Large Language Models

Part II: Encoder-only and Encoder-Decoder Models

Aaron Mueller
CAS CS 505: Introduction to Natural Language Processing
Boston University
Spring 2026

Admin

- **HW1** is due tonight at 11:59pm!

Overview of Concepts

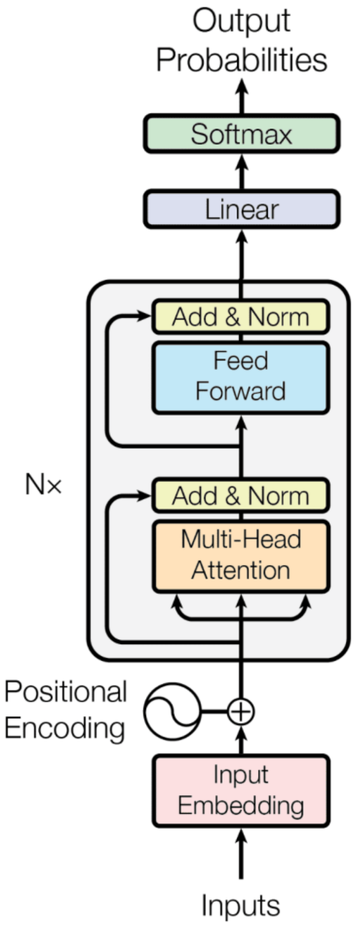
Encoder models include those like BERT and RoBERTa.

Contextual embeddings are representations of tokens that take their surrounding context into account.

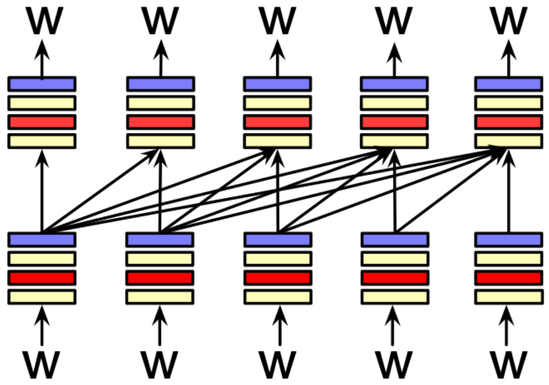
Masked language modeling and **next sentence prediction** are the training objectives used to train encoder models.

Encoder-decoder models include those like T5.

Natural language inference, **question answering**, and **word sense disambiguation** are common applications of encoder(-decoder) models.



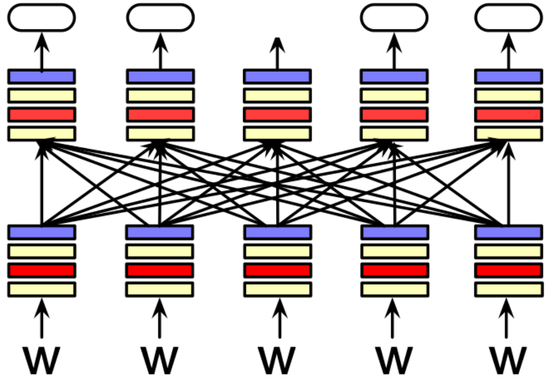
Architectures



Decoder

Left-to-right: generate the next word given prior context. Language models!

Good for generation.

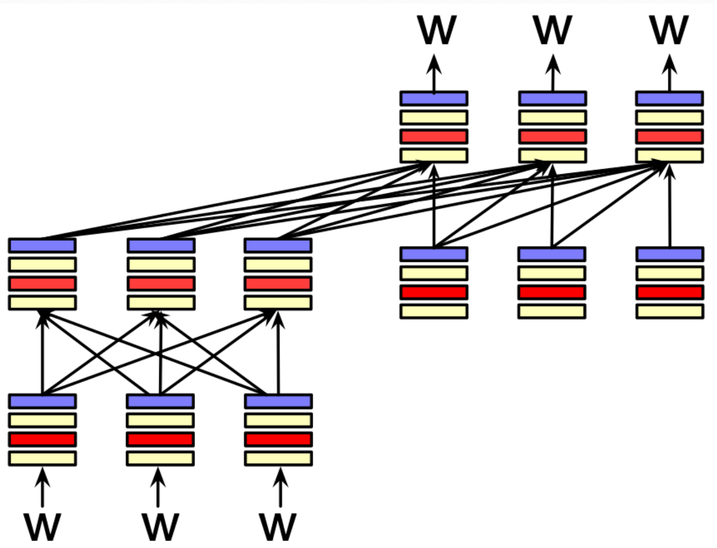


Encoder

Bidirectional: predict a word given left *and* right context.

Can only generate 1 token.

Good for classification.



Encoder-Decoder

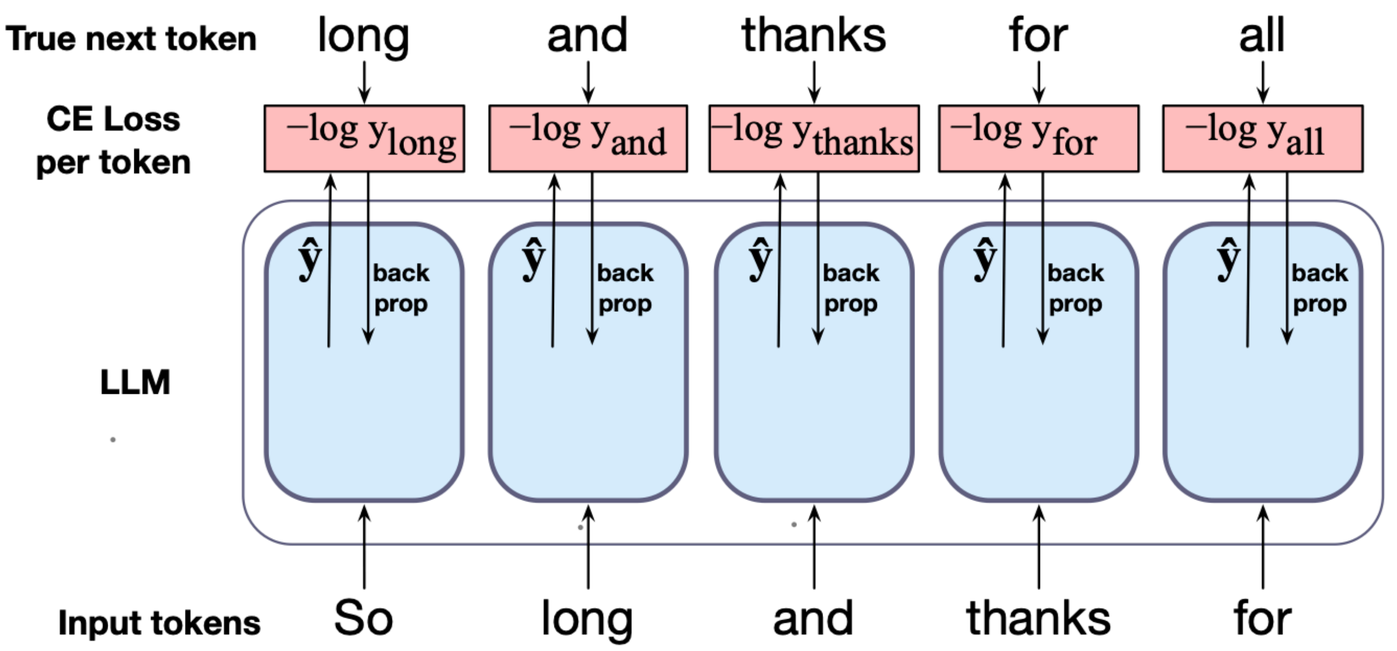
Best of both worlds, maybe?

Used to be more common; not so much anymore.

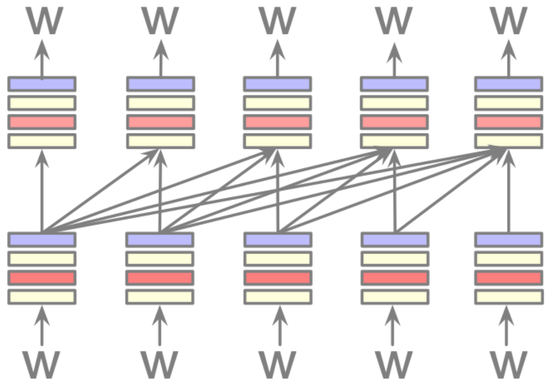
Last Time

Recall from last class that **decoder(-only)** models process inputs left-to-right.

Today, we'll discuss a very different kind of model that does not process text in this way: a bidirectional encoder.



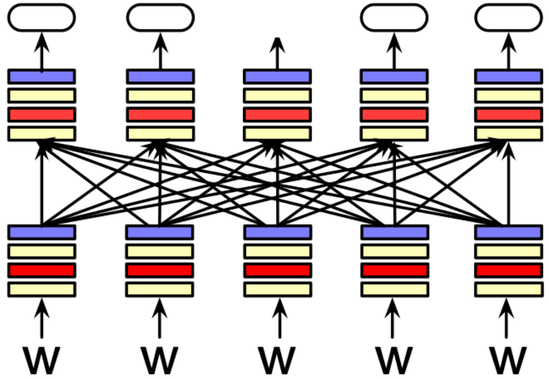
Architectures



Decoder

Left-to-right: generate the next word given prior context. Language models!

Good for generation.

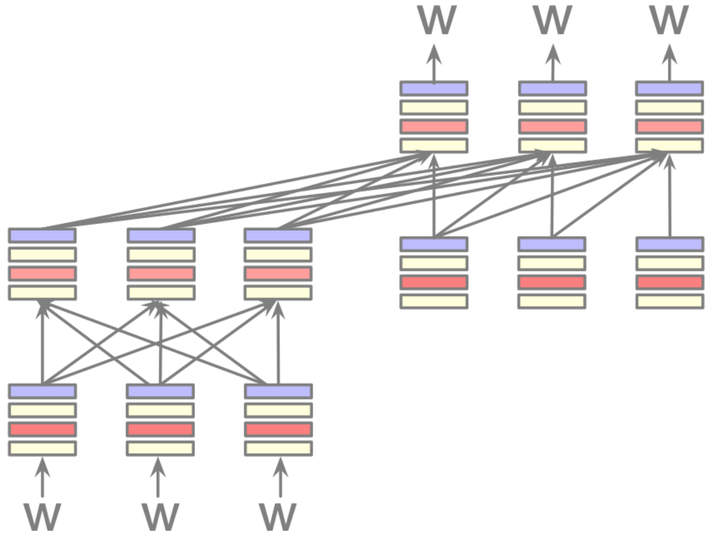


Encoder

Bidirectional: predict a word given left *and* right context.

Can only generate 1 token.

Good for classification.



Encoder-Decoder

Best of both worlds, maybe?

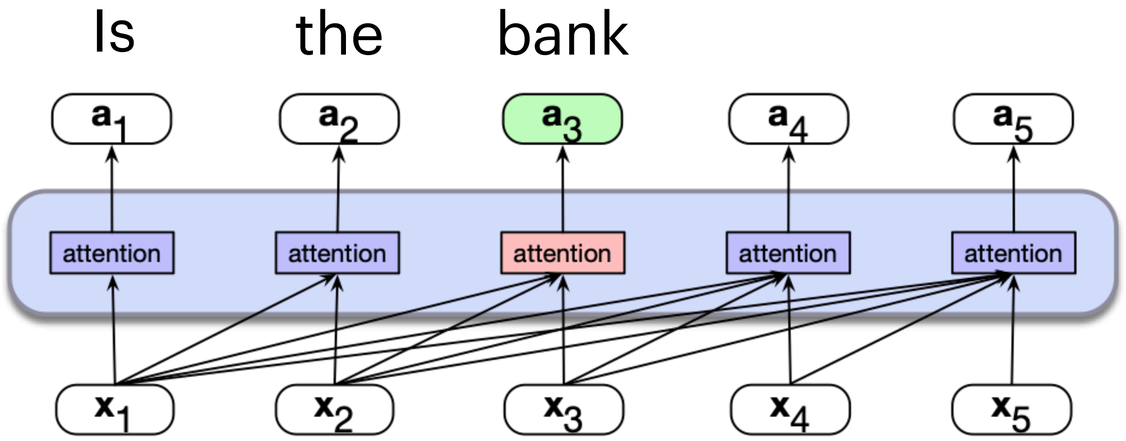
Used to be more common; not so much anymore.

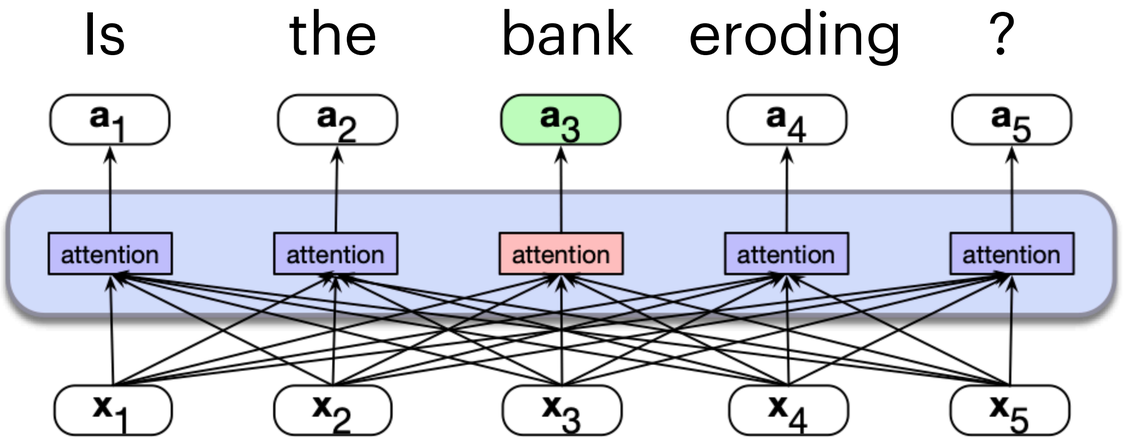
History

- ELMo (based on LSTMs) was released in spring 2018
  - GPT (Transformer-based ELMo) released in summer 2018
  - BERT released in fall 2018

- Ways that BERT differed from ELMo:
  - Transformers instead of LSTMs
  - Bidirectional model with “masked language modeling” objective
  - Operates over word pieces (BPE)

Bidirectional Encoders





Causal self-attention (decoder)

Each token's hidden representation is a function of the current token *and* prior tokens to its left.

Can be used for generating sequences.

Bidirectional self-attention (encoder)

Each token's hidden representation is a function of the left *and* right context.

Cannot generate sequences.

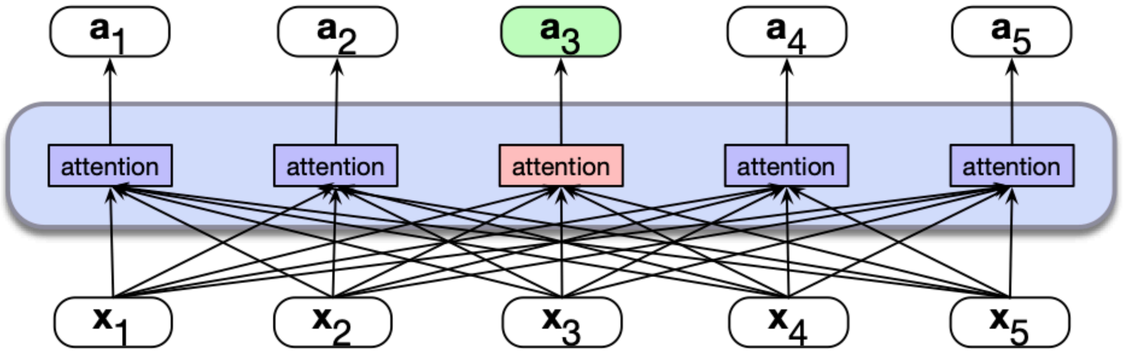
Both compute contextualized representations of a token.

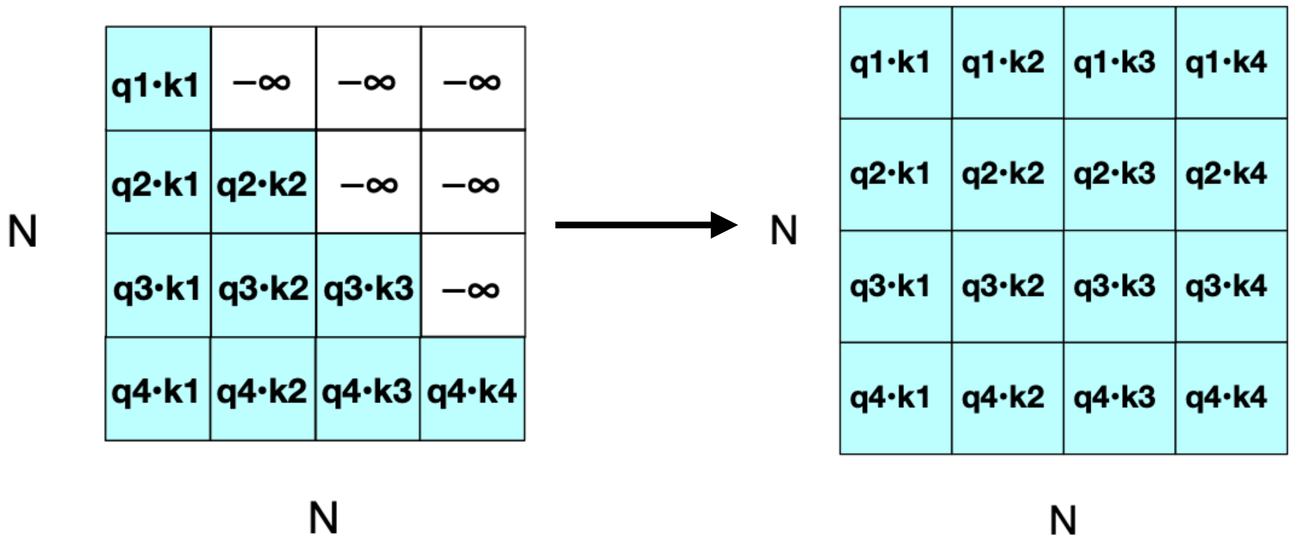
Implementing Bidirectional Encoders

How do we implement this?

Simple: just remove the causal mask when computing the attention scores!

$$\textbf{head} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$





Training Bidirectional Encoders

The Cloze Task

- Because we aren't masking the right context, the original language modeling loss function doesn't work.
- What is the bidirectional equivalent of language modeling?
- Consider something like a **cloze task**:

Yesterday, Abby went to the store with _____ dog.

Are we _____ yet?

- This will form the foundation of our training objective.

Training Bidirectional Encoders

Masked Language Modeling

Idea: replace some fraction of tokens in the input with a special `[MASK]` token; have the model predict these tokens.

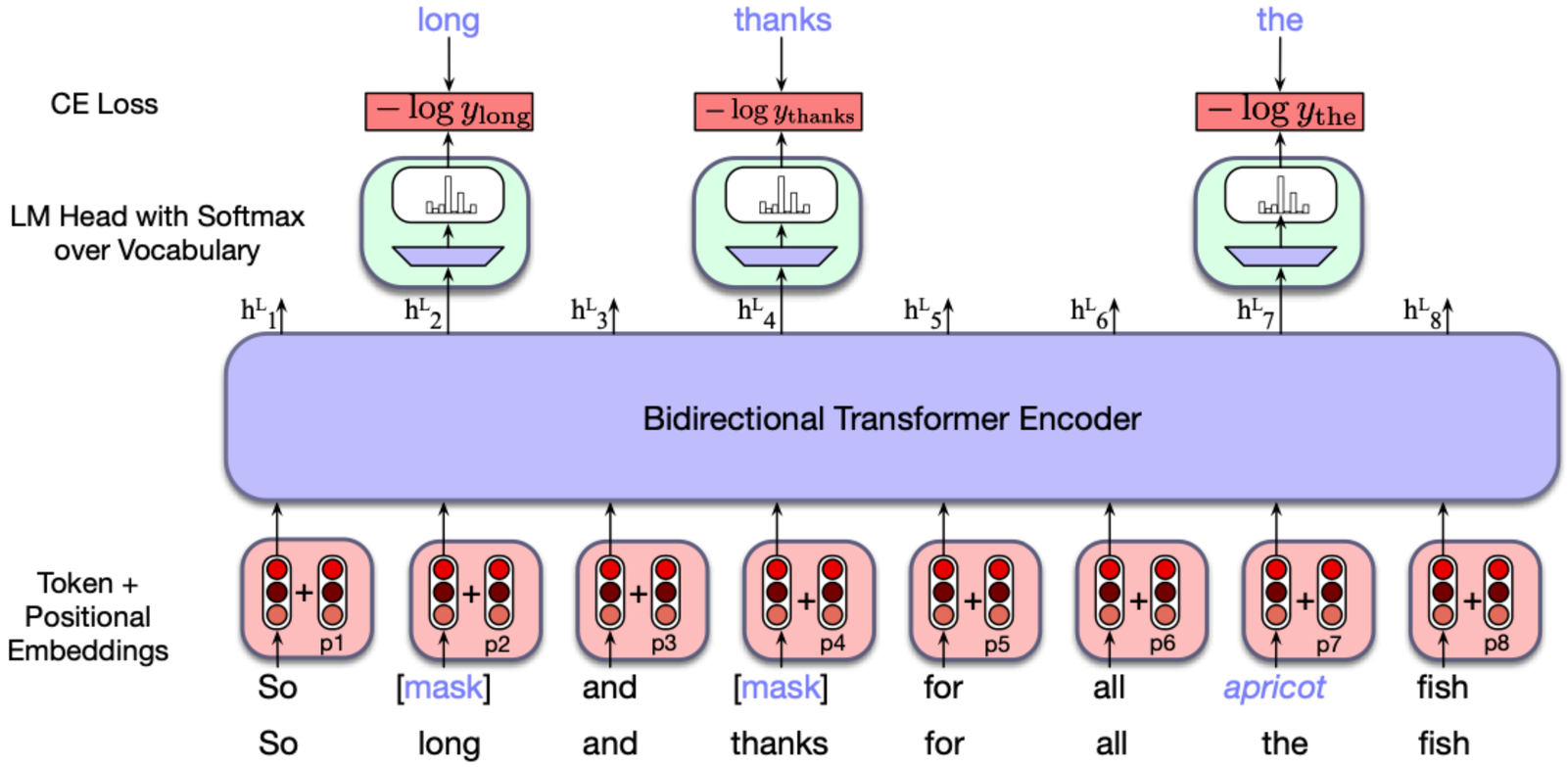
$$\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T = \text{Encoder}(w_1, w_2, \ldots, w_T)$$

$$\mathbf{u}_i = \mathbf{h}_i^L E^\top$$

$$\mathbf{y}_i = \text{softmax}(\mathbf{u}_i)$$

$$L_{\text{MLM}}(x_i) = -\log p(x_i \mid \mathbf{h}_i^L)$$

We *only* compute losses over the masked tokens.

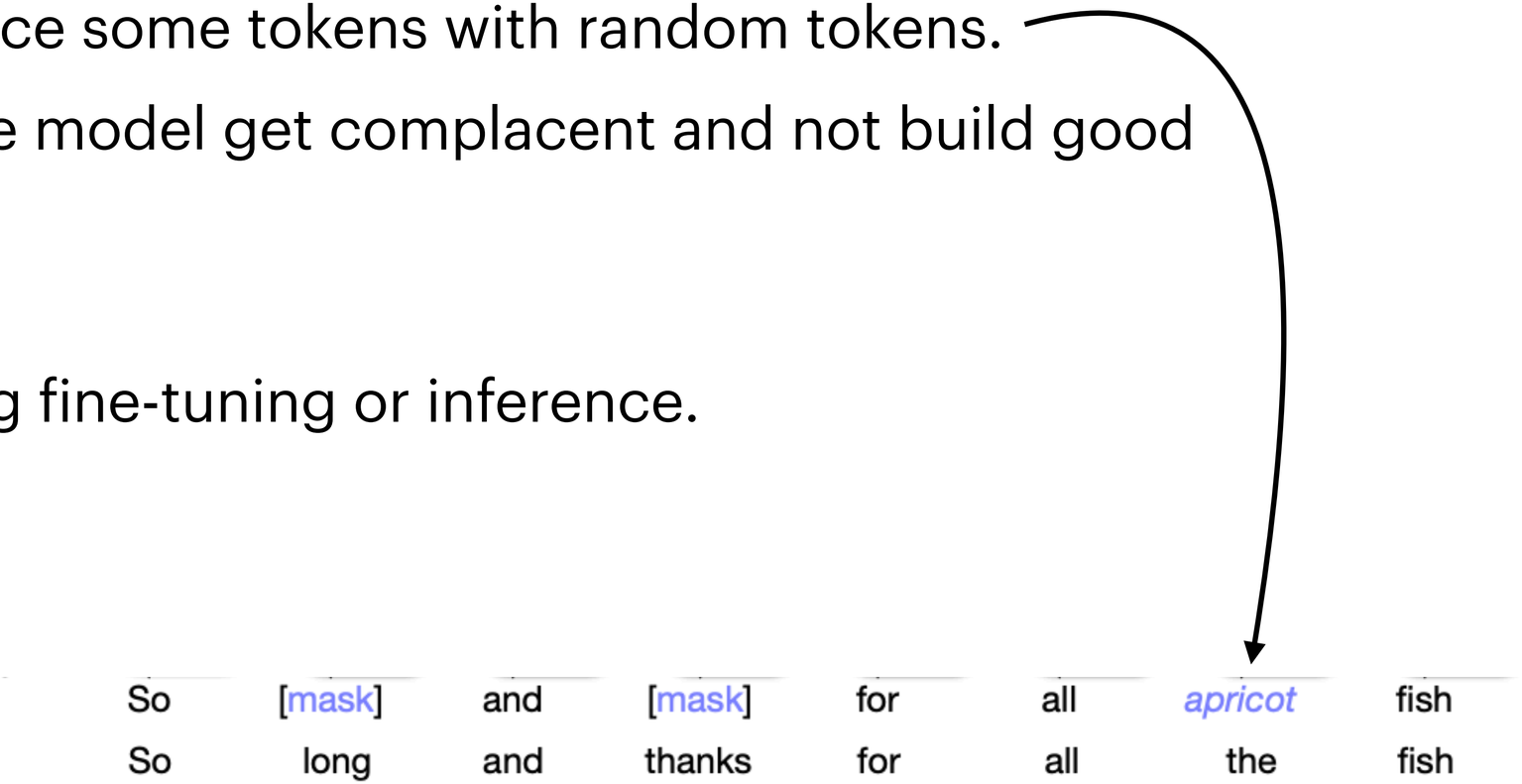


Training Bidirectional Encoders

Masked Language Modeling

- In addition, we also replace some tokens with random tokens.
  - Why? We can't let the model get complacent and not build good representations!

- No masks are seen during fine-tuning or inference.



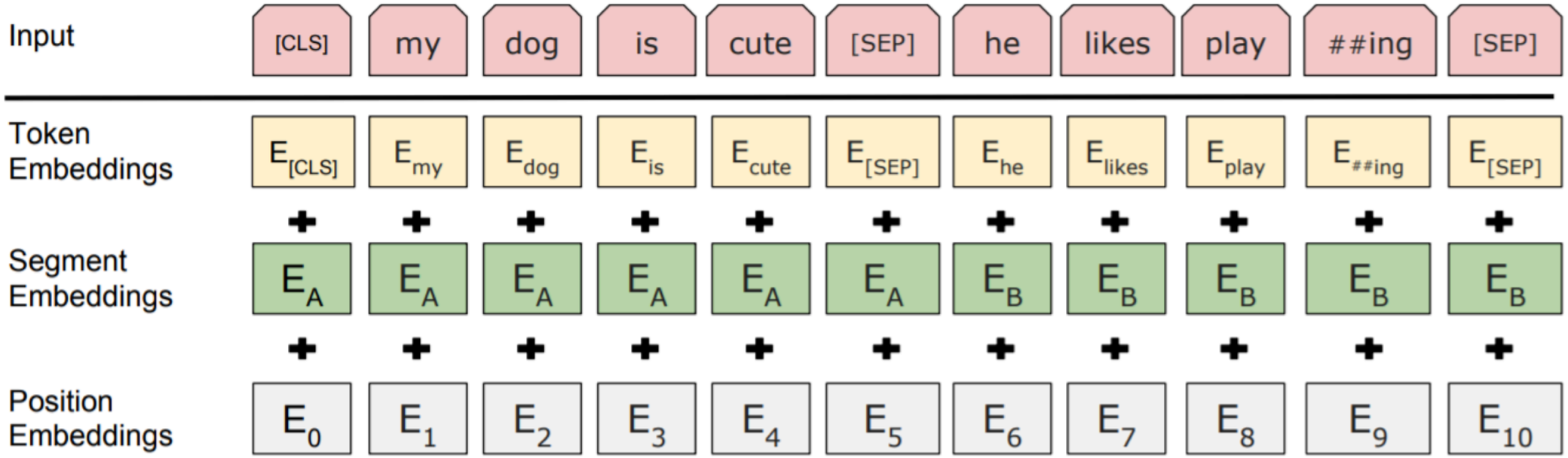
BERT: Bidirectional **E**ncoder **R**epresentations from **T**ransformers

- In 2018, Devlin et al. proposed the MLM objective, and **released the weights of a pre-trained Transformer encoder**.
- This model was called **BERT**. Here are some training hyperparameters:
  - Select a random 15% of tokens. For each token:
    - Replace 80% of these with a [MASK] token.
    - Replace with a random token 10% of the time.
    - Leave the remaining 10% of tokens unchanged.
  - Then, predict the selected tokens given the hidden state.
- They also proposed a second loss term: the next sentence prediction loss.

BERT: Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Next Sentence Prediction

- The pretraining input is two separate sequences of text, separated by a [SEP] token:



- In addition to the MLM loss, BERT was also trained to predict whether one sequence actually followed the other in the training corpus or was randomly sampled
  - This is the **next sentence prediction (NSP)** objective

BERT: Bidirectional **E**ncoder **R**epresentations from **T**ransformers

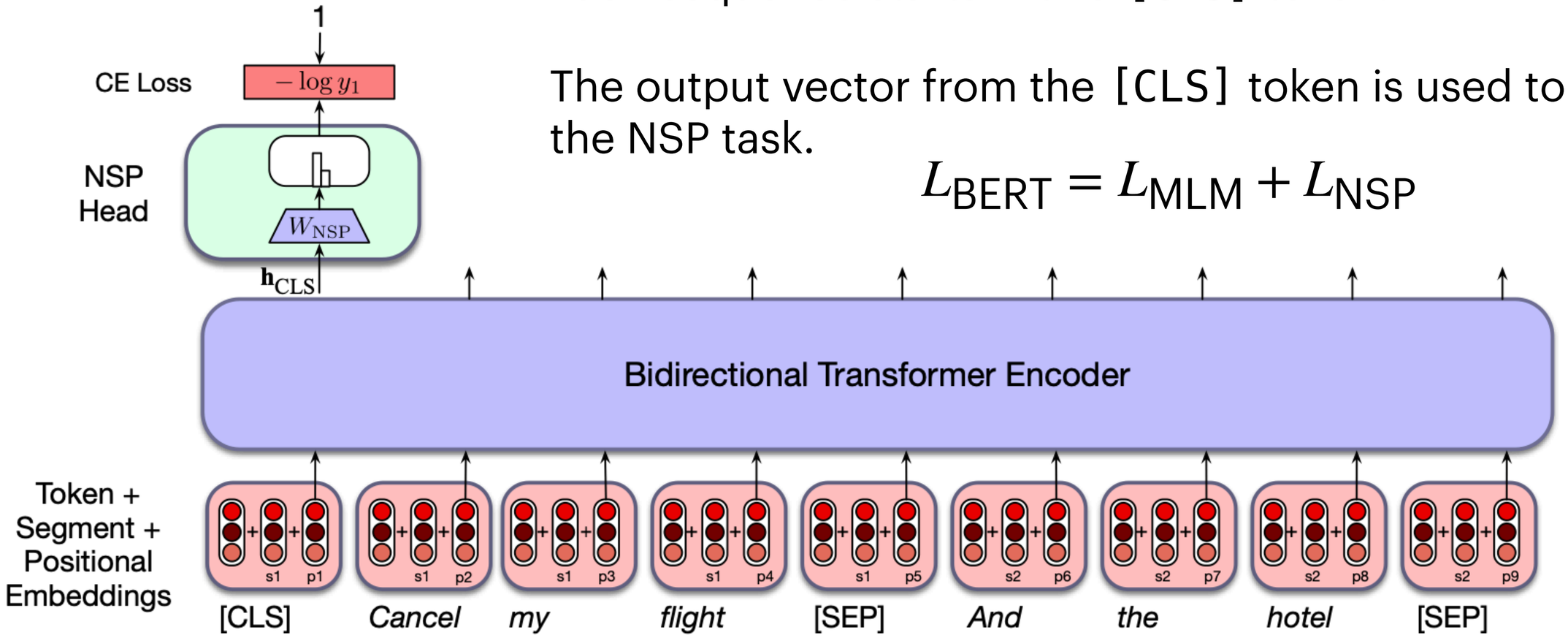
$$\mathbf{y}_1 = \text{softmax}(\mathbf{h}^L_{\text{CLS}} W_{\text{NSP}})$$

Next Sentence Prediction

Each sequence starts with a `[CLS]` token.

The output vector from the `[CLS]` token is used to do the NSP task.

$$L_{\text{BERT}} = L_{\text{MLM}} + L_{\text{NSP}}$$

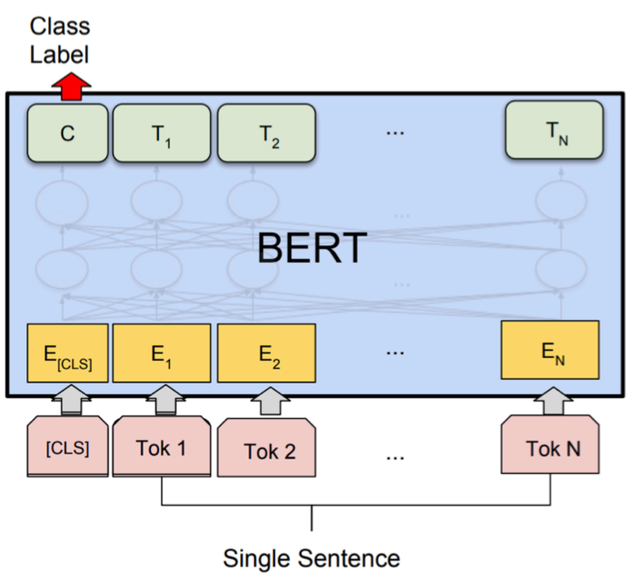


BERT: Bidirectional **E**ncoder **R**epresentations from **T**ransformers

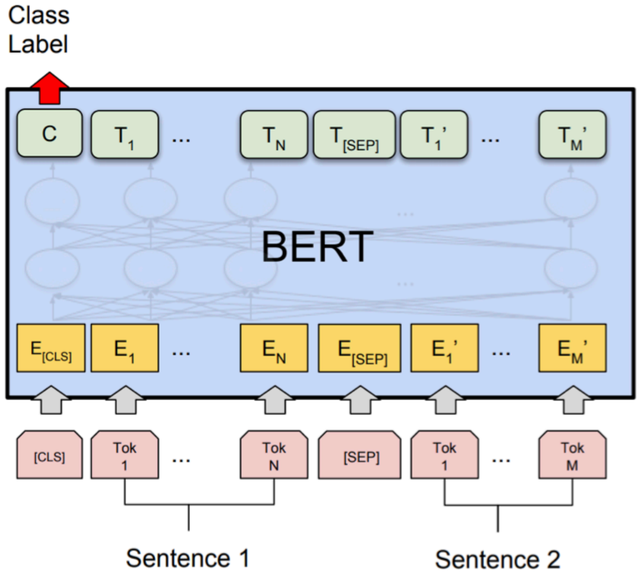
Further Details

- Two versions were released:
  - BERT-base: 12 layers, 768-dimensional hidden states, 12 attention heads, 110M parameters
  - BERT-large: 24 layers, 1024-dimensional hidden states, 16 attention heads, 340M parameters
- Training data:
  - BooksCorpus (3.3 million words)
  - English Wikipedia (2.5 **b**illion words)
- They had to use 64 TPUs for 4 days to train this model!
- However, fine-tuning is practical on just 1 GPU.

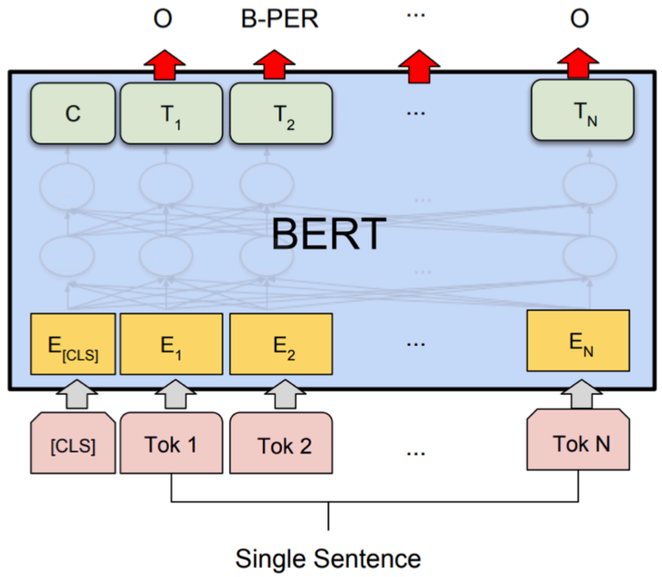
What Can BERT Do?



(b) Single Sentence Classification Tasks: SST-2, CoLA



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

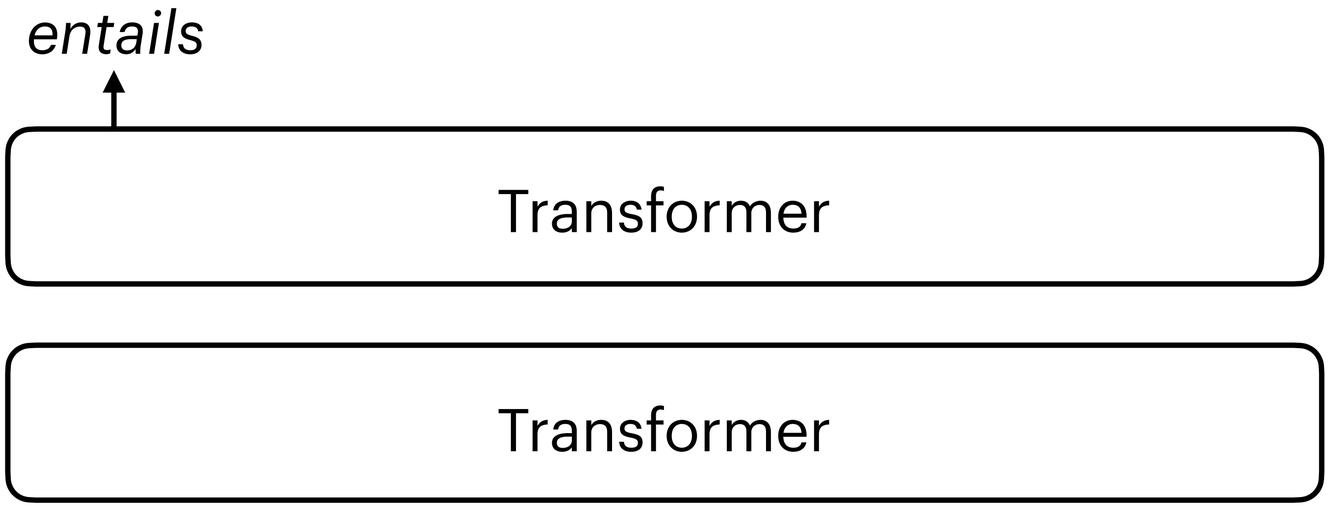
- [CLS] token representation is used as the input to a text classifier
- Sentence pair classification: feed both sentences into BERT
- Can also do sequence tagging by predicting tags at every token

Natural Language Inference (NLI)

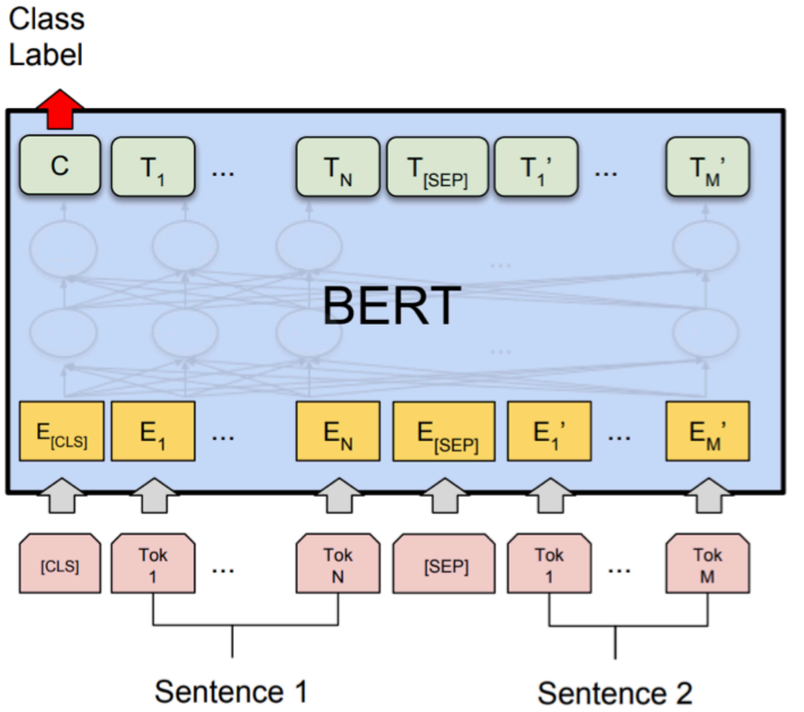
Premise		Hypothesis
He eats chicken	*entails*	He is not vegetarian
The cat is asleep	*neutral*	The cat is dreaming of fish
I am teaching	*contradiction*	I am on vacation

Among the earliest tasks used to distinguish models that could “reason” from those that couldn’t.

What Can BERT Do?



[CLS] He eats chicken [SEP] He is not vegetarian [SEP]



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

- Why is BERT able to model pairs of sentences?
- Transformers can capture interactions between sentences

The Rise of Fine-tuning

The BERT paper was the first time this paradigm of pre-train-then-fine-tune became massively popular. Why?

- BERT was *hugely* versatile: it led to state-of-the-art results on many tasks!

Natural language inference *Paraphrase detection* *Textual similarity detection* *Grammaticality judgment*

Sentiment classification

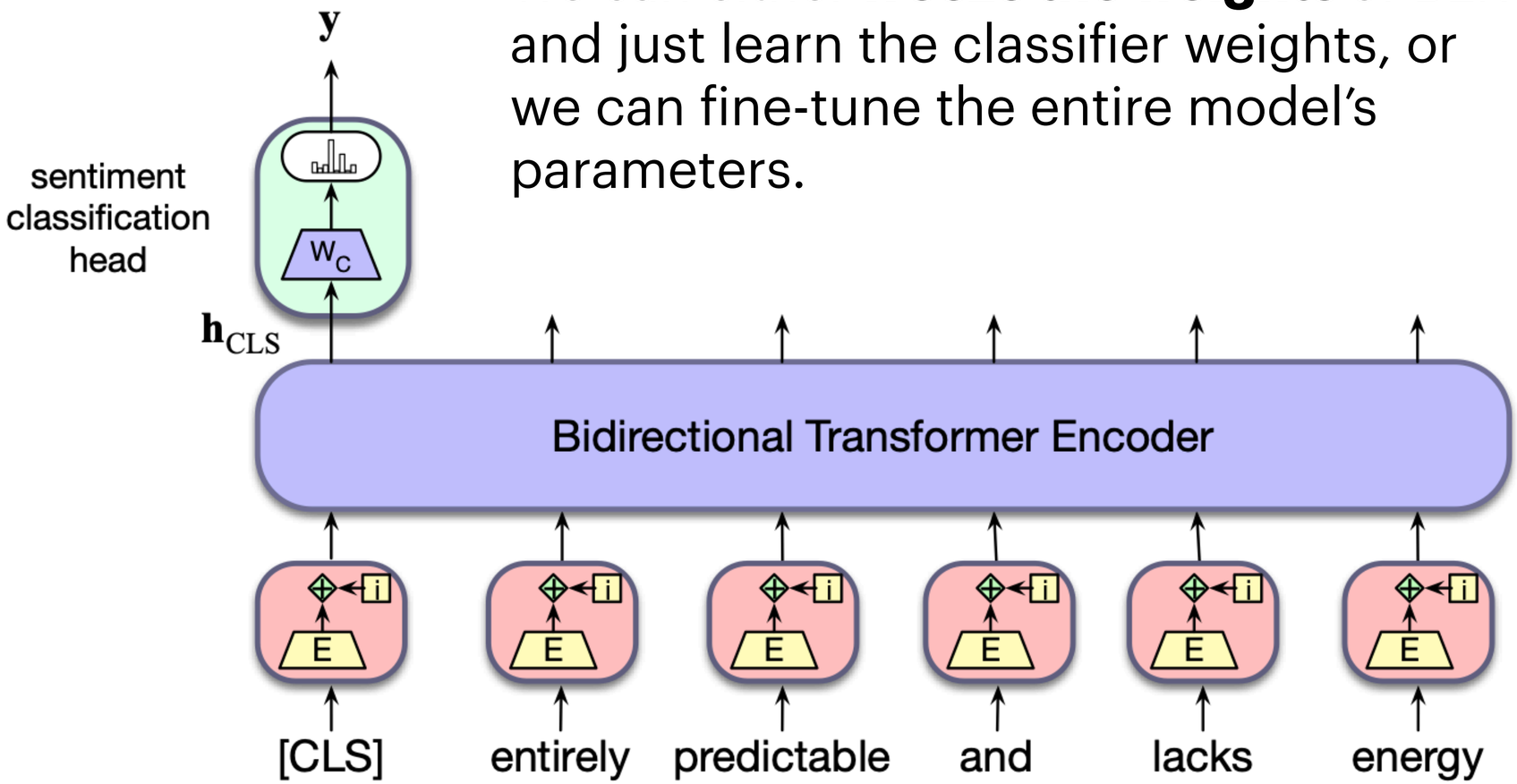
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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**

Fine-tuning BERT

During fine-tuning, we can co-opt the [CLS] token to do sequence classification tasks!

We learn weights W_C on top of the CLS representation $\mathbf{h}_{\text{CLS}}$ to score each category:

$$\mathbf{y} = \text{softmax}(\mathbf{h}^L_{\text{CLS}} W_C)$$



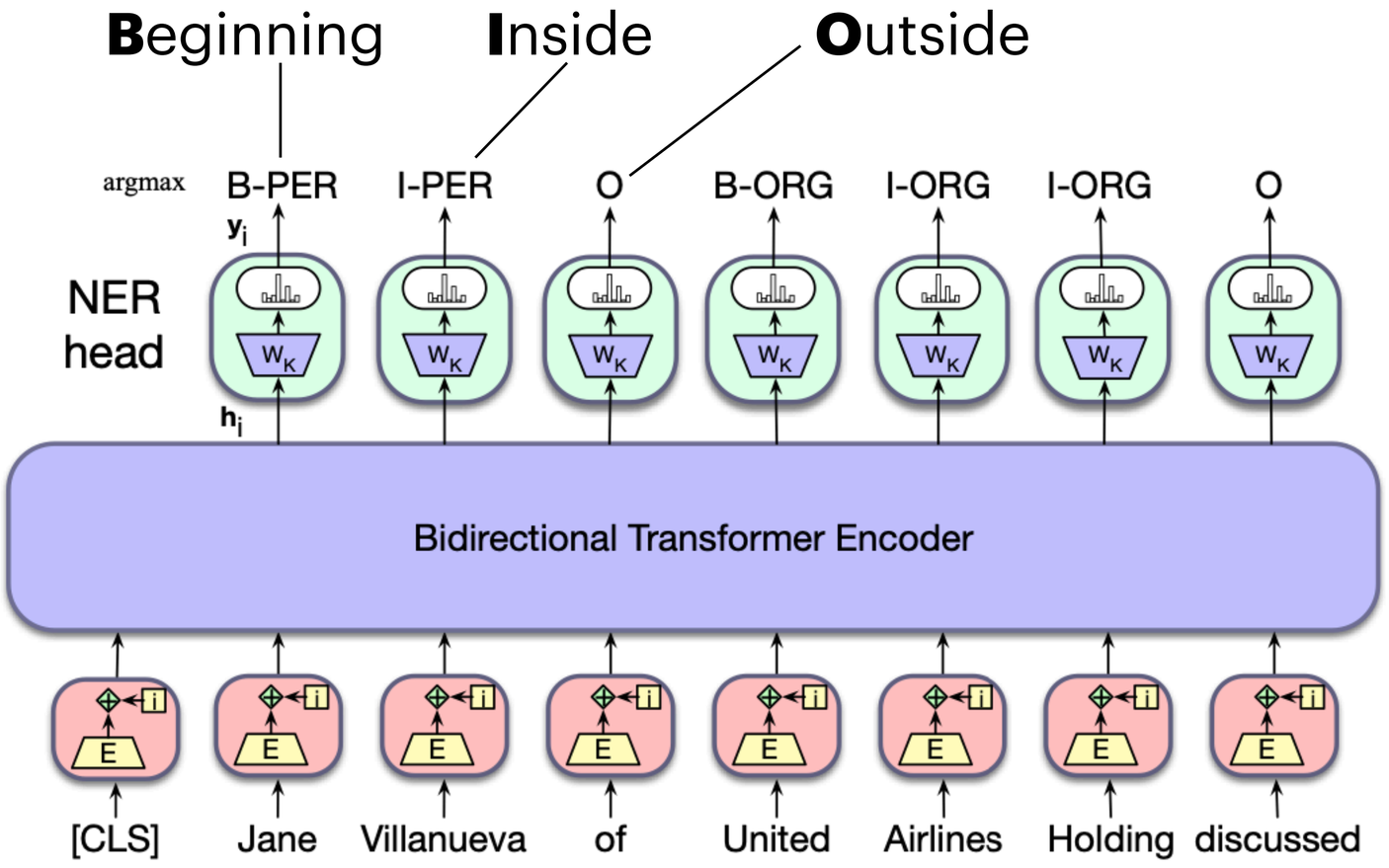
We can either **freeze the weights** of BERT and just learn the classifier weights, or we can fine-tune the entire model's parameters.

Fine-tuning BERT

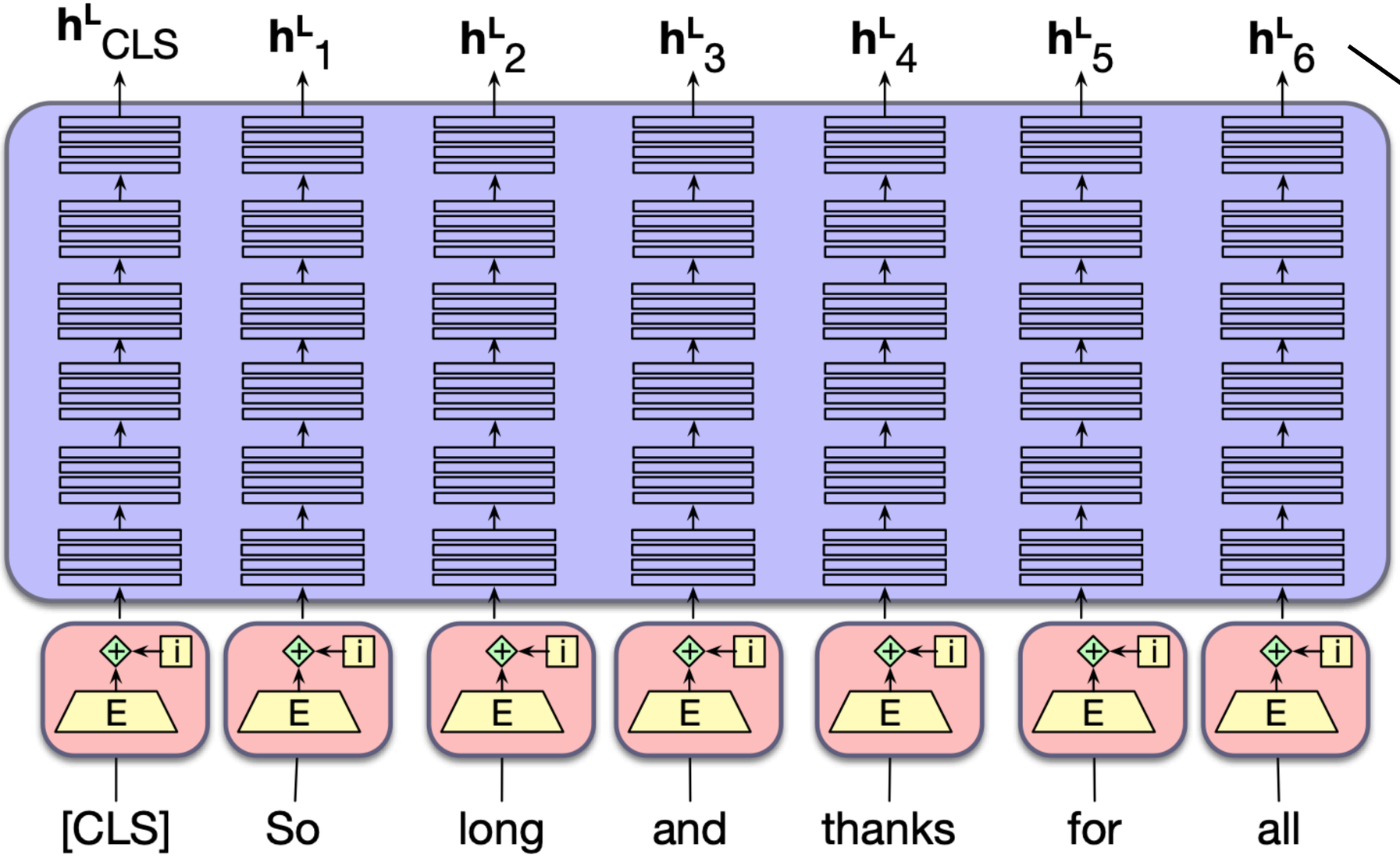
We can also do sequence labeling (classifying every token in the sequence) by feeding each token representation through a classifier.

$$\mathbf{y}_i = \text{softmax}(\mathbf{h}_i^L W_K)$$

The task shown here is **named entity recognition**: classify whether the token is part of a named entity (similar to a proper noun).



Contextual Word Embeddings



The token representations in the final layer(s) encode some aspect of the token's meaning in context.

Word Senses

- Recall a recurring problem we've been discussing: *the same word can have multiple meanings*. I.e., words can be **polysemous.**
- Usually, *context helps us figure out which sense is meant:*

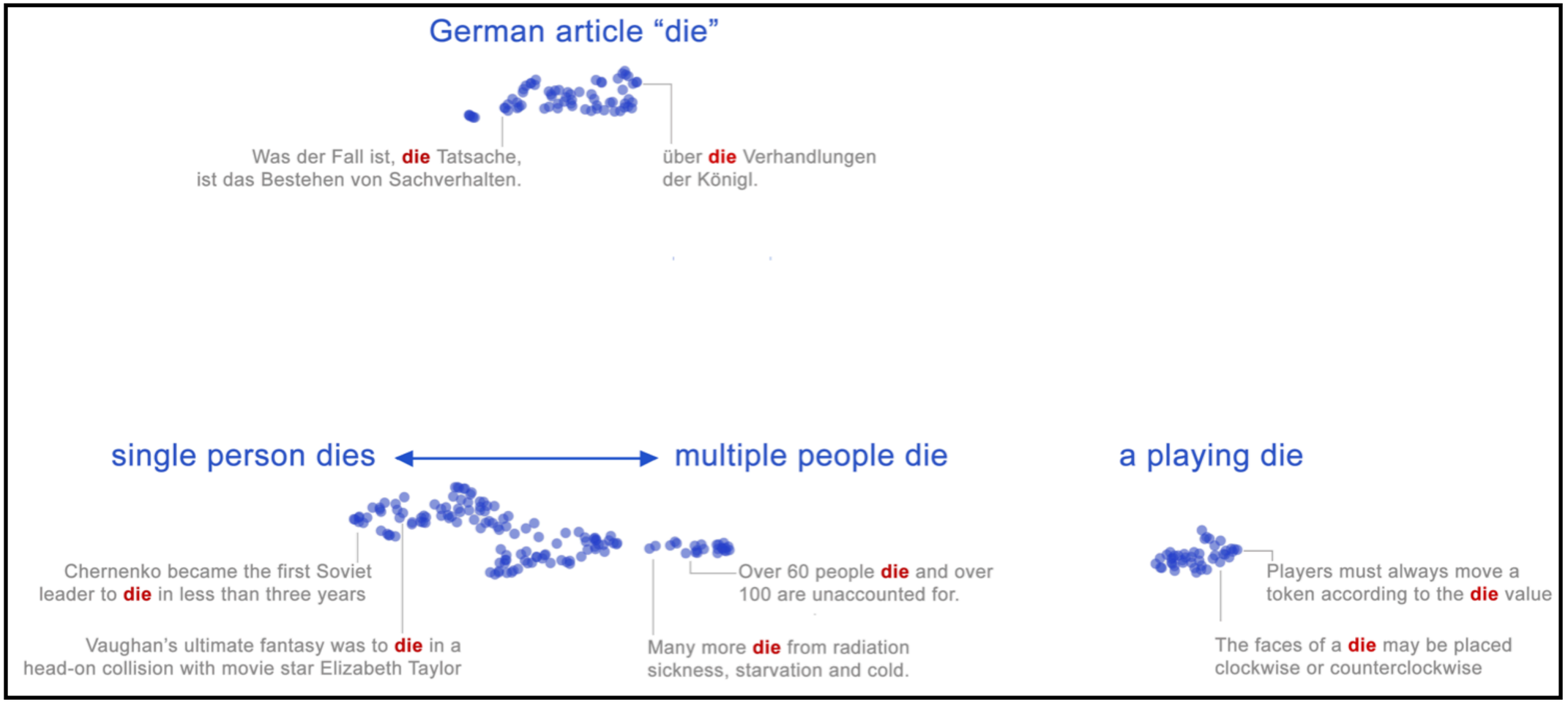
park[1]**:** a place in a town/city with greenery

park[2]**:** to leave one's car someplace

record[1]**:** a circular musical storage device

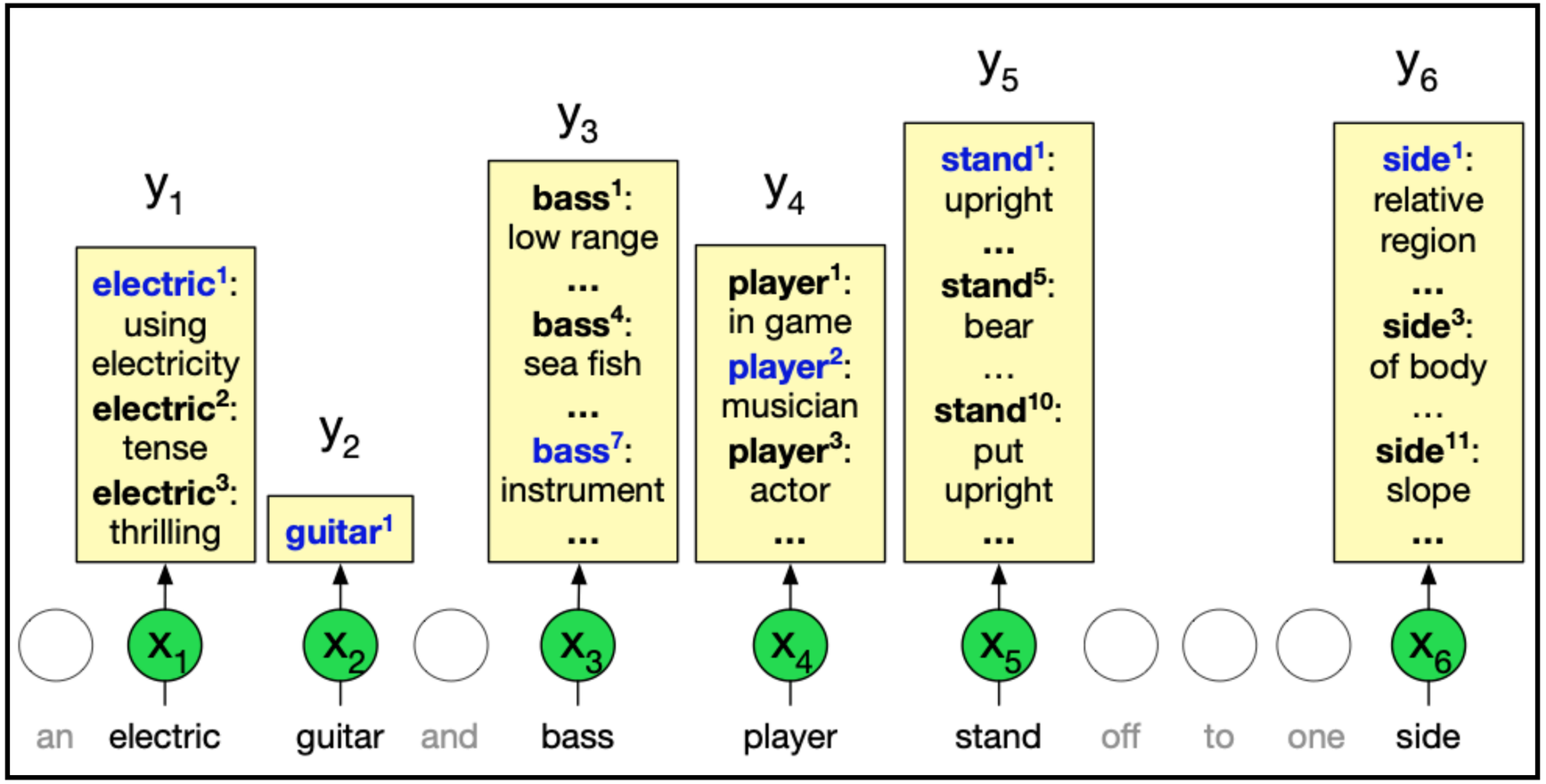
record[2]**:** to set down in some permanent form

Contextual Word Embeddings



Contextual embeddings of “die” across contexts, projected into two dimensions using UMAP.

Word Sense Disambiguation



Figuring out which sense of a word is meant is trivial for humans, but surprisingly hard for a neural network!

A decent heuristic: *one sense per discourse* **[Gale et al., 1992]**.

Word Sense Disambiguation

The state-of-the-art word sense disambiguation (WSD) method is based on contextual embeddings:

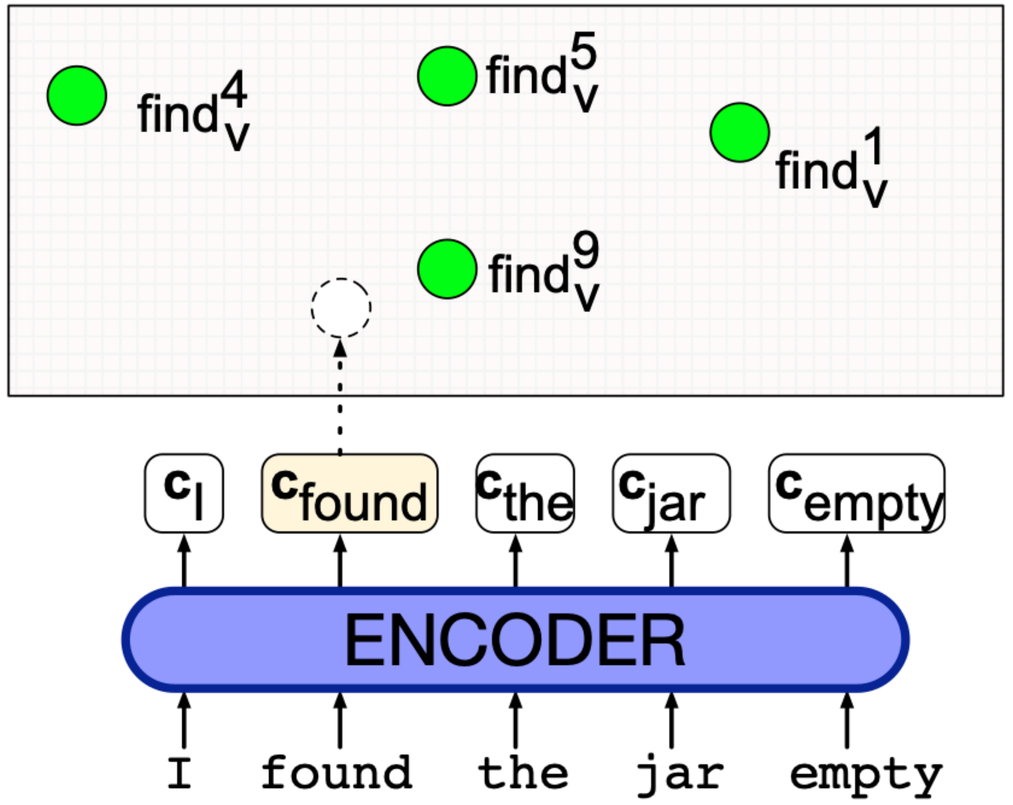
1. To train, pass sentences from a sense-labeled dataset through a contextual embedding model.

   Use embeddings $\mathbf{v}_i$ to compute one sense embedding per sense:

$$\bar{\mathbf{v}}_s = \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i$$

2. During testing, compare a word's contextual embedding with each sense embedding, and take the max to classify:

$$\text{sense}(t) = \arg\max_{s \in senses(t)} \cos(\mathbf{t}, \bar{\mathbf{v}}_s)$$



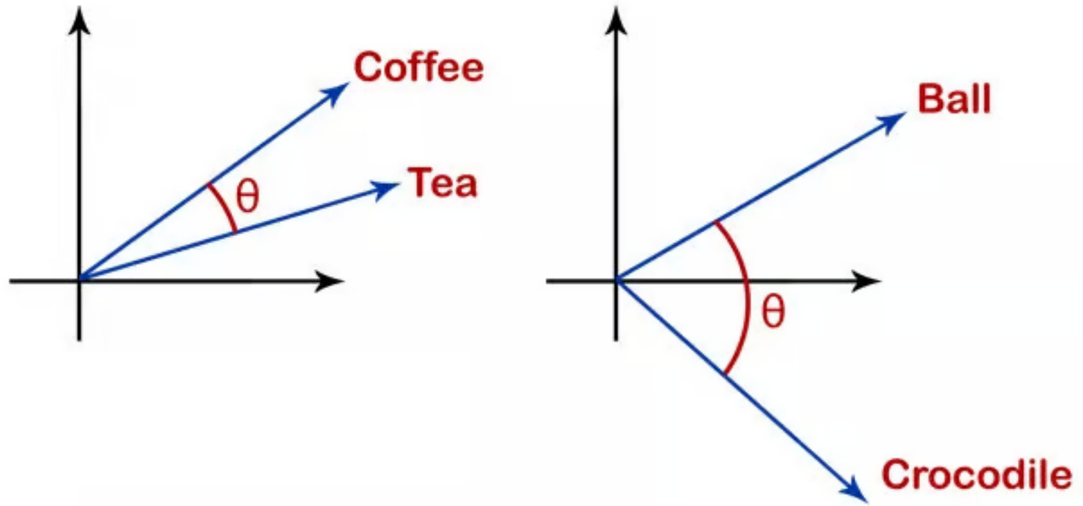
Comparing Word Similarities

Cosine Similarity

How do we compare the similarity of two embedding vectors?

By far the most common similarity metric is the **cosine** similarity:

$$\text{sim}(\mathbf{a}, \mathbf{b}) = \cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$



Intuition: if the angle between two vectors is low (i.e., if they point in similar directions), those vectors are similar to each other.

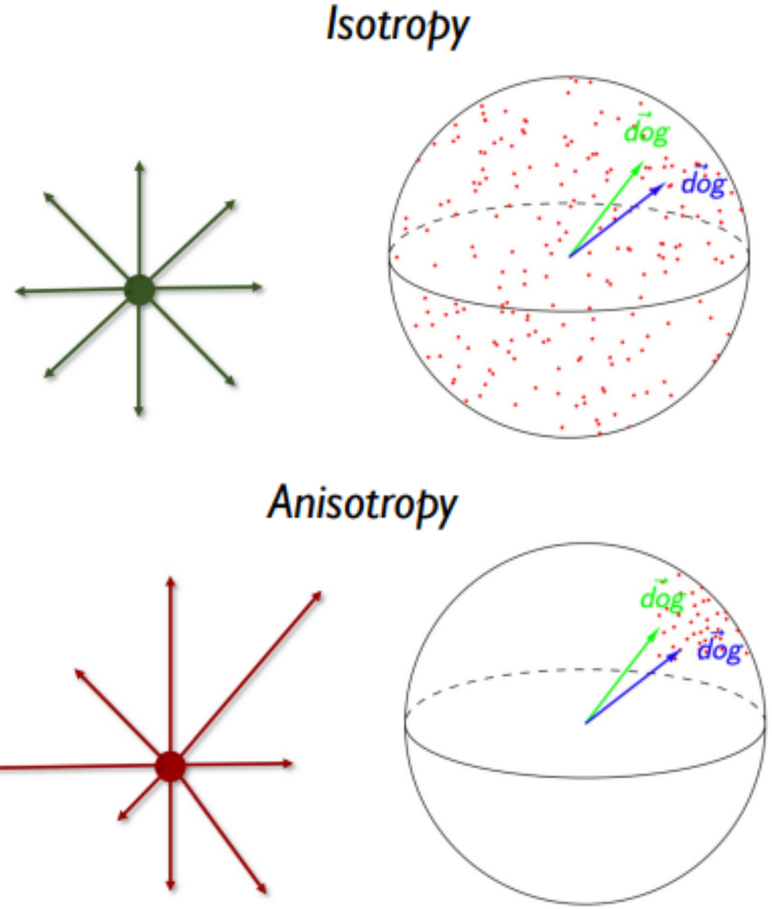
Issues in Using Contextual Embeddings

Anisotropy

Word embeddings are often not perfectly randomly distributed throughout the embedding space!

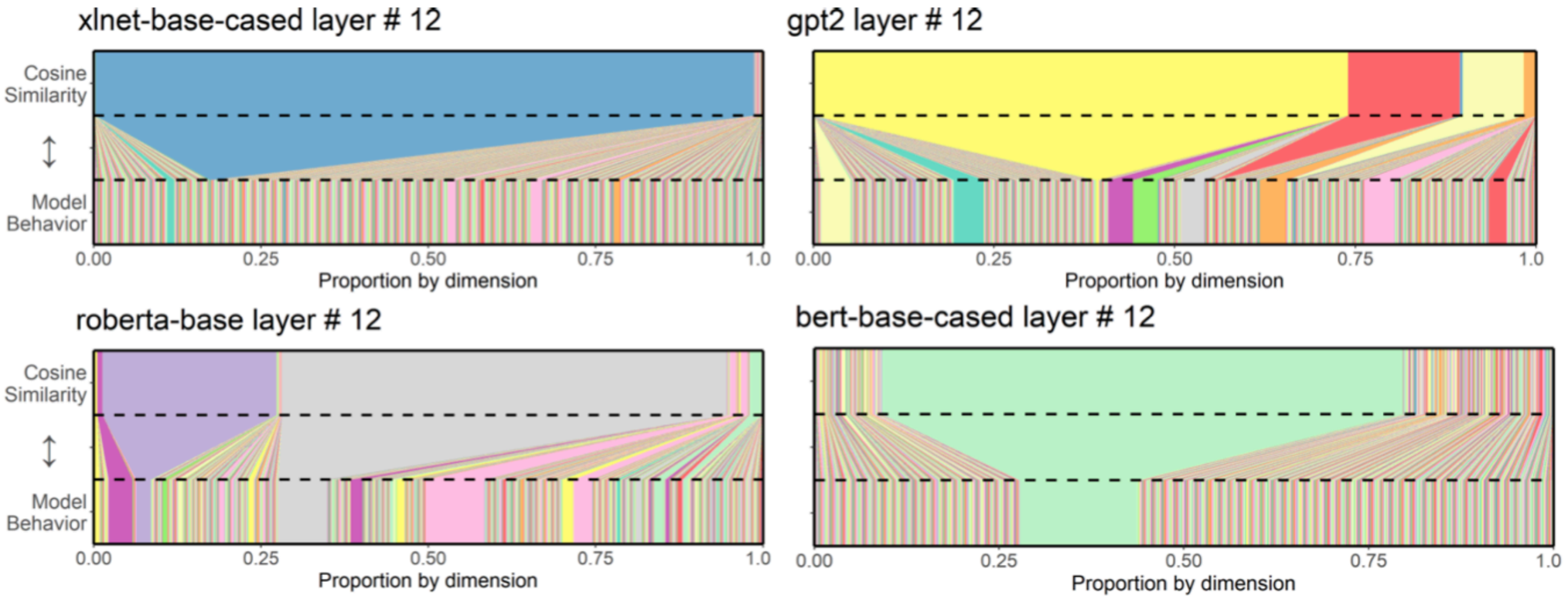
> **Anisotropy:** vectors in a system all tend toward the same directions.
>
> Expected cosine similarity of two words will be closer to 1 than expected.

Ideally, we would want **isotropy**, where we have uniformity of points in all directions.



Issues in Using Contextual Embeddings

Rogue Dimensions

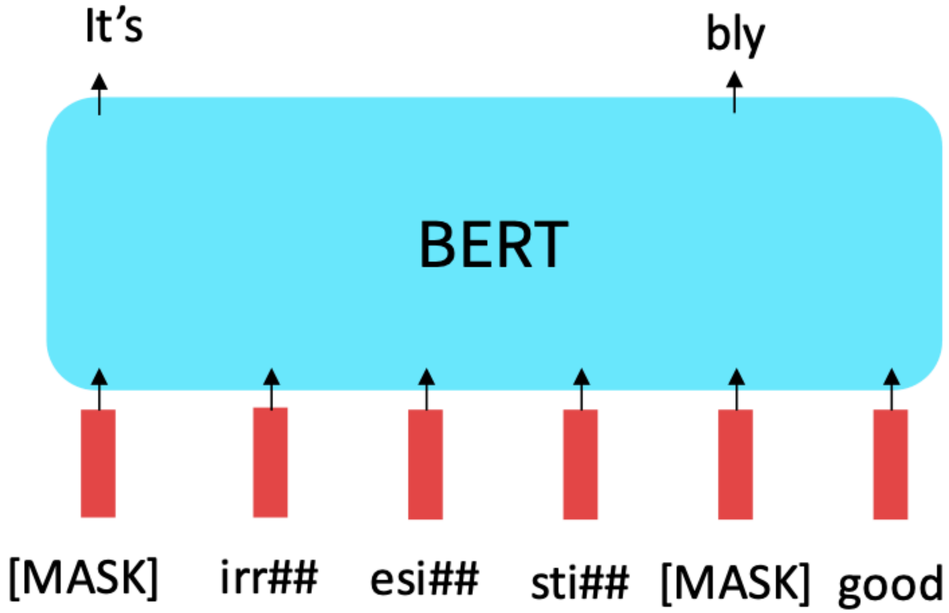


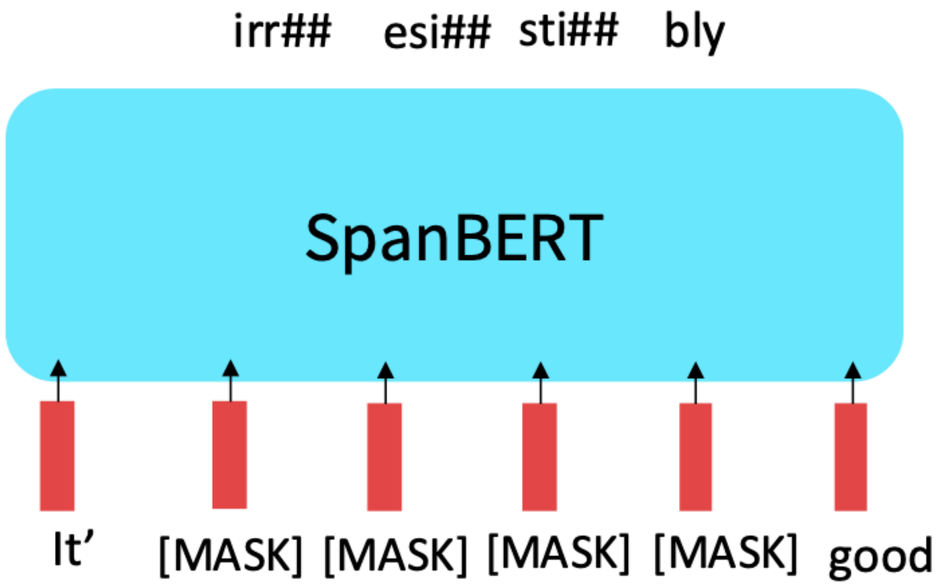
A small number of “rogue dimensions” with very large magnitudes and very high variance often dominate the cosine similarity.

We can fix this by standardizing (z-scoring) each dimension.

Extensions of BERT

- There are many improved versions of BERT: RoBERTa, SpanBERT, and many others
  - RoBERTa: just train BERT on more data and remove the NSP objective
  - SpanBERT: mask *spans of* tokens, instead of individual tokens





Extensions of BERT

The RoBERTa paper showed us that more compute and more data can improve pre-training, even when we don't change the underlying architecture.

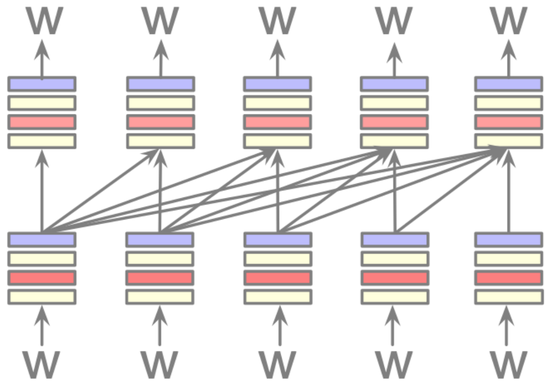
Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

What Can't BERT Do?

- BERT cannot (easily) generate text
  - You can predict a [MASK] token, but can't generate left-to-right
  - (You could technically put [MASK] at the end over and over, but this would be very slow)

- Masked language models/encoders are primarily meant for classification and analysis tasks

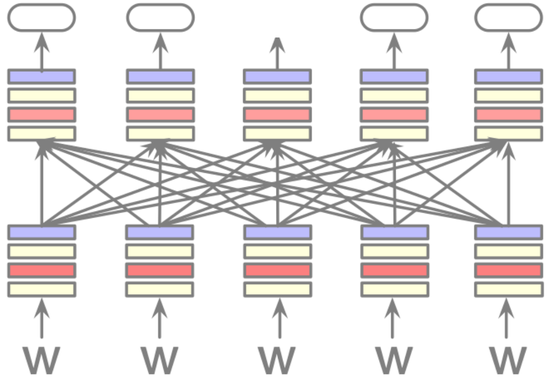
Architectures



Decoder

Left-to-right: generate the next word given prior context. Language models!

Good for generation.

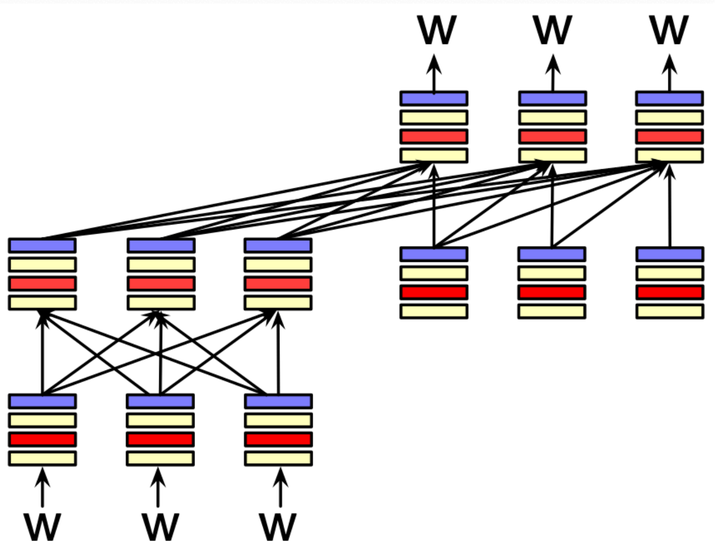


Encoder

Bidirectional: predict a word given left *and* right context.

Can only generate 1 token.

Good for classification.



Encoder-Decoder

Best of both worlds, maybe?

Used to be more common; not so much anymore.

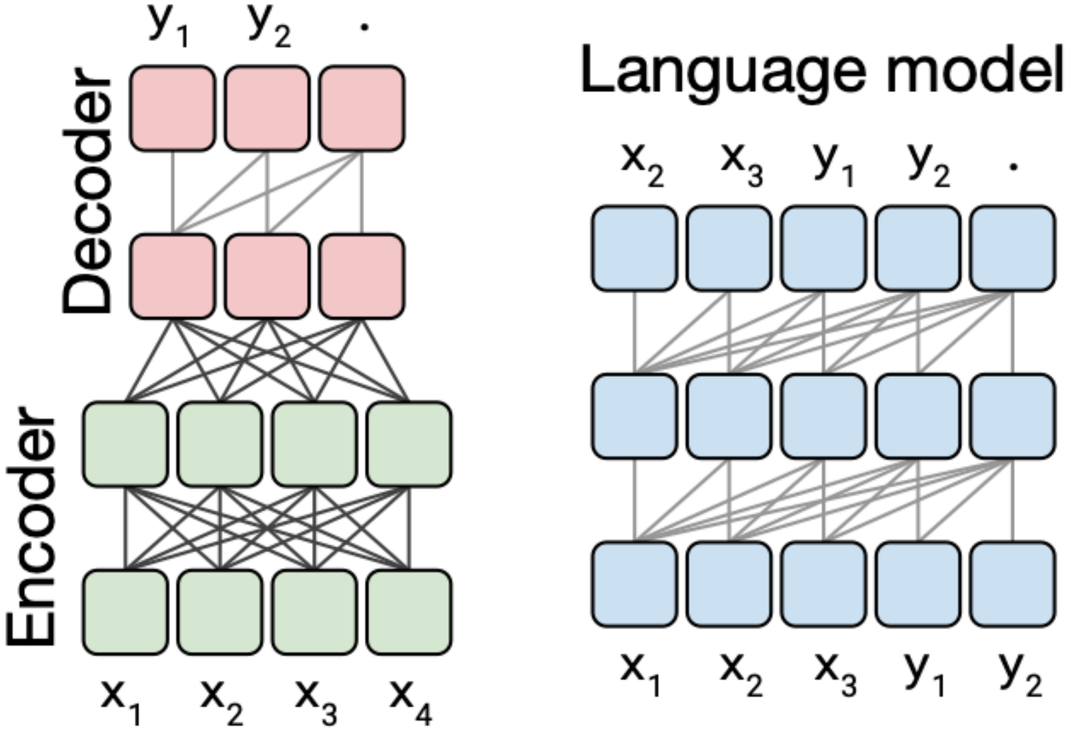
Encoder-Decoder Models

The encoder-decoder model first uses an **encoder**, which benefits fom bidirectional context.

The **decoder** portion is used to train the whole model (including the encoder) via a language modeling loss.

$\mathbf{x}$ = The cat sat on the mat.

$\mathbf{y}$ = Le chat s'est assis sur le tapis.



Encoder-Decoder Models

Training Objectives

Usually, we use **span corruption** to train encoder-decoder models.

Replace spans of varying lengths with unique masks:

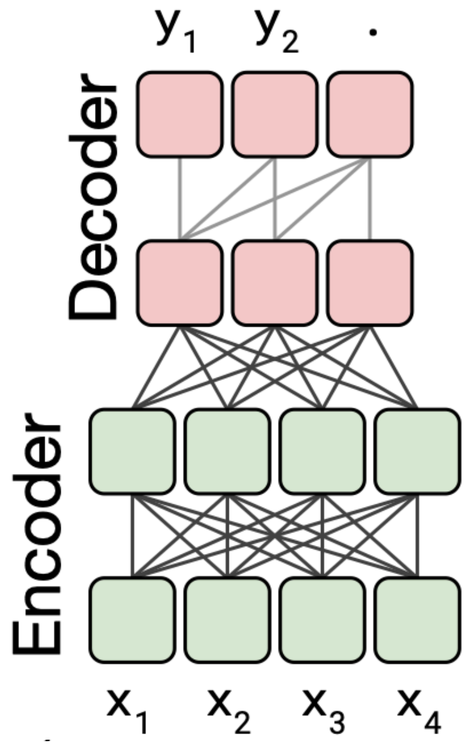
Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Then, train the model to reconstruct the masked spans. Note: the objective still looks like language modeling for the decoder.

Targets

<X> for inviting <Y> last <Z>

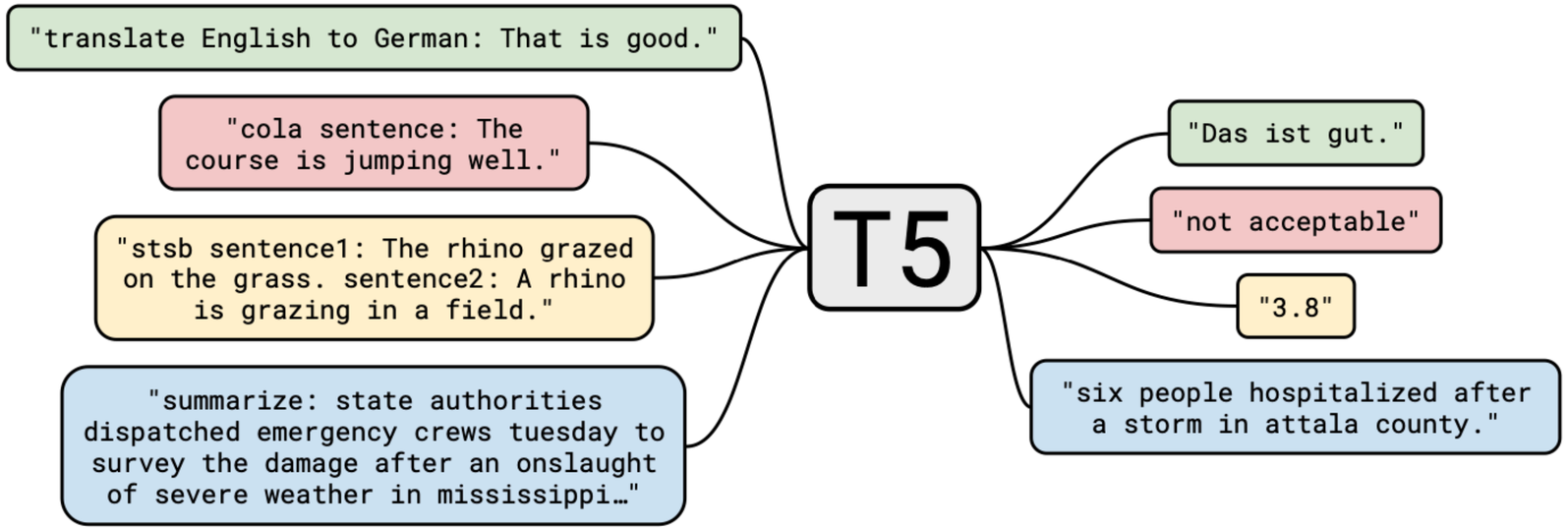


Inputs

Thank you <X> me to your party <Y> week.

T5: Text-**t**o-**T**ext **T**ransfer **T**ransformer

- **T5** was among the most popular encoder-decoder models.
  - 12-layer encoder and 12-layer decoder
  - Mask 15% of tokens, average mask span of 3
  - Trained on 34 **b**illion tokens of web text (C4)

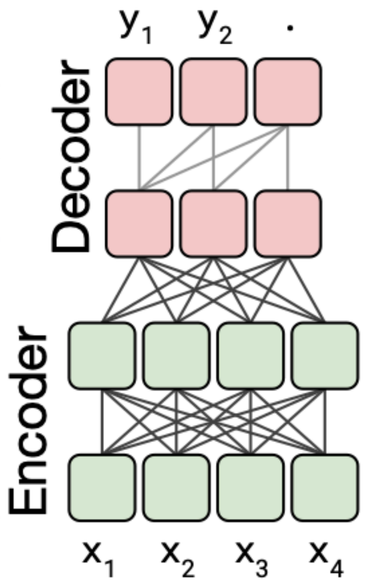


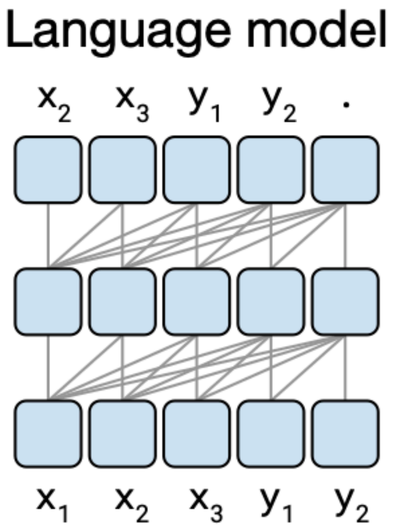
Comparison of Objectives

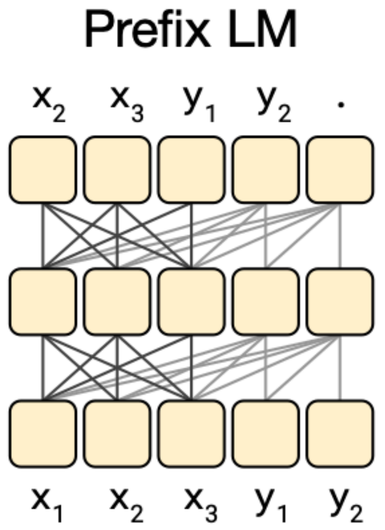
Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

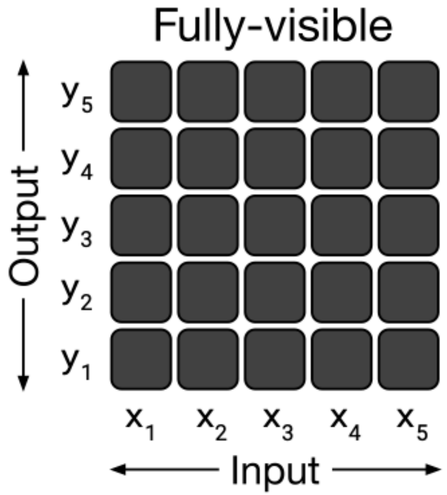
Comparison of Objectives

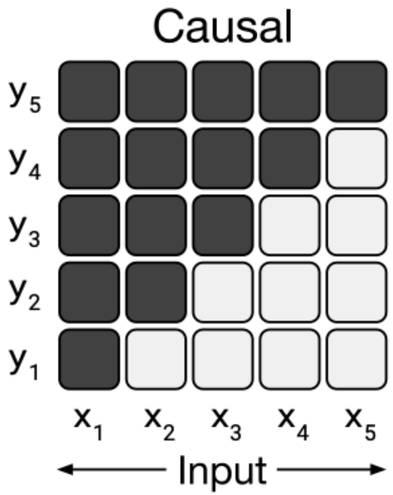
Objective	
Prefix language modeling	ırty last week .
BERT-style Devlin et al. (201	)
Deshuffling	)
MASS-style Song et al. (2019)	)
I.i.d. noise, replace spans	ıg <Y> last <Z>
I.i.d. noise, drop tokens	st
Random spans	ıg me <Y> your party last <Z>

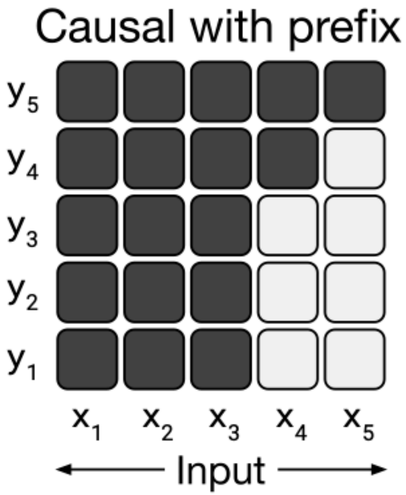












SQuAD

Extractive Question Answering

Extractive QA: you must *extract* the answer from a context passage

Q: In what R&B group did Beyonce become well-known in the 1990s?

Context: Beyoncé Knowles-Carter is an American singer, songwriter, record producer and actress. Born and raised in Houston, TX, she performed in various singing and dancing competitions as a child, and rose to fame in the 1990s as lead singer of R&B girl-group Destiny's Child. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and features the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".

Answer: Destiny's Child

SQuAD

Extractive Question Answering

Extractive QA: you must *extract* the answer from a context passage

Q: In what R&B group did Beyonce become well-known in the 1990s?

Context: Beyoncé Knowles-Carter is an American singer, songwriter, record producer and actress. Born and raised in Houston, TX, she performed in various singing and dancing competitions as a child, and rose to fame in the 1990s as lead singer of R&B girl-group Destiny's Child. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and features the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".

Can use BERT or T5 to predict (start, end) indices given Q and C:

$p(\text{start} \mid q, c) =$ 0.01 0.01 0.01 0.82 0.01

R&B girl group Destiny 's Child

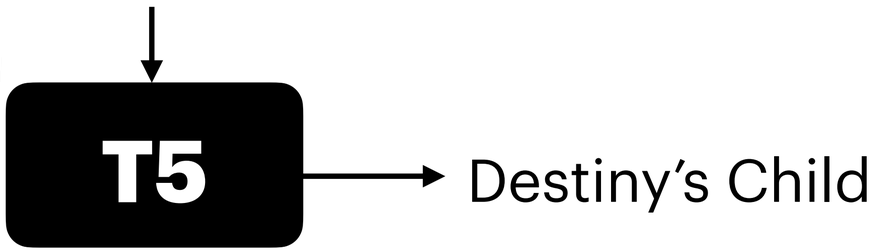
SQuAD

Extractive Question Answering

Extractive QA: you must *extract* the answer from a context passage

Q: In what R&B group did Beyonce become well-known in the 1990s?

Context: Beyoncé Knowles-Carter is an American singer, songwriter, record producer and actress. Born and raised in Houston, TX, she performed in various singing and dancing competitions as a child, and rose to fame in the 1990s as lead singer of R&B girl-group Destiny's Child. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and features the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".



T5 can just encode the question and context, and generate the answer.

Which objective works best?

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

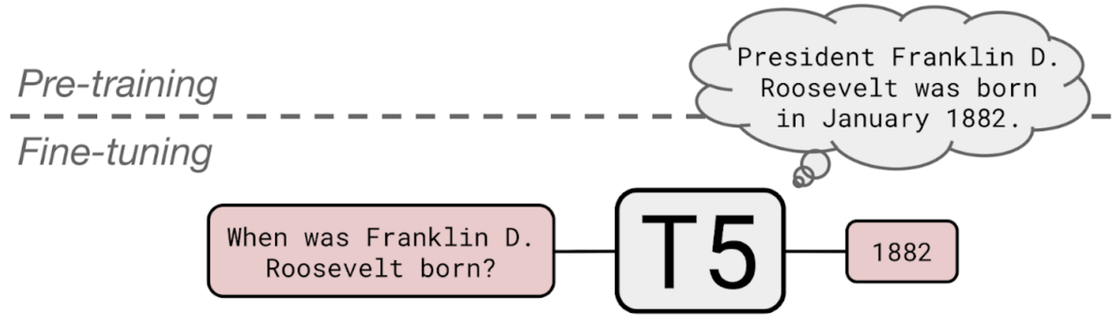
State-of-the-art results (for its time)!

Improved Question Answering

T5 was particularly good at *open-domain* question answering for its time.

Q: When was Franklin D. Roosevelt born?
A: 1882

After fine-tuning, it could answer a wide variety of questions by retrieving knowledge from its parameters.



	NQ	WQ	TQA	
			dev	test
Karpukhin et al. (2020)	**41.5**	42.4	**57.9**	–
T5.1.1-Base	25.7	28.2	24.2	30.6
T5.1.1-Large	27.3	29.5	28.5	37.2
T5.1.1-XL	29.5	32.4	36.0	45.1
T5.1.1-XXL	32.8	35.6	42.9	52.5
T5.1.1-XXL + SSM	35.2	**42.8**	51.9	**61.6**