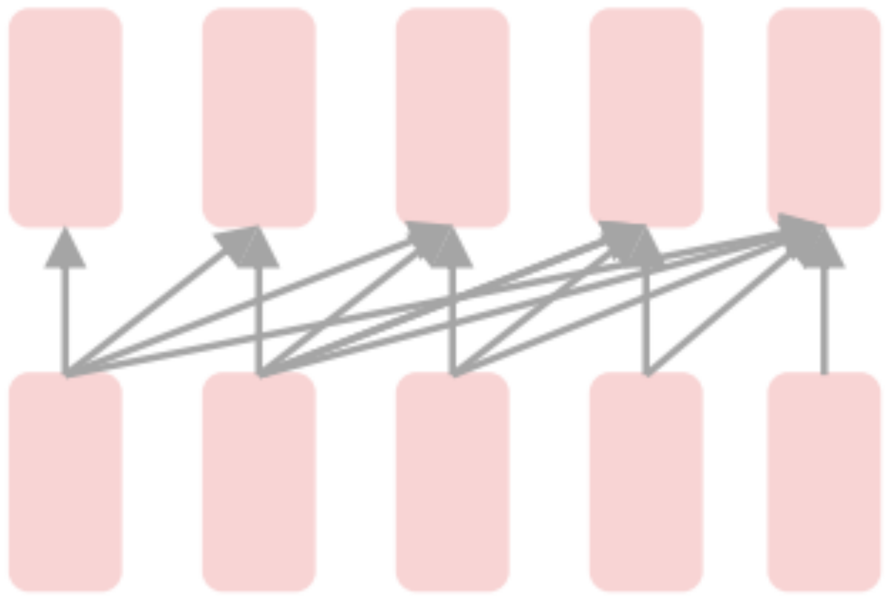
USC Viterbi

Pretraining for three types of architectures



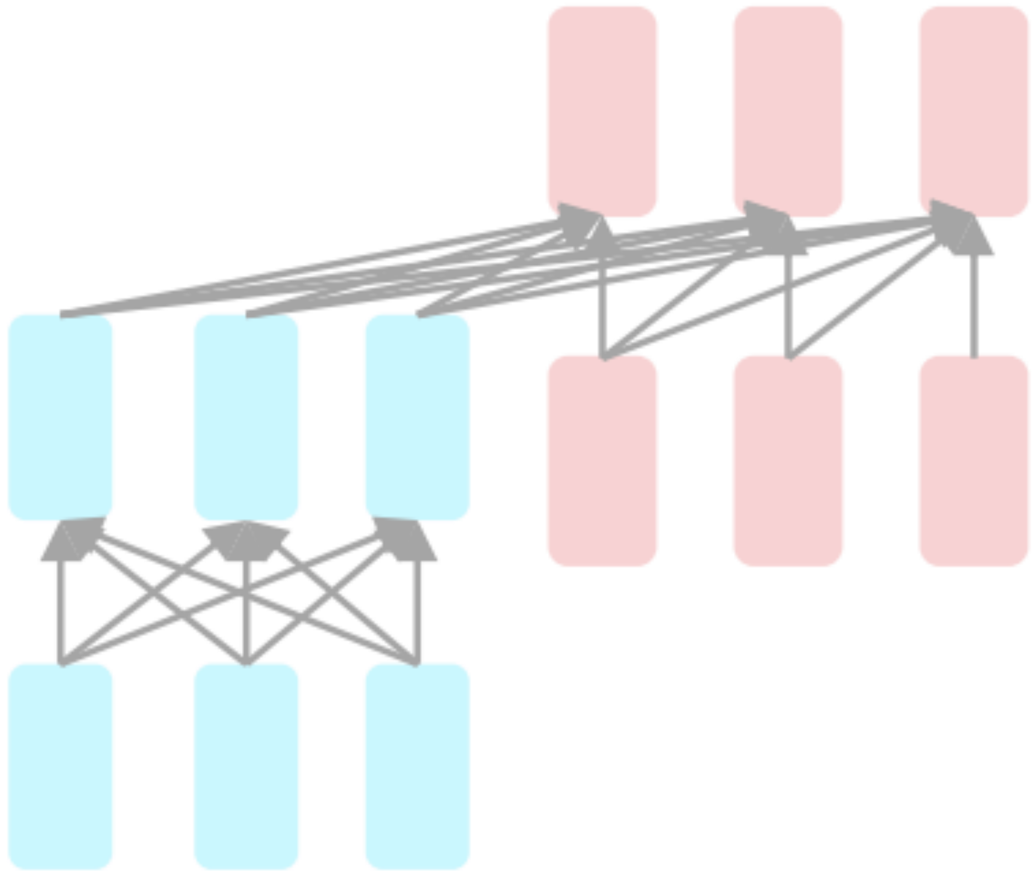
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

Sequence-to-sequence

USC Viterbi

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

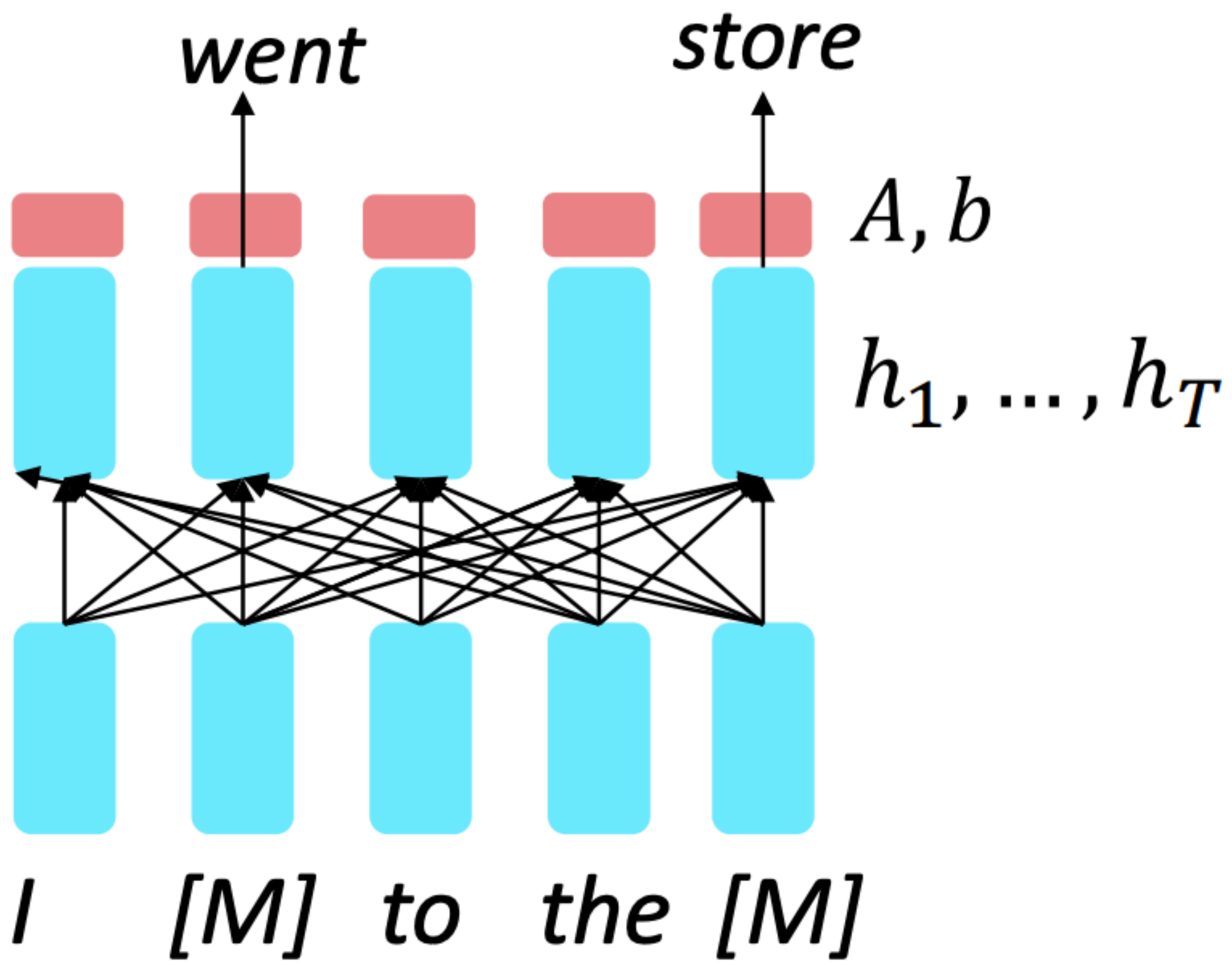
Problem: Input Reconstruction

'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.
  - $h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$
  - $y_i \approx Ah_i + b$



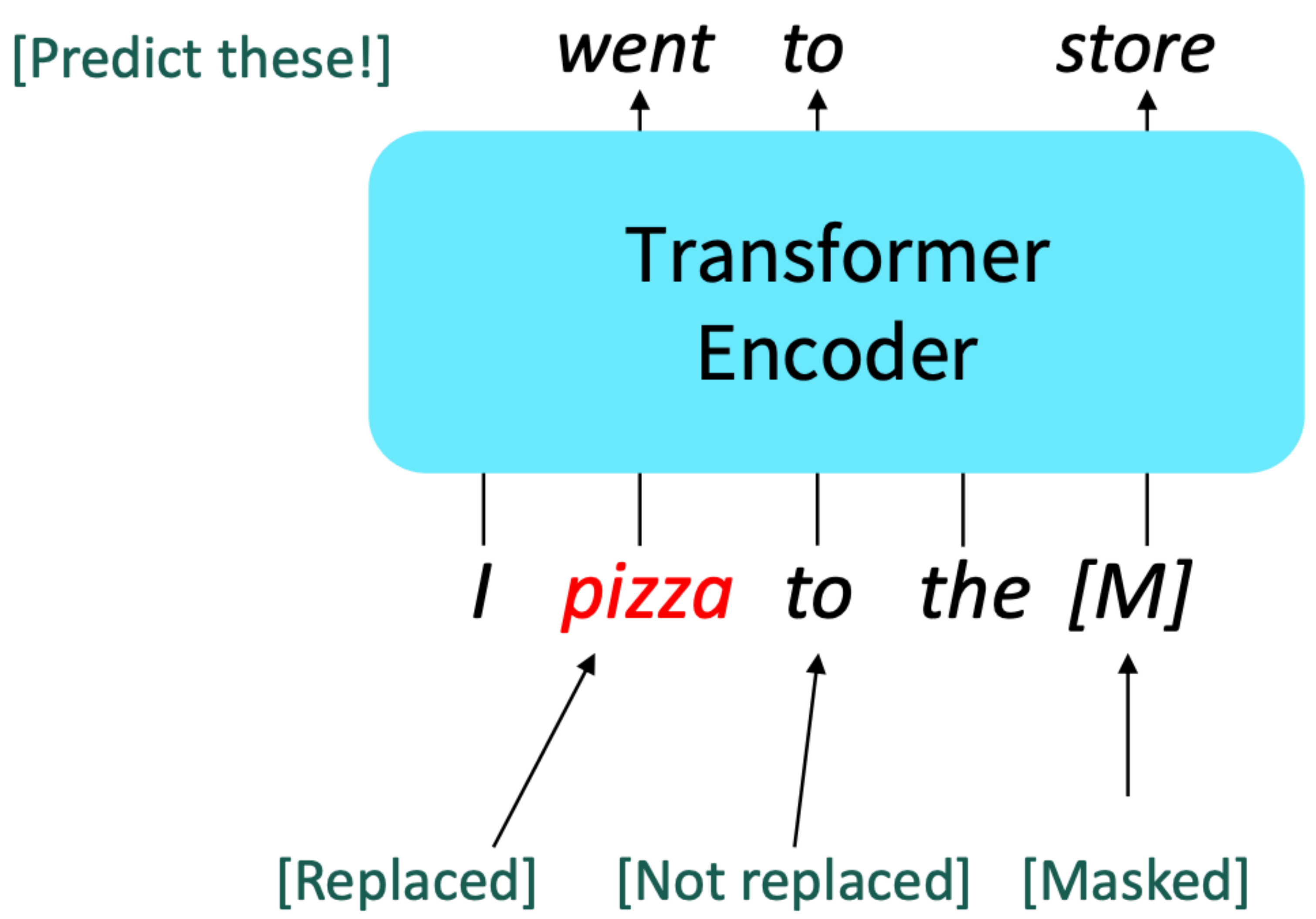
- Only add loss terms from words that are "masked out."
- If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(\tilde{x} \mid x)$.
- Called Masked LM
- Special type of language modeling

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:

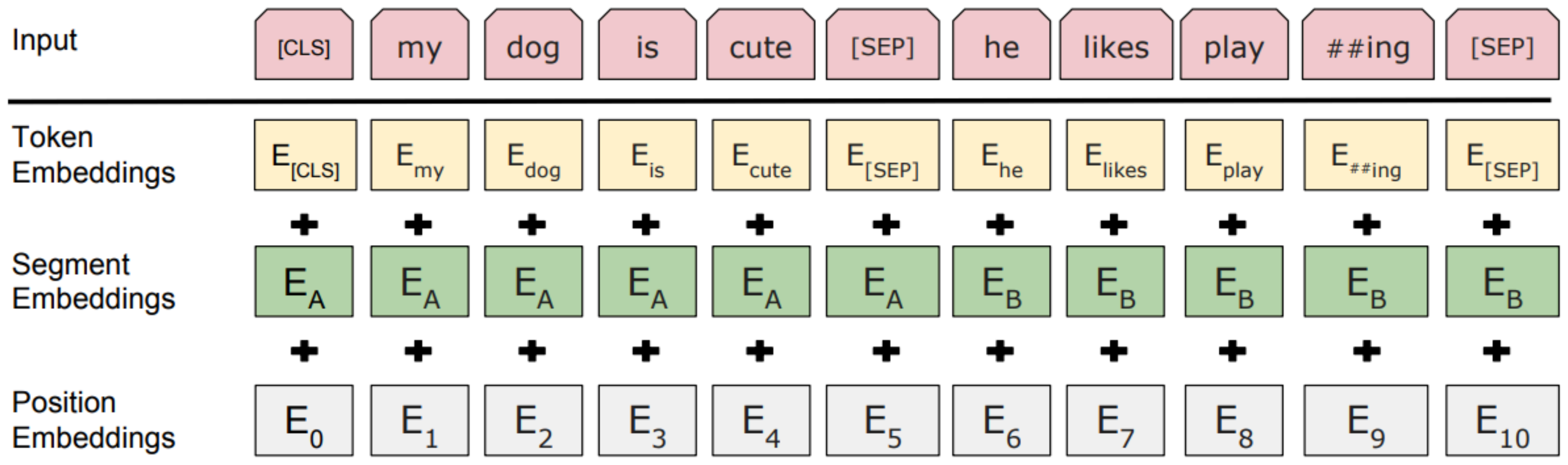
- 15% of the input tokens in a training sequence are sampled for learning, these are to be predicted by the model
- Of these
  - 80% are replaced with [MASK]
  - 10% are replaced with randomly selected tokens,
  - Remaining 10% are left unchanged



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:



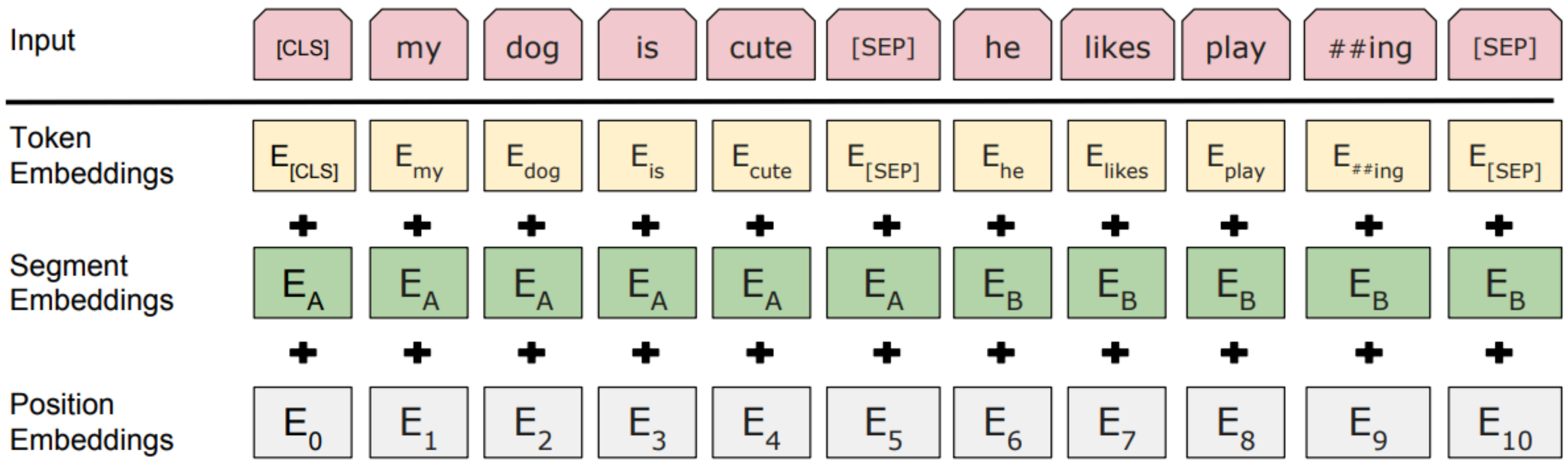
- BERT was trained to predict whether one chunk follows the other or is randomly sampled.
  - [CLS] and [SEP] tokens
  - [SEP] is used for next sentence prediction - do these sentences follow each other?
  - [CLS] for text classification / connection to fine-tuning

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."

USC Viterbi

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)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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

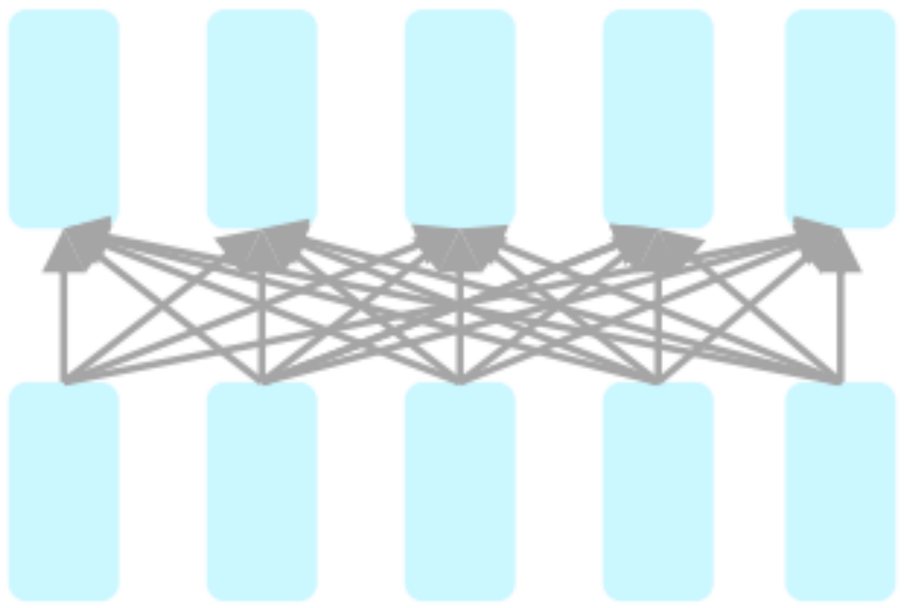
BERT: Extensions

- 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

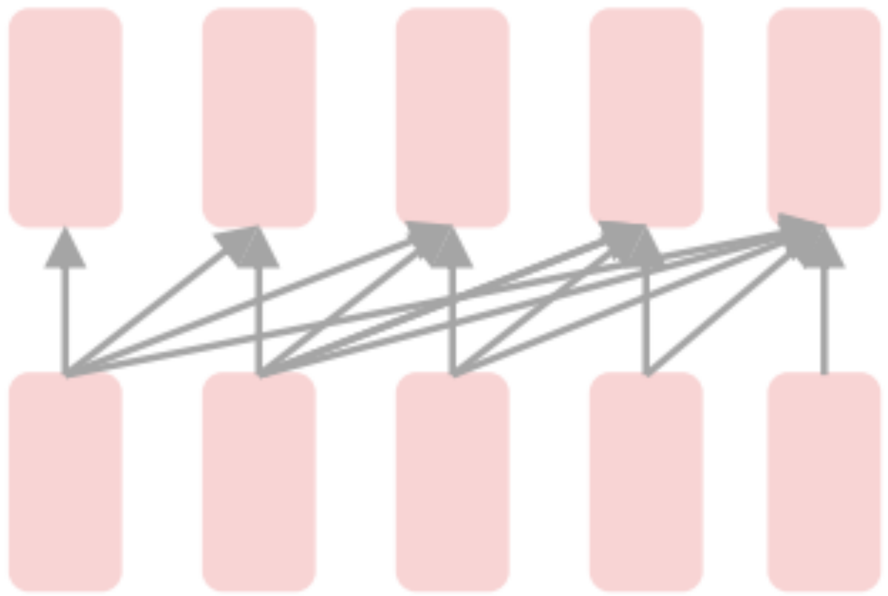
- [SEP]: Later work has argued this "next sentence prediction" is not necessary
- In general, more compute, more data can improve pretraining even when not changing the underlying Transformer encoder
- Results in contextual embeddings
- Key Limitation:
  - Cannot be used for generation
  - No pretraining 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!

Pretraining for three types of architectures



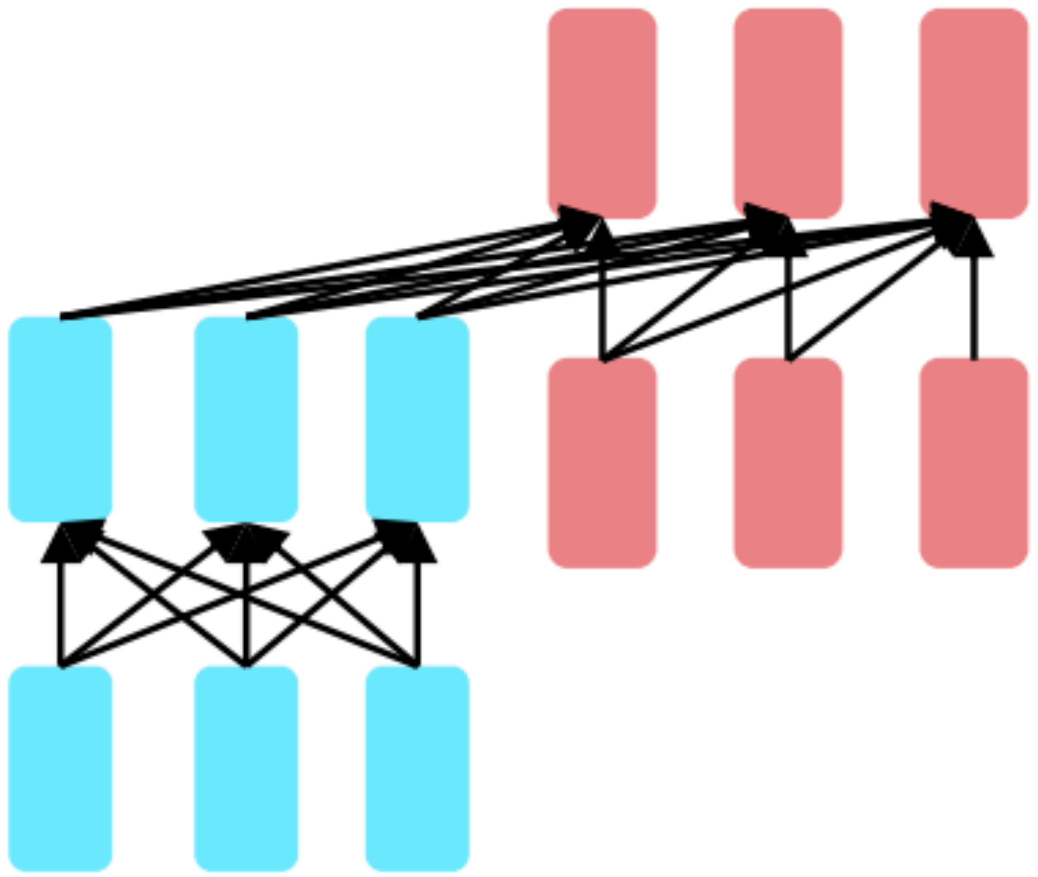
Encoders

Bidirectional Context



Decoders

Language Models



Encoder-Decoders

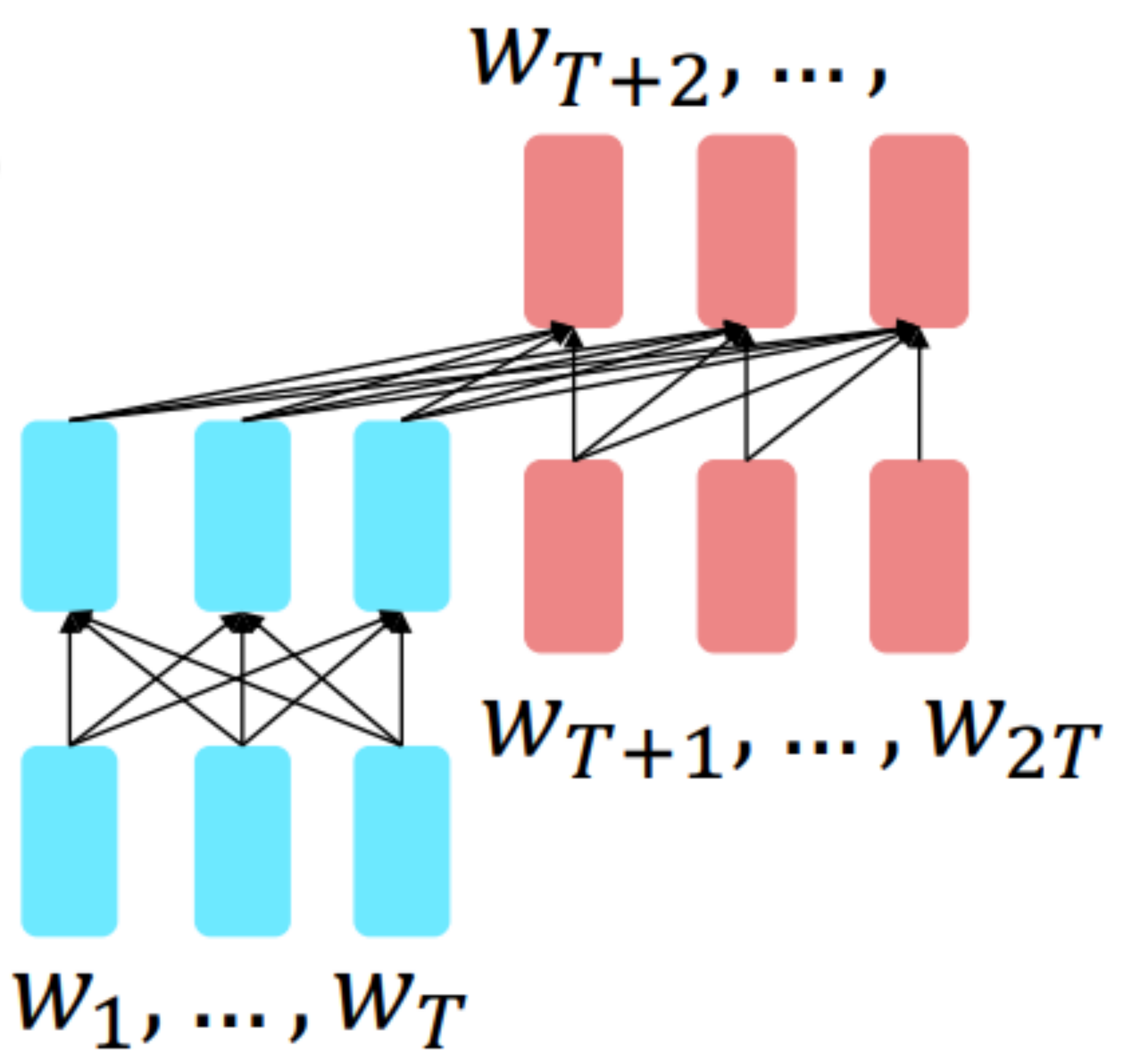
Sequence-to-sequence

USC Viterbi

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.

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



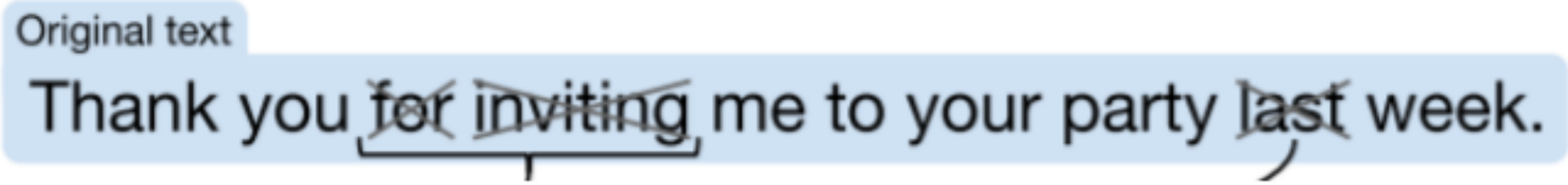
The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling.

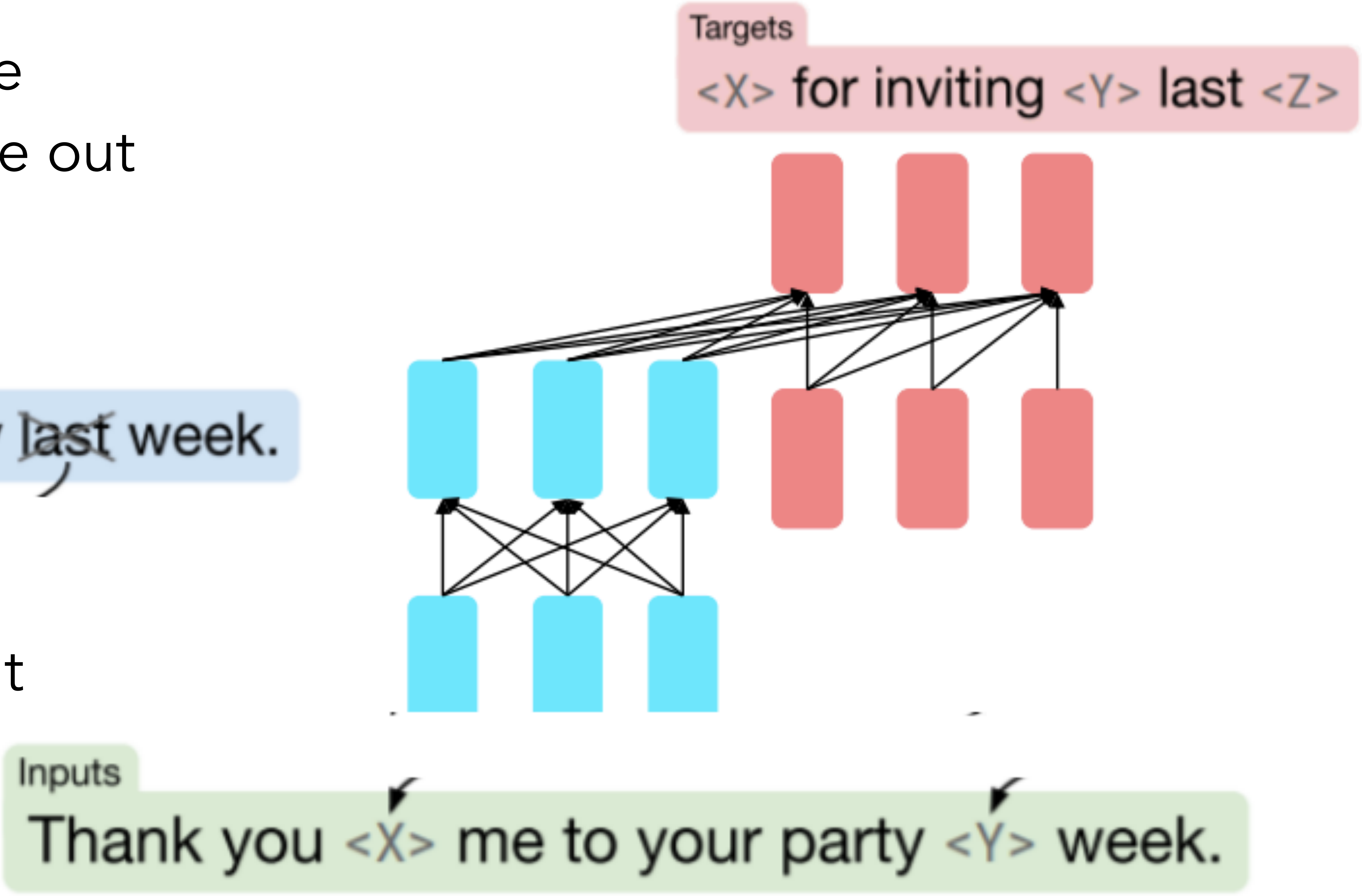
 USC Viterbi

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!

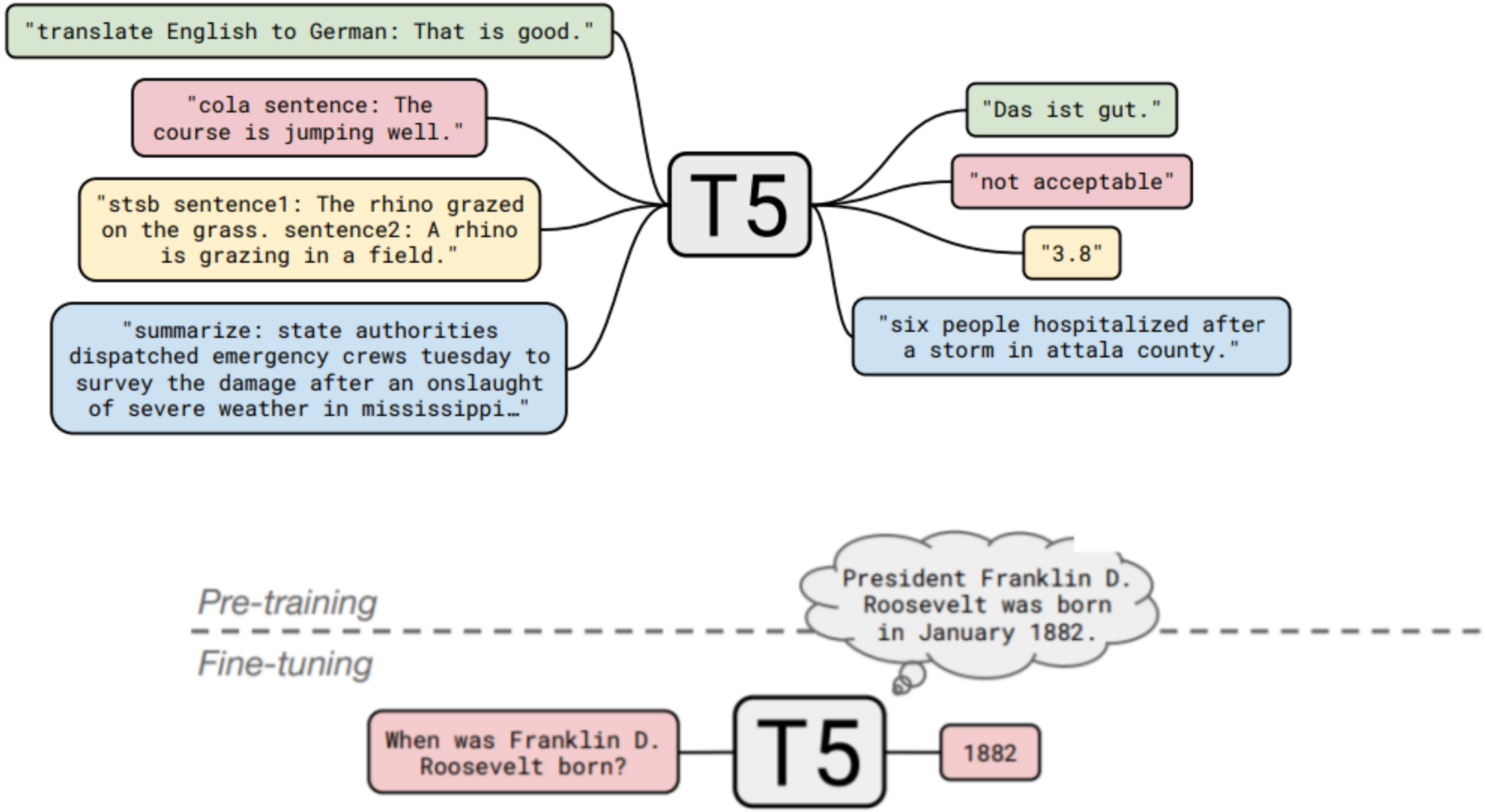




This is implemented in text preprocessing: it's still an objective that looks like language modeling at the decoder side.

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.

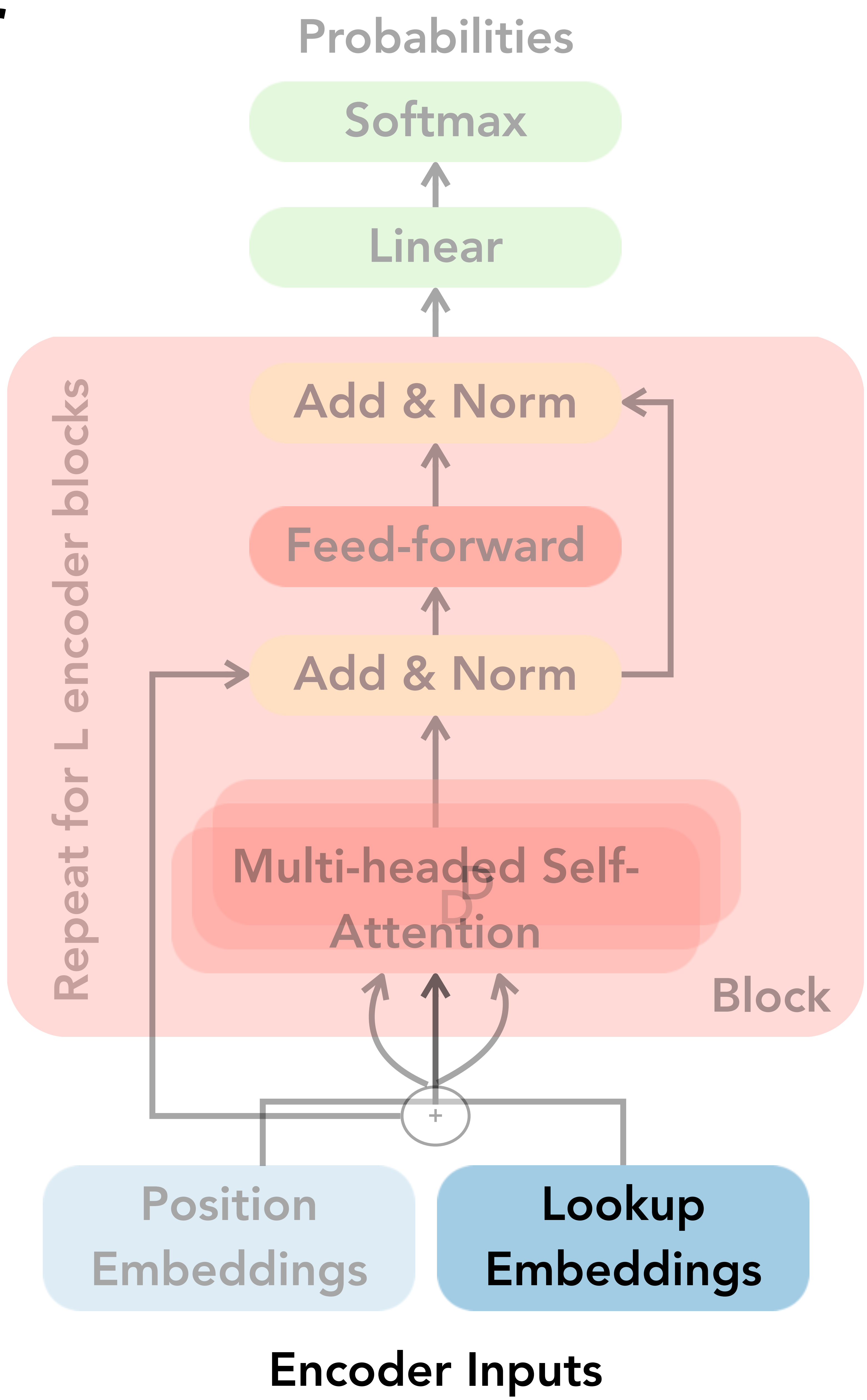
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	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

Tokenization in Transformers

USC Viterbi

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

USC Viterbi

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	

Polarity	Persons 1st Sg.	Persons 1st Pl.	Persons 2nd Sg.	Persons 2nd Pl.	Persons / Classes 3rd / M-wa Sg. / 1	Persons / Classes 3rd / M-wa Pl. / 2	Classes M-mi 3	Classes M-mi 4	Classes Ma 5	Classes Ma 6	Classes Ki-vi 7	Classes Ki-vi 8	Classes N 9	Classes N 10	Classes U 11 / 14	Classes Ku 15 / 17	Classes Pa 16	Classes Mu 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 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>)

Hello how are U tday?

↓

hello how are u tday?

↓

[hello, how, are, u, tday, ?]

↓

[hello, how, are, u, td, ##ay, ?]

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!
     - 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>

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)	

Vocabulary

d	e	i	l	n	o	s	t	w
es	est							

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

USC Viterbi

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/