COS 484

Natural Language Processing

L10: Contextualized Representations and Pre-training

Spring 2026

This lecture

- **Contextualized word embeddings**
- **Pre-training** and **fine-tuning**
- GPT, **ELMo**, **BERT**

Contextualized Word Embeddings

Limitations of word2vec

- One vector for each word type
  - (a.k.a. static embeddings)

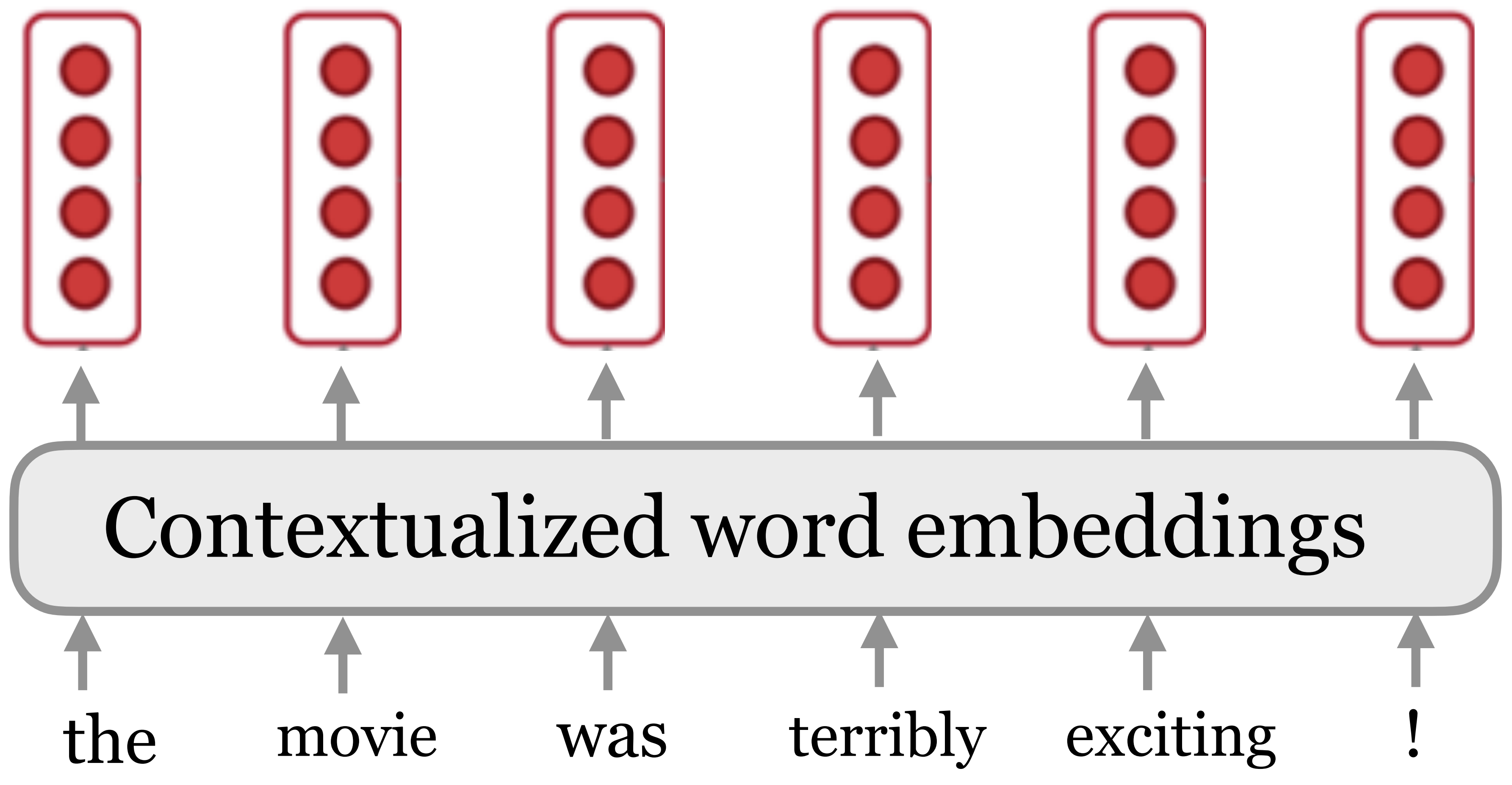
$$v(\text{play}) = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix}$$

- Complex characteristics of word use: syntax and semantics
- Polysemous words (e.g., mouse, bank)

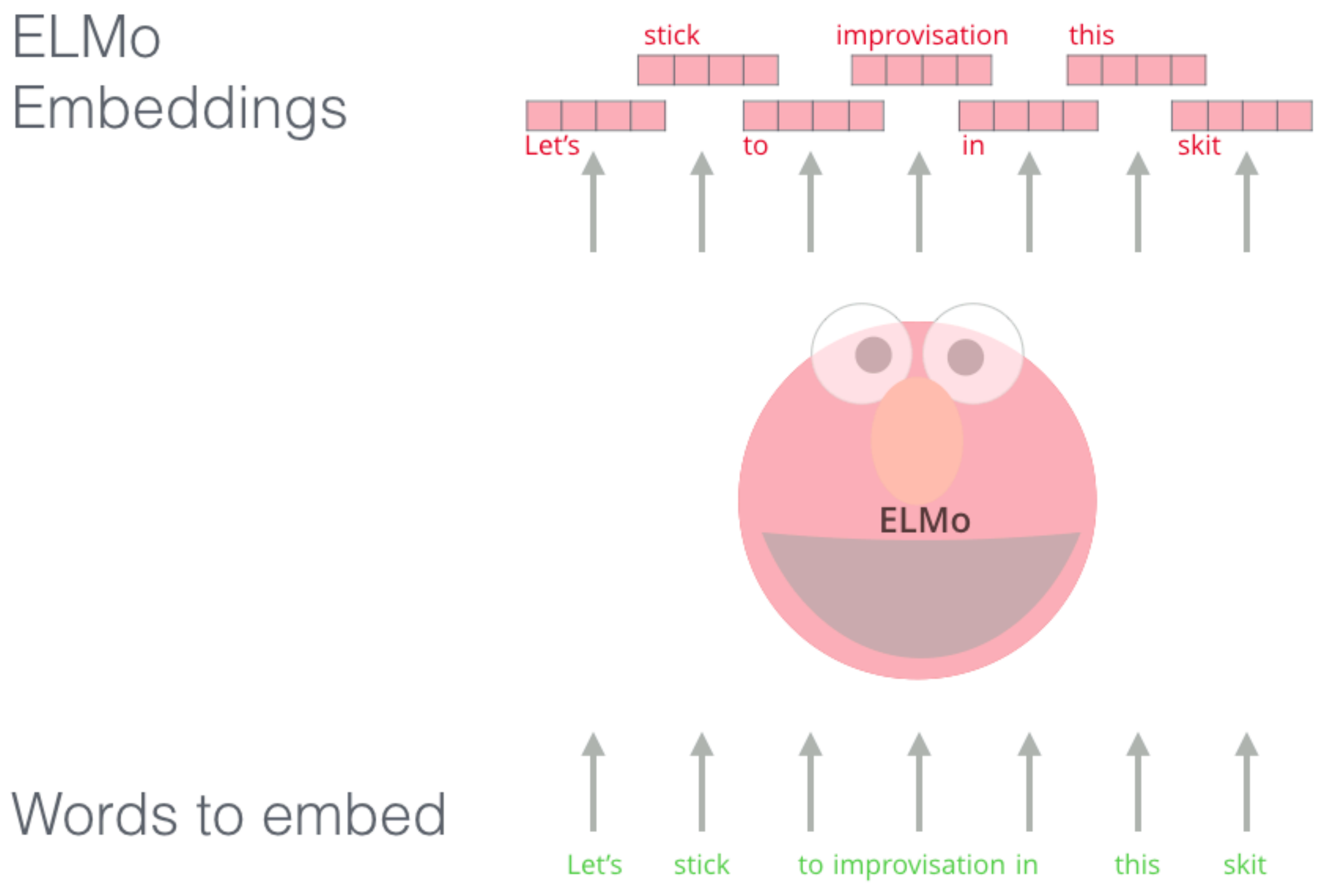
mouse^1 : .... a *mouse* controlling a computer system in 1968.
mouse^2 : .... a quiet animal like a *mouse*
bank^1 : ...a *bank* can hold the investments in a custodial account ...
bank^2 : ...as agriculture burgeons on the east *bank*, the river ...

Contextualized word embeddings

Let's build a vector for each word conditioned on its **context**!



$$f : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$



http://jalammar.github.io/illustrated-bert/

Contextualized word embeddings

Hey ELMo, what's the embedding of the word "stick"?
There are multiple possible embeddings! Use it in a sentence.
Oh, okay. Here: "Let's stick to improvisation in this skit"
Oh in that case, the embedding is: -0.02, -0.16, 0.12, -0.1 ….etc

http://jalammar.github.io/illustrated-bert/

Contextualized word embeddings

Sent #1: Chico Ruiz made a spectacular **play** on Alusik's grounder {. . . }	$v(\textbf{play}) = ?$
Sent #2: Olivia De Havilland signed to do a Broadway **play** for Garson {. . . }	$v(\textbf{play}) = ?$
Sent #3: Kieffer was commended for his ability to hit in the clutch , as well as his all-round excellent **play** {. . . }	$v(\textbf{play}) = ?$
Sent #4: {. . . } they were actors who had been handed fat roles in a successful **play** {. . . }	$v(\textbf{play}) = ?$
Sent #5: Concepts **play** an important role in all aspects of cognition {. . . }	$v(\textbf{play}) = ?$

Contextualized word embeddings

Sent #1: Chico Ruiz made a spectacular **play** on Alusik's grounder {. . . }

Which of the following $v(\text{play})$ is expected to have the most similar vector to the first one?

(A) Olivia De Havilland signed to do a Broadway **play** for Garson {. . . }

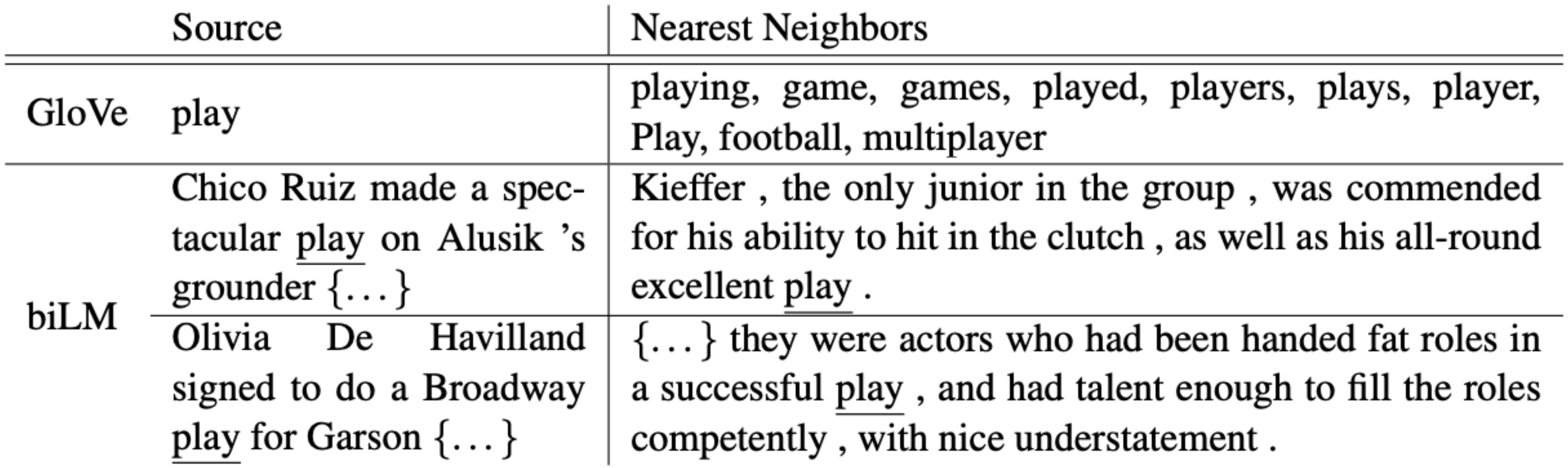
(B) Kieffer was commended for his ability to hit in the clutch , as well as his all-round excellent **play** {. . . }

(C) {. . . } they were actors who had been handed fat roles in a successful **play** {. . . }

(D) Concepts **play** an important role in all aspects of cognition {. . . }

(B) is correct.

Contextualized word embeddings



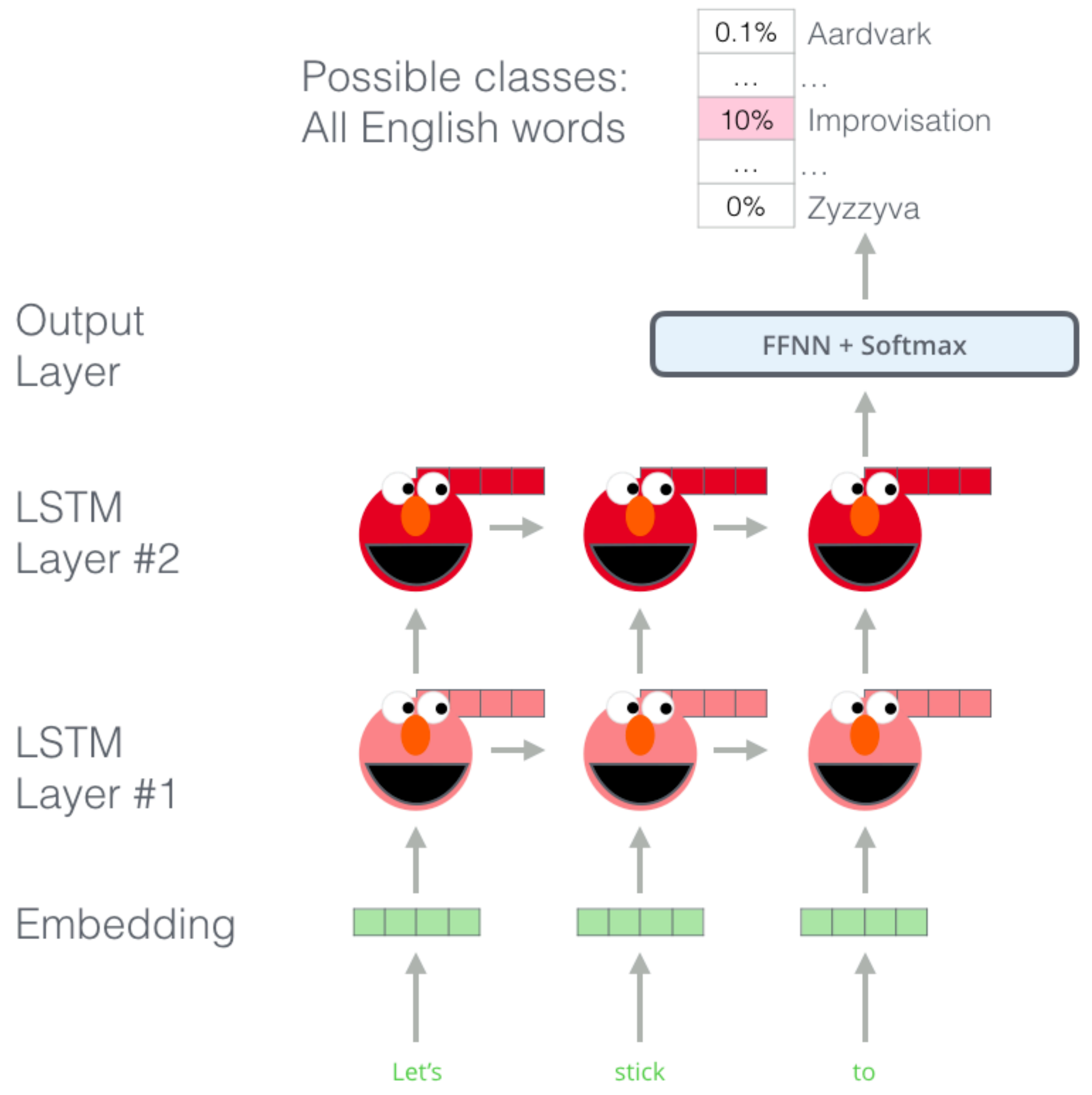
	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {. . . }	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {. . . }	{. . . } they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

(Peters et al, 2018): Deep contextualized word representations

ELMo: Embeddings from Language Models (Released in 2018/2)

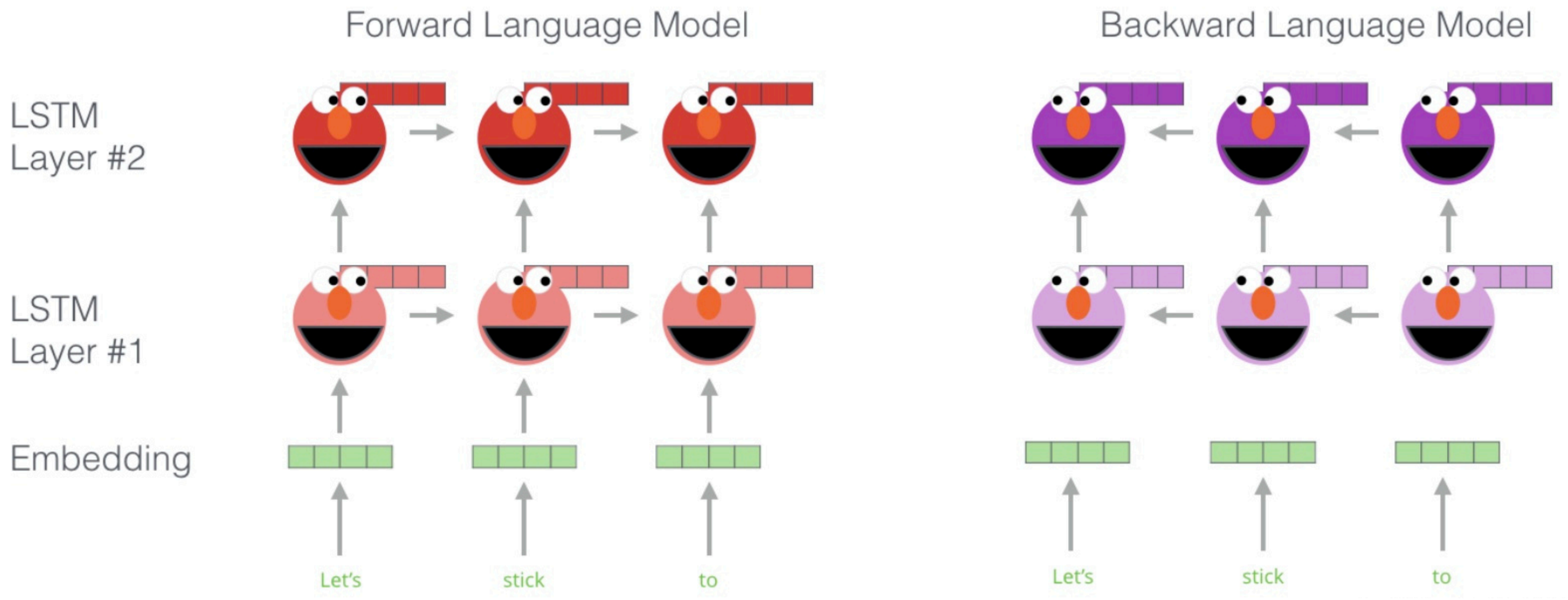
The key idea of ELMo:

- Train *two* stacked LSTM-based language models on a **large** corpus
- Use the **hidden states** of the LSTMs for each token to compute a vector representation of each word



http://jalammar.github.io/illustrated-bert/

How does ELMo work?



$$\boxed{\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}}$$

Contextualized word embeddings =
The weighted average of input embeddings + all hidden representations

The weights γ^{task}, s_j^{task} are task-dependent and learned

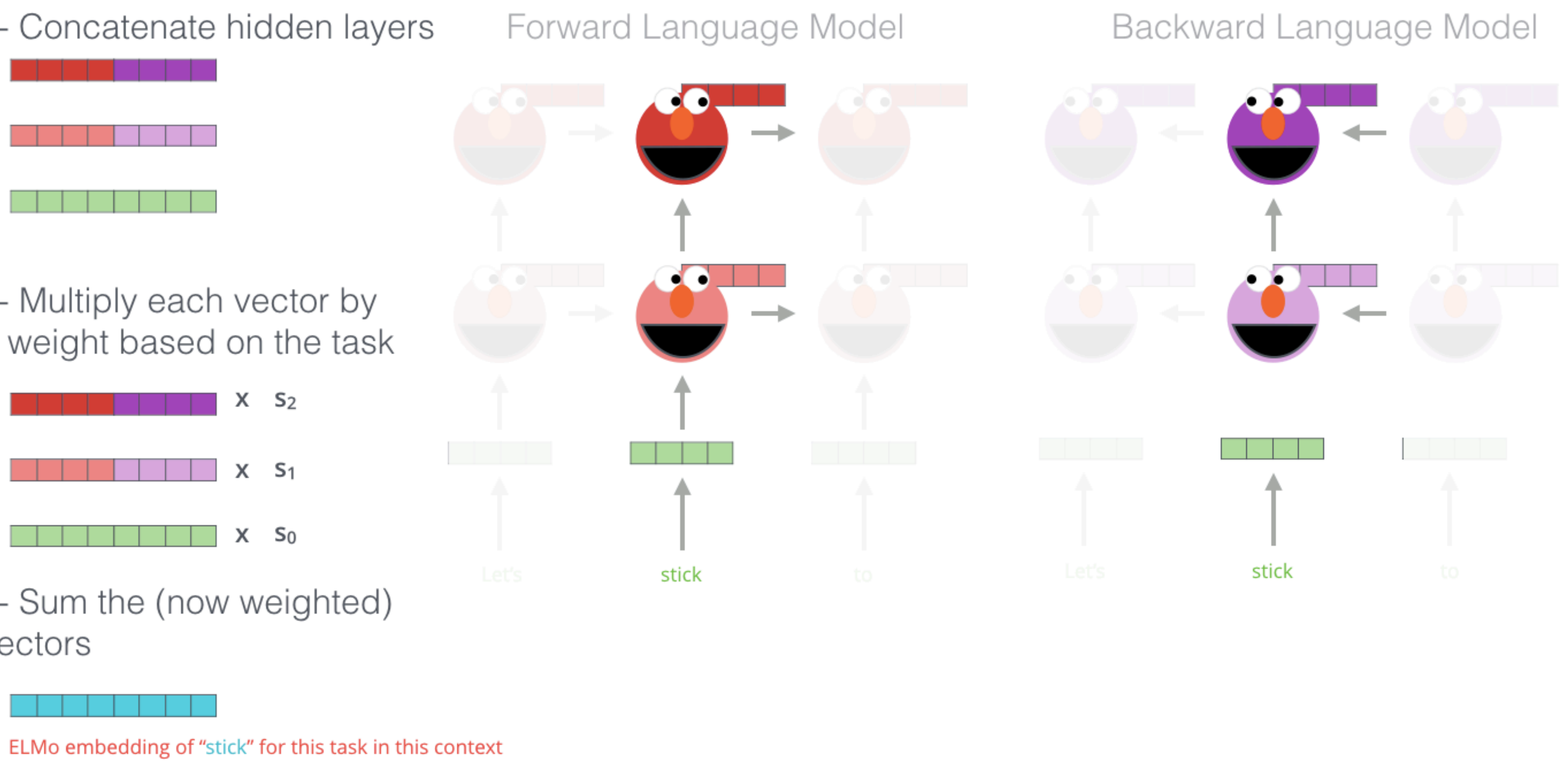
http://jalammar.github.io/illustrated-bert/

How does ELMo work?

1- Concatenate hidden layers

2- Multiply each vector by a weight based on the task

3- Sum the (now weighted) vectors

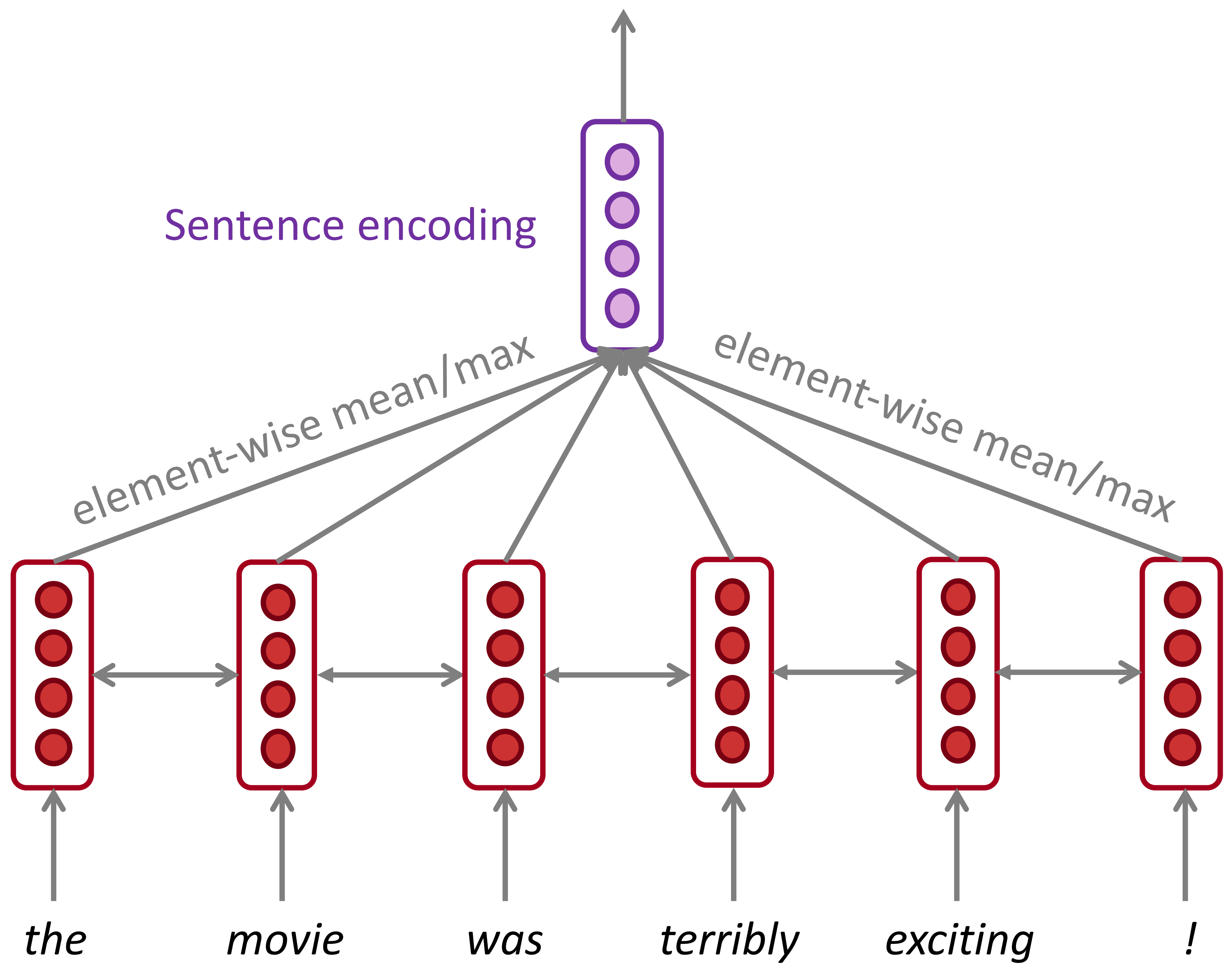


http://jalammar.github.io/illustrated-bert/

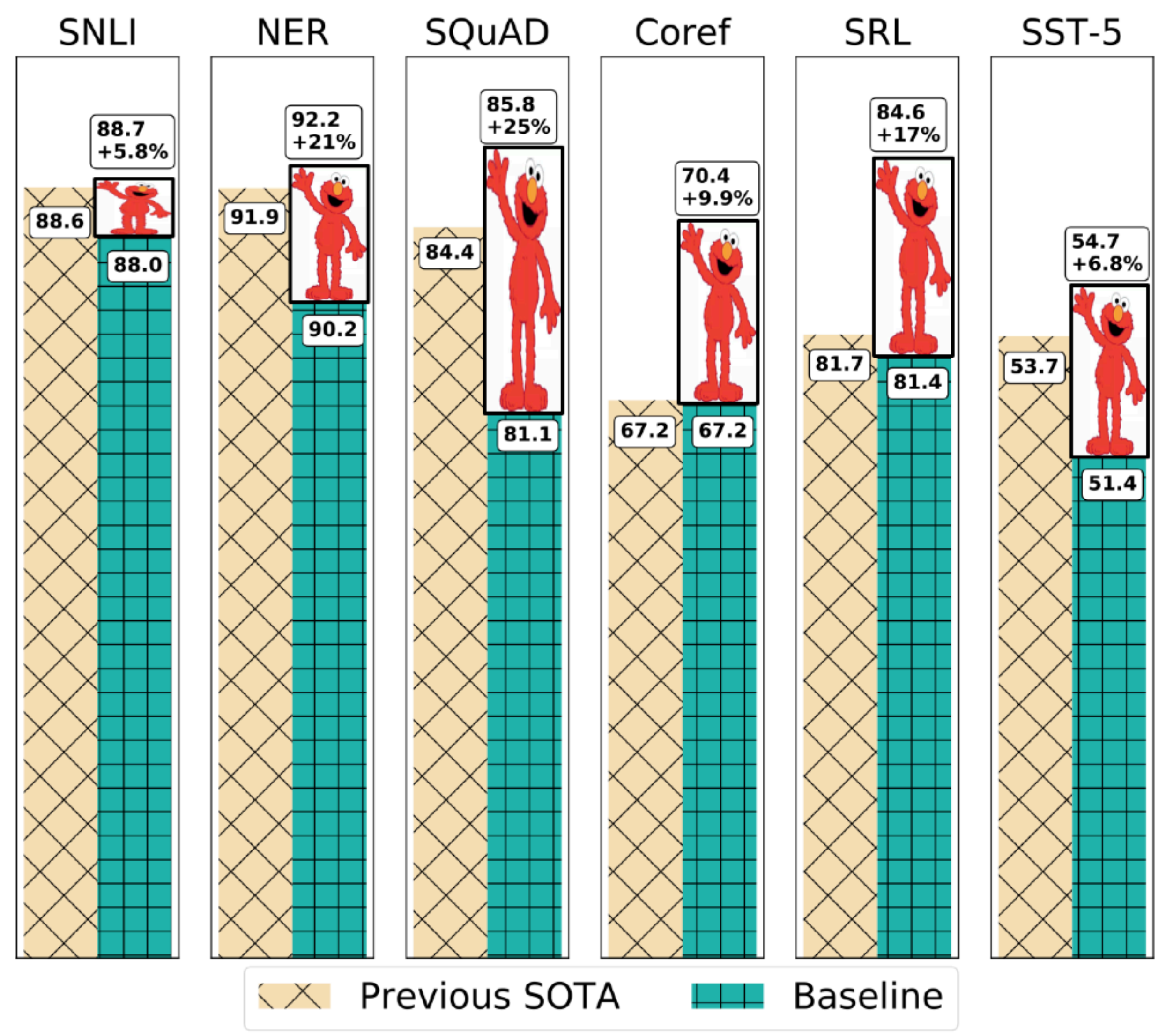
ELMo: pre-training and the use

- Data: 10 epochs on 1B Word Benchmark (trained on **single sentences**)
- Training time: 2 weeks on 3 NVIDIA GTX 1080 GPUs

Example use: A BiLSTM model for sentiment classification



$$[e(\text{terribly}); \text{ELMo}_4^{\text{task}}]$$



(Peters et al, 2018): Deep contextualized word representations

ELMo: some take-aways

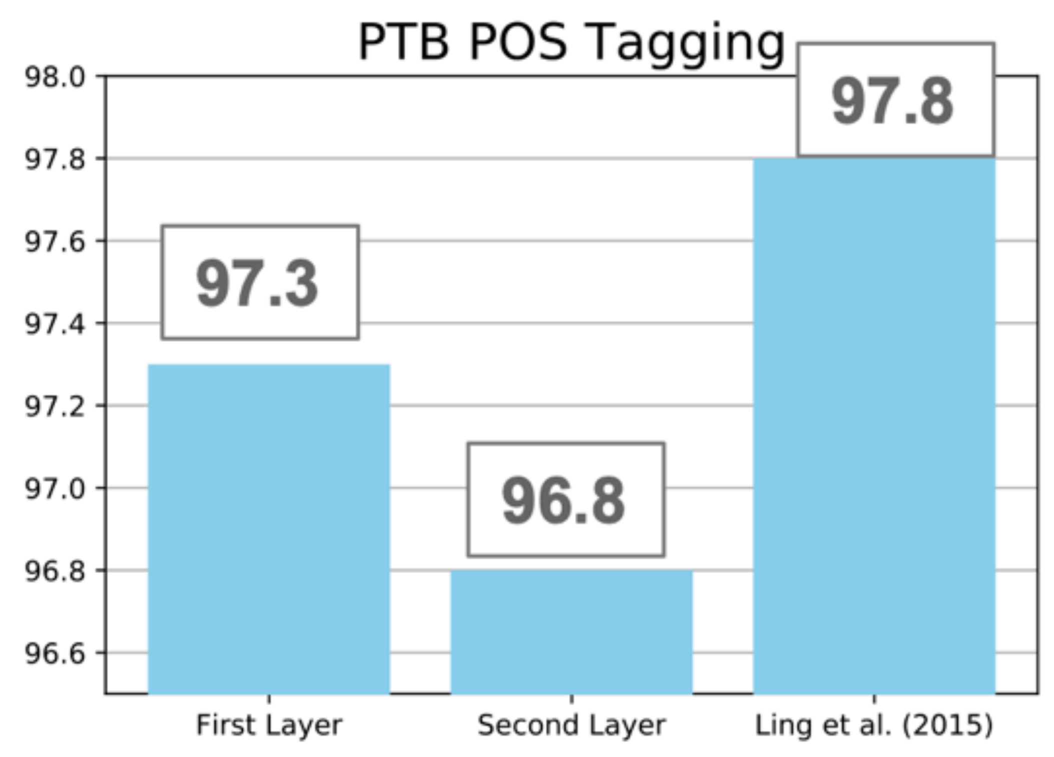
Q: Why use both forward and backward language models?

Because it is important to model both left and right context!

Bidirectionality is very important in language understanding tasks!

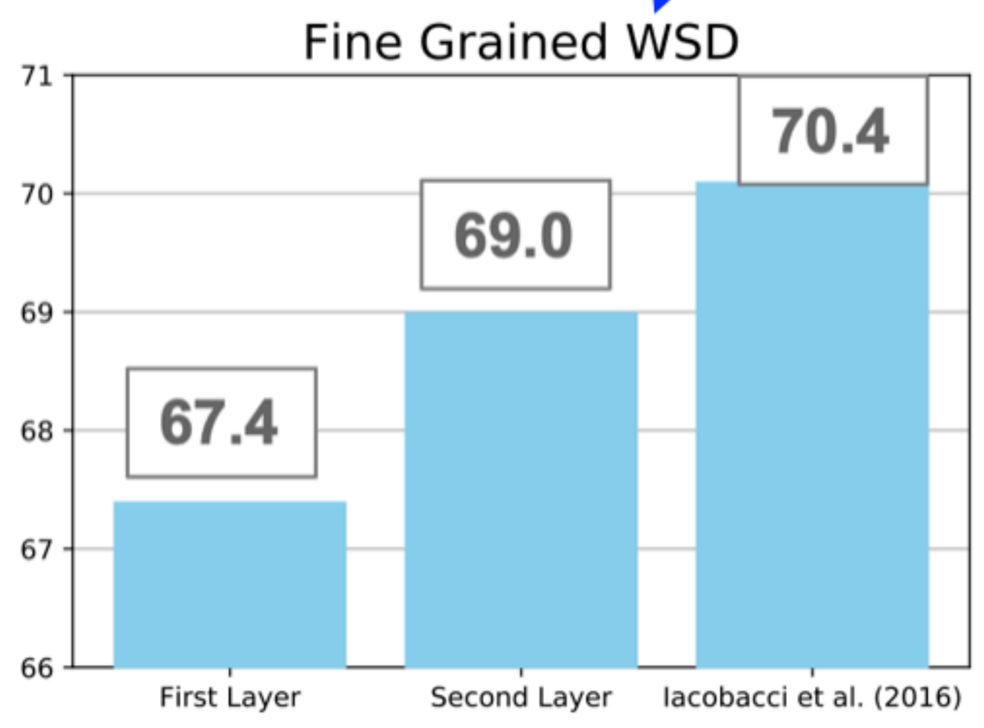
Q: Why use the weighted average of different layers instead of just the top layer?

Because different layers are expected to encode different information.



first layer > second layer

WSD = word sense disambiguation



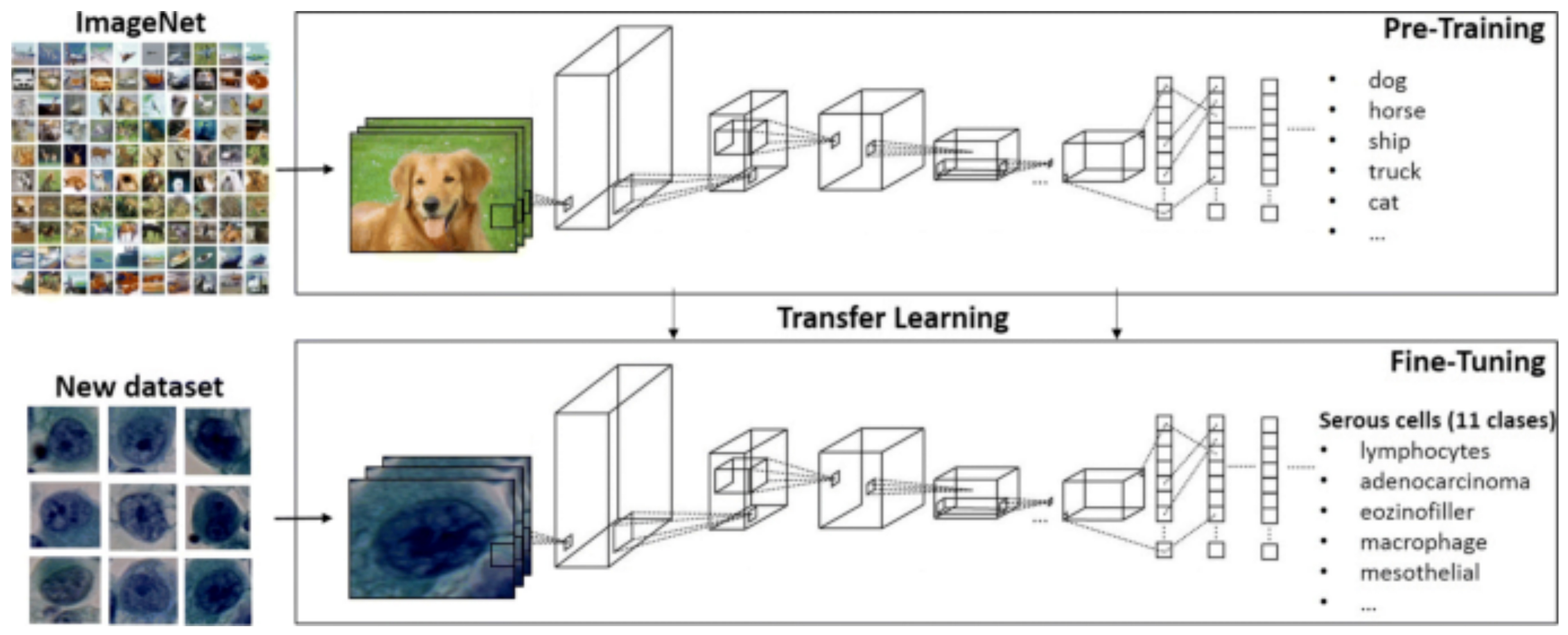
second layer > first layer

Pre-training and Fine-tuning

What is pre-training / fine-tuning?

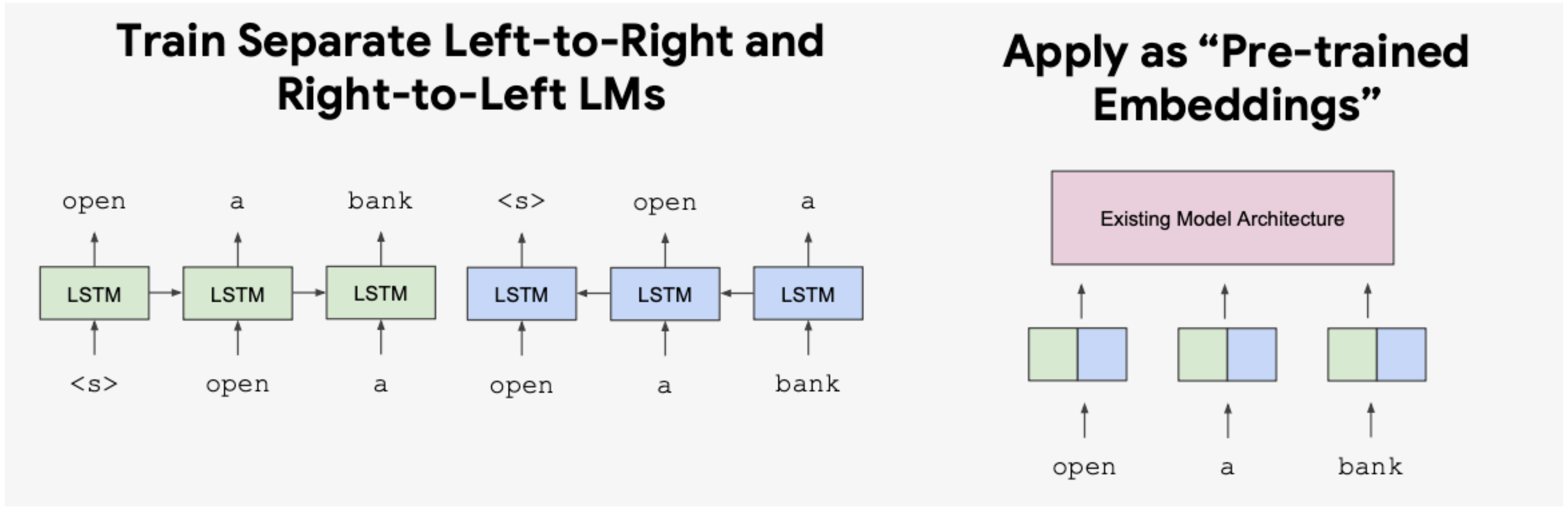
- “Pre-train” a model on a large dataset for task X, then “fine-tune” it on a dataset for task Y
- Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well
- ImageNet pre-training is huge in computer vision: learning generic visual features for recognizing objects

Can we find some task X that can be useful for a wide range of downstream tasks Y?



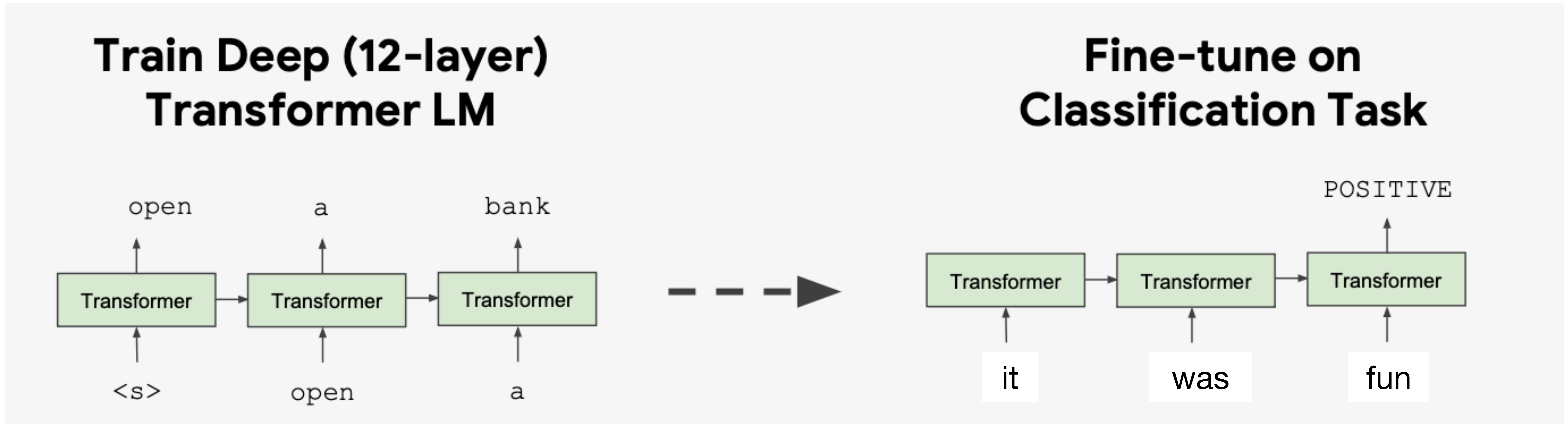
Feature-based vs fine-tuning approaches

- ELMo is a feature-based approach which only produces word embeddings that can be used as **input representations** of existing neural models


Train Separate Left-to-Right and Right-to-Left LMs
open
a
bank
<s>
open
a
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
<s>
open
a
open
a
bank
Apply as "Pre-trained Embeddings"
Existing Model Architecture
open
a
bank

Feature-based vs fine-tuning approaches

- GPT / BERT (and most of following models) are **fine-tuning approaches**
    - Almost all model weights will be **re-used**, and only a small number of task-specific will be added for downstream tasks



Most of pre-training is reconstructing the input

- Princeton is located in ____________.

Slide credit: John Hewitt, Stanford CS224N

What can we learn from reconstructing the input?

- Princeton is located in ____________.
- I went to the ocean and saw fish, turtles, __________ and seals.
  - General semantics
- I put ____ fork down on the table
  - Syntactic constraints
- The woman walked across the street checking for traffic over _____ shoulder.
  - Co-reference, relations between different entities within the sentence
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ______.
  - Sentiment

Slide credit: John Hewitt, Stanford CS224N

Pre-training for three types of architectures

- The neural architecture influences the type of pre-training and natural use cases:
  - **Encoders**: Gets bidirectional context
  - **Encoder-decoders**: Gets good parts of encoders and decoders?
  - **Decoders**: Language models!

Slide credit: John Hewitt, Stanford CS224N

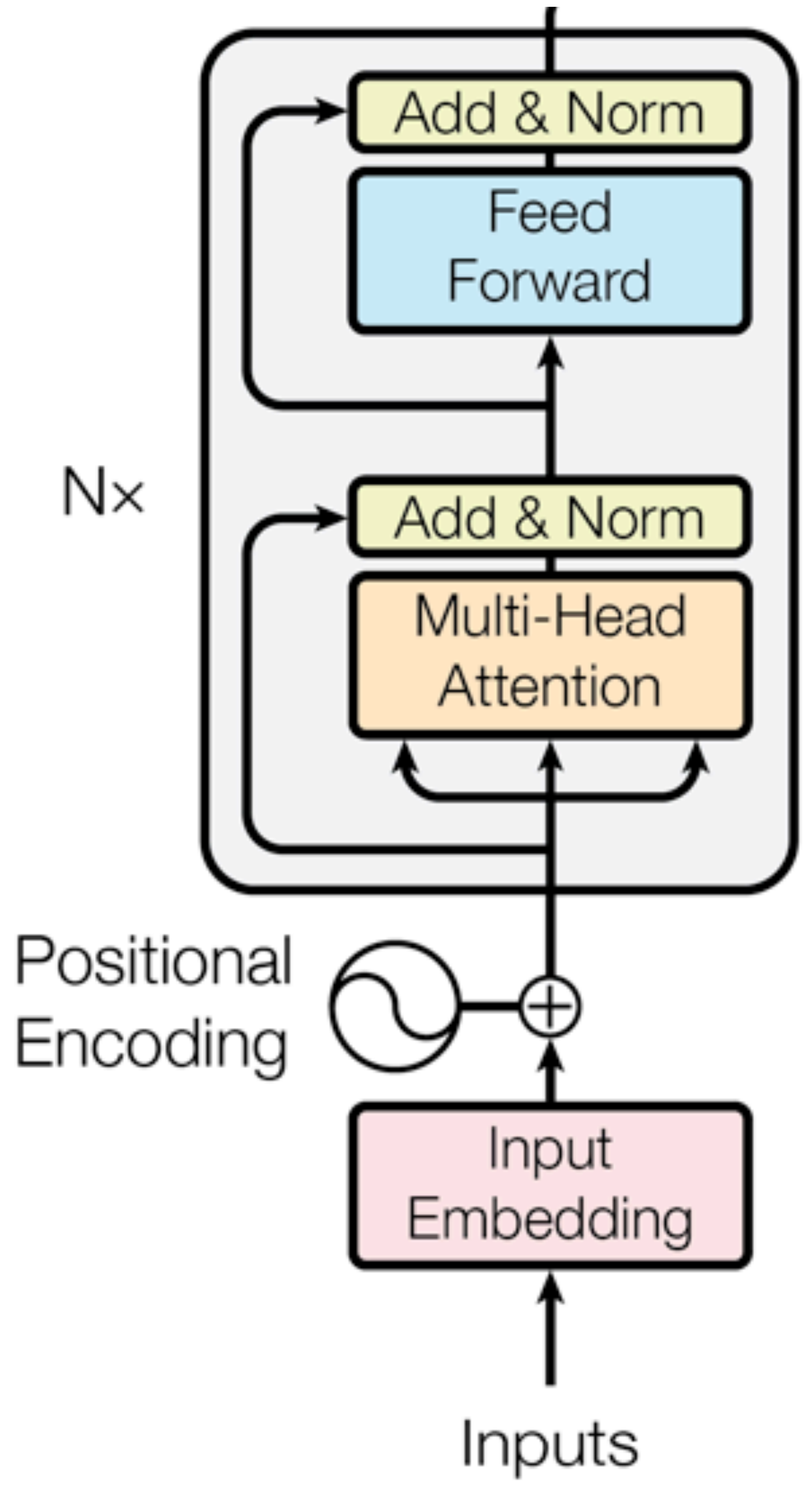
Pre-training for three types of architectures

- The neural architecture influences the type of pre-training and natural use cases:
  - **Encoders**: Gets bidirectional context
  - **Encoder-decoders**: Gets good parts of encoders and decoders?
  - **Decoders**: Language models!

Slide credit: John Hewitt, Stanford CS224N

BERT: Bidirectional Encoder Representations from Transformers

(Released in 2018/10)



- It is a fine-tuning approach based on a deep **bidirectional Transformer encoder** instead of a Transformer decoder
- The key: learn representations based on **bidirectional contexts**

Example #1: we went to the river bank.

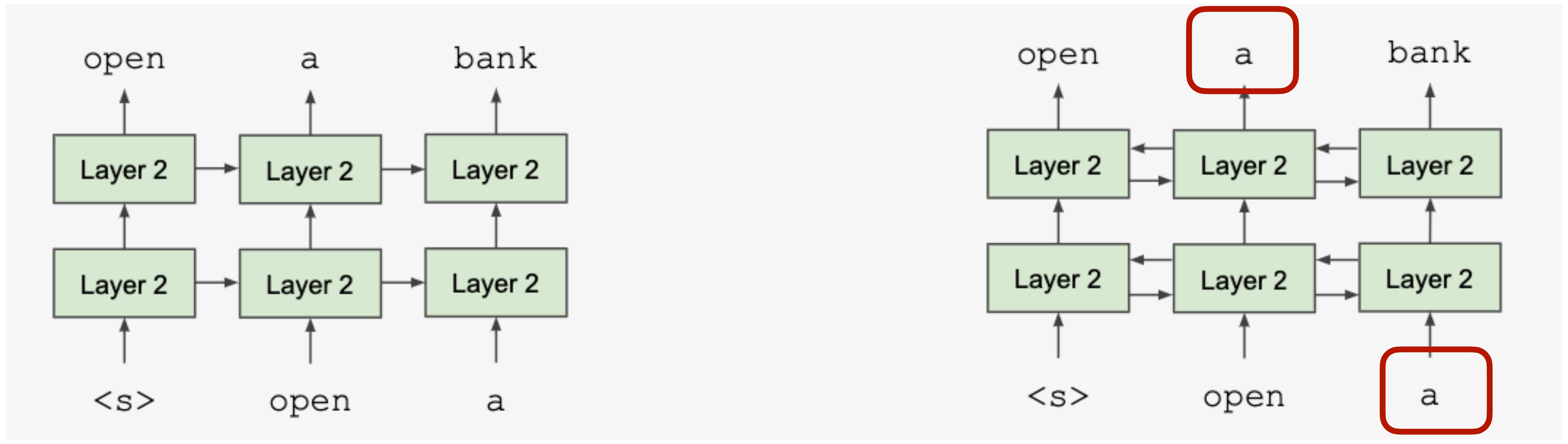
Example #2: I need to go to bank to make a deposit.

- Two new pre-training objectives:
  - **Masked language modeling (MLM)**
  - Next sentence prediction (NSP) - Later work shows that NSP hurts performance though..

(Devlin et al, 2019): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Masked Language Modeling (MLM)

- Q: Why we can't do language modeling with bidirectional models?



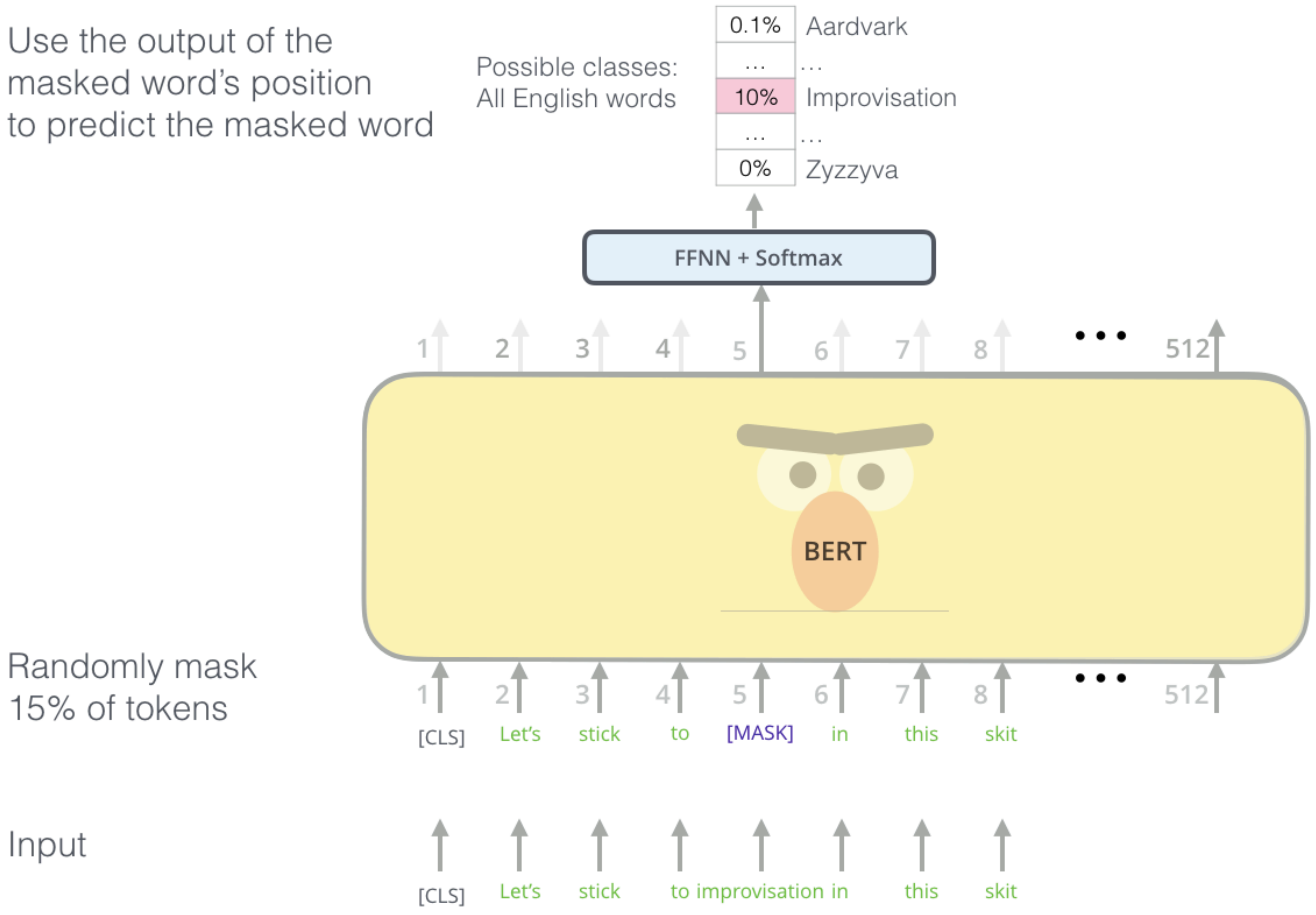
- Solution: Mask out k% of the input words, and then predict the masked words

store ↑ gallon ↑

the man went to [MASK] to buy a [MASK] of milk

k = 15% in practice

Masked Language Modeling (MLM)



Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
... ...
10% Improvisation
... ...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
1 2 3 4 5 6 7 8 ... 512
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

http://jalammar.github.io/illustrated-bert/

MLM: 80-10-10 corruption

For the 15% predicted words,

- 80% of the time, they replace it with [MASK] token

went to the store $\longrightarrow$ went to the [MASK]

- 10% of the time, they replace it with a random word in the vocabulary

went to the store $\longrightarrow$ went to the running

- 10% of the time, they keep it unchanged

went to the store $\longrightarrow$ went to the store

Why? Because [MASK] tokens are never seen during fine-tuning

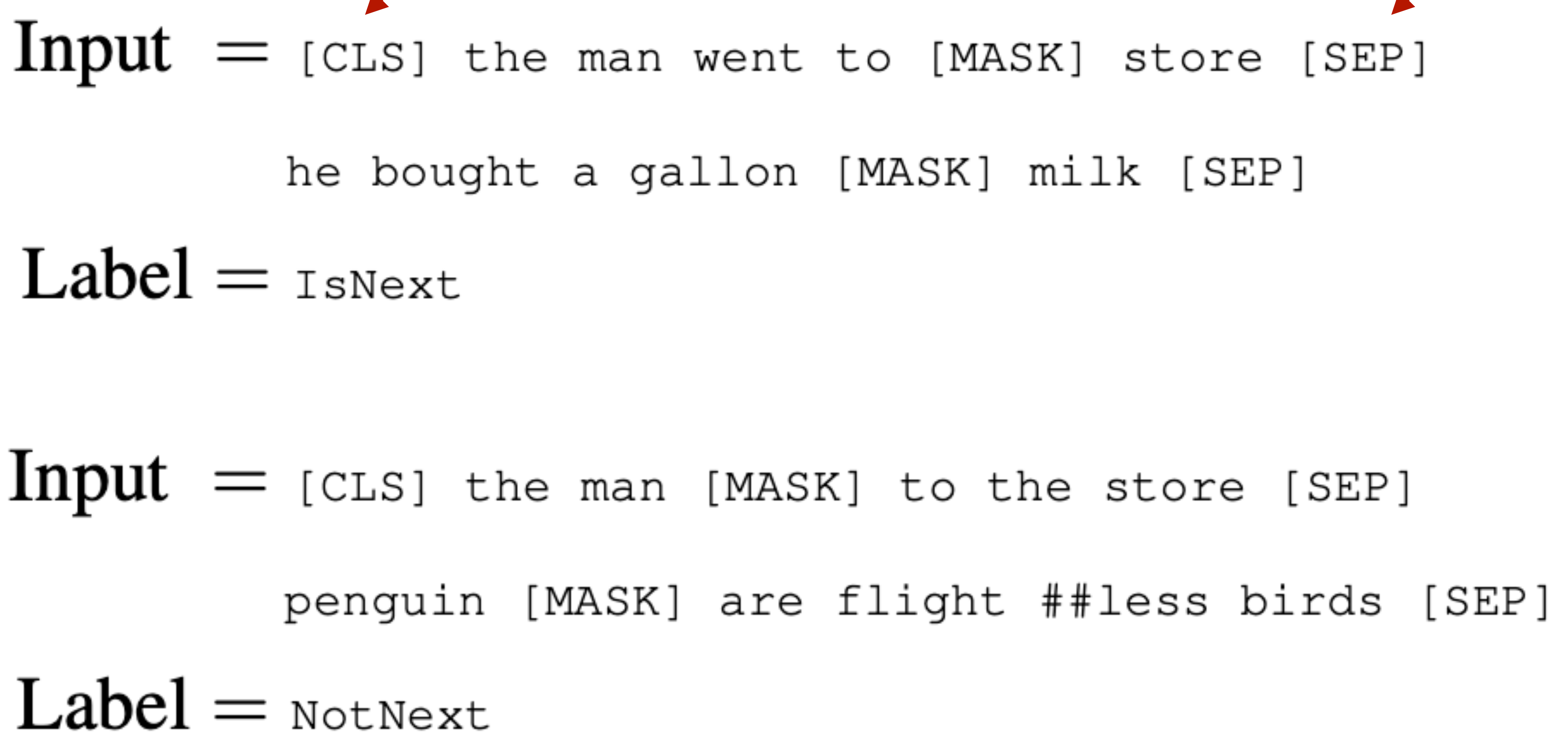
(See Table 8 of the paper for an ablation study)

Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA)
- NSP is designed to reduce the gap between pre-training and fine-tuning

[CLS]: a special token always at the beginning

[SEP]: a special token used to separate two segments



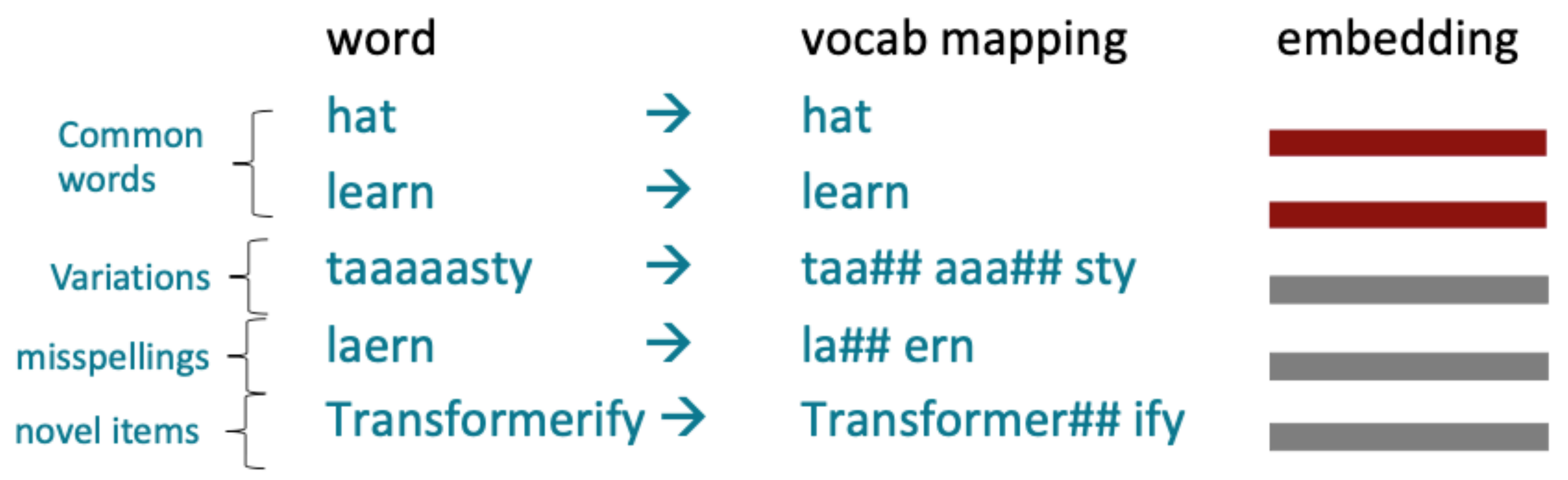
Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

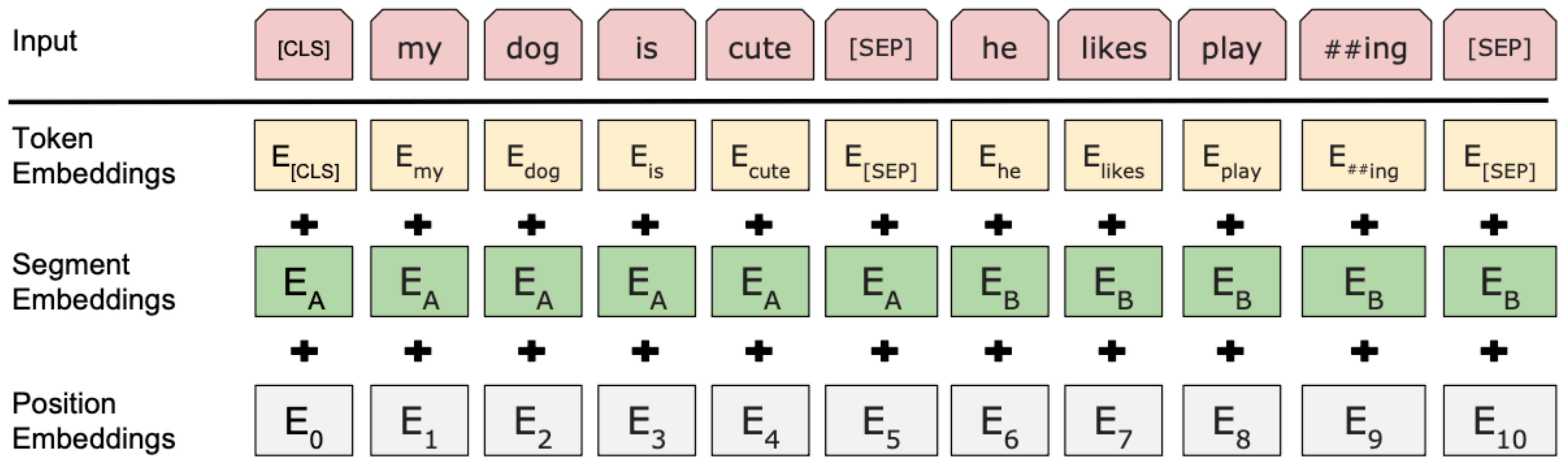
BERT pre-training

- Vocabulary size: 30,000 wordpieces (common sub-word units) (Wu et al., 2016)



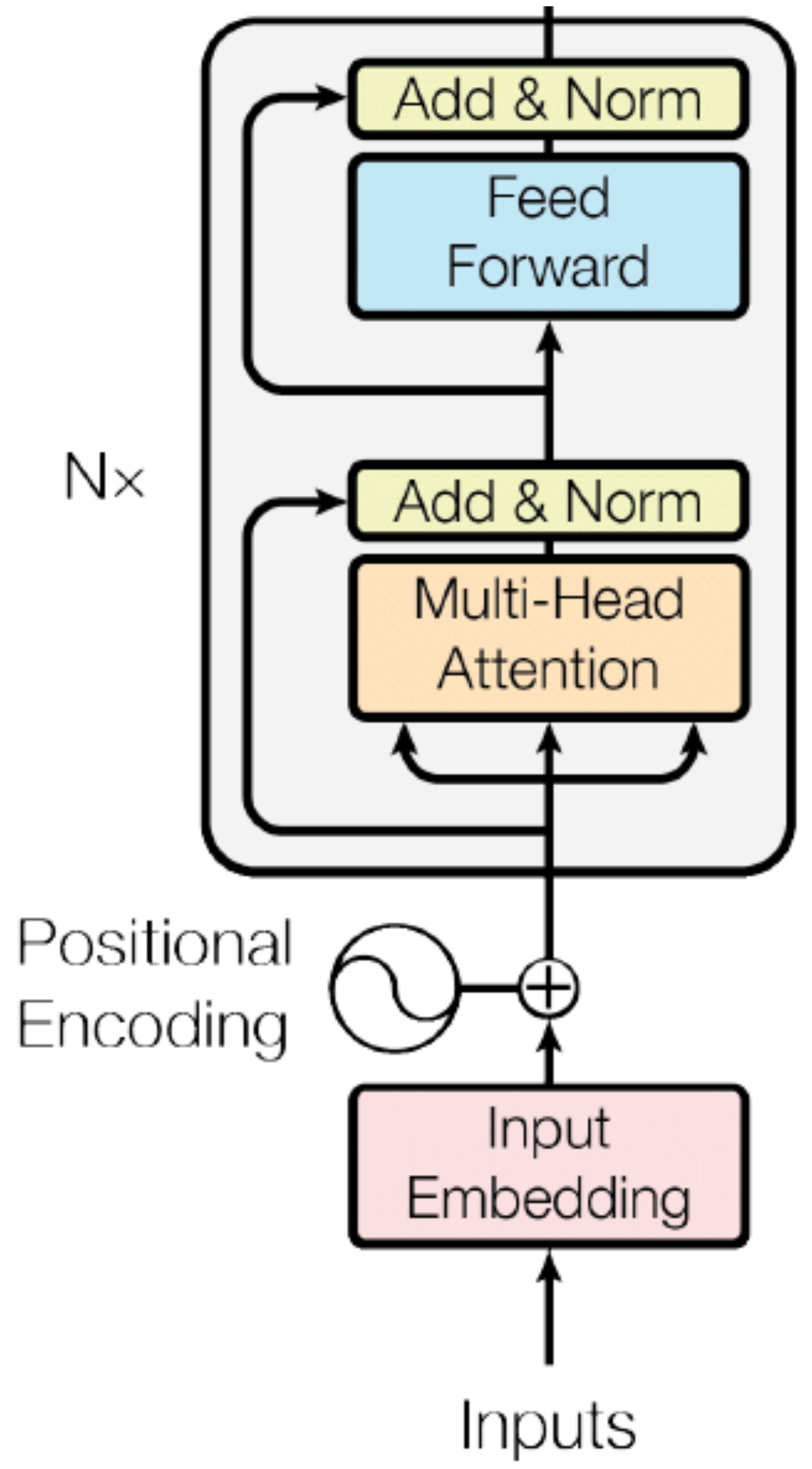
(Image: Stanford CS224N)

- Input embeddings:


Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
E[CLS] Emy Edog Eis Ecute E[SEP] Ehe Elikes Eplay E##ing E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB EB
Position Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 E10

Separate two segments

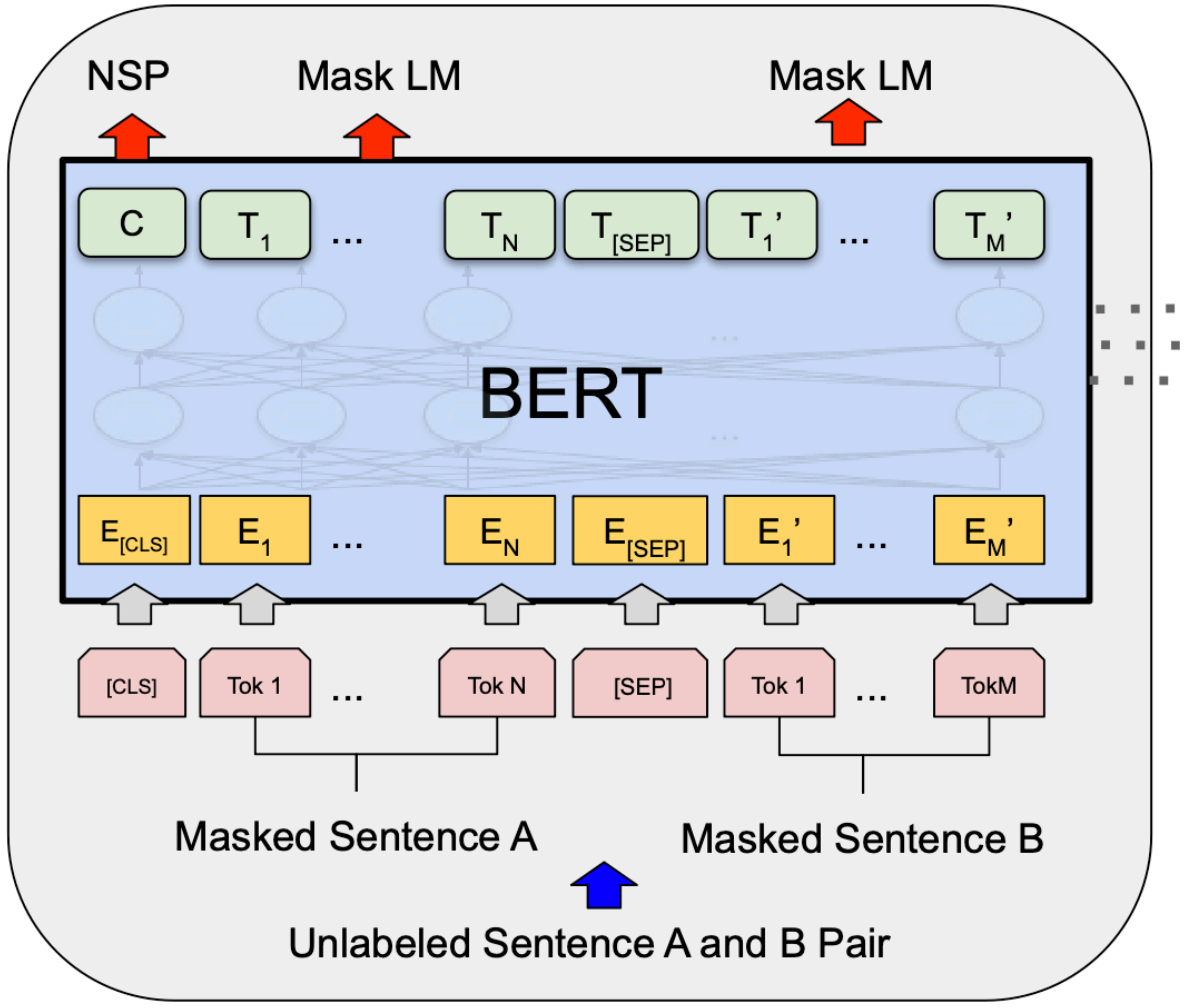
BERT pre-training



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters

- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

BERT pre-training



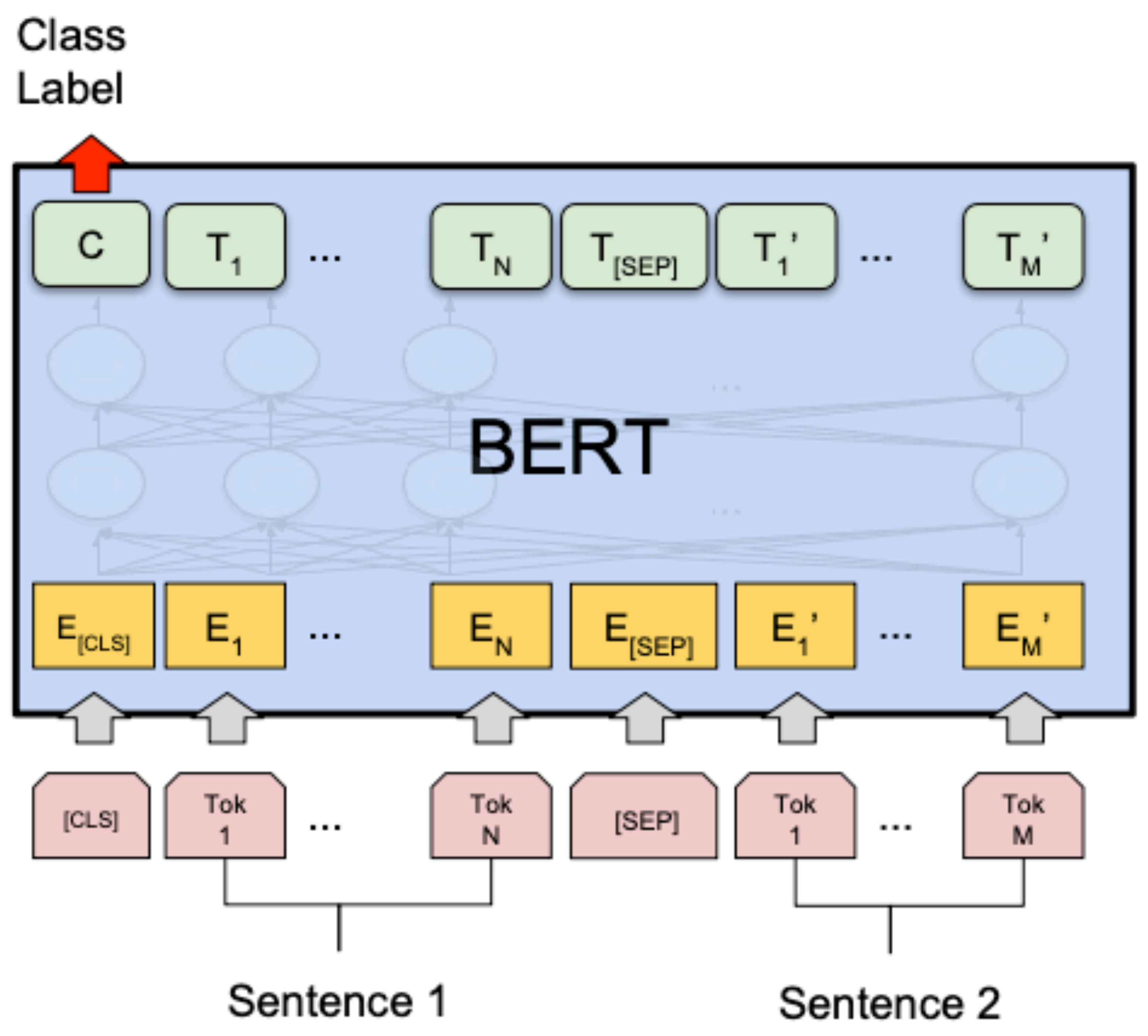
Pre-training

- MLM and NSP are trained together
- [CLS] is pre-trained for NSP
- Other token representations are trained for MLM

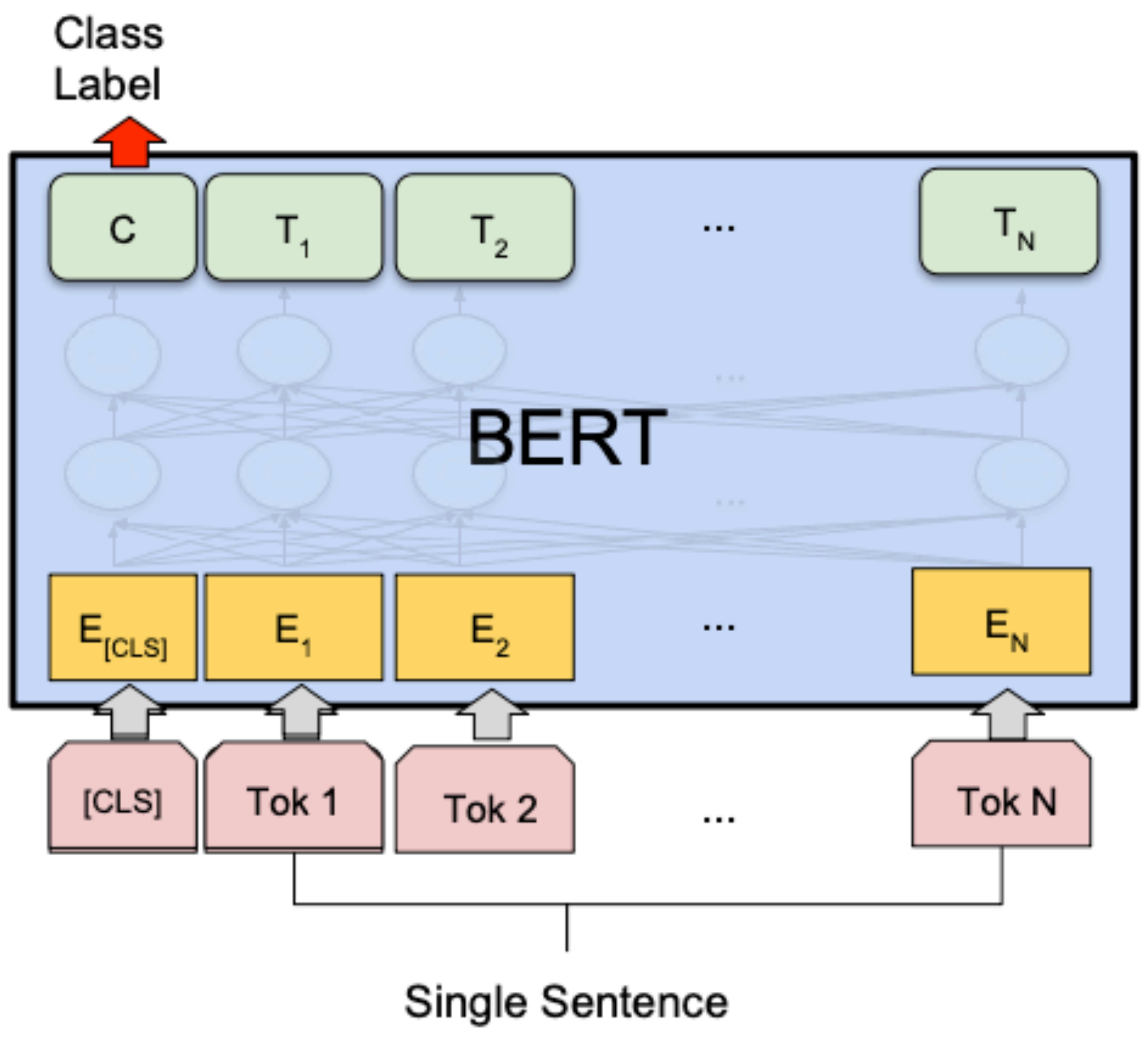
BERT fine-tuning

"Pre-train once, finetune many times."

sentence-level tasks



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

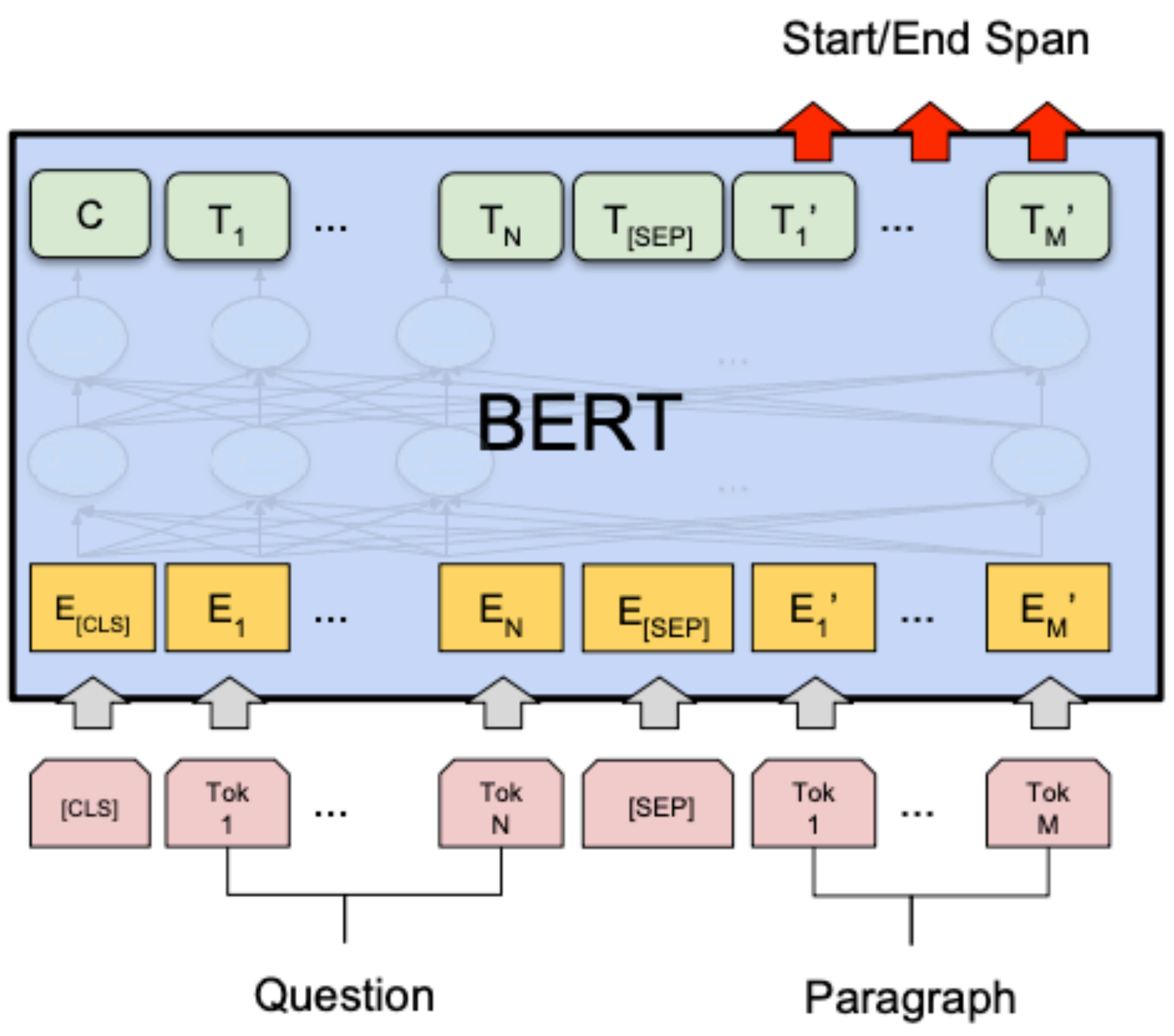


(b) Single Sentence Classification Tasks: SST-2, CoLA

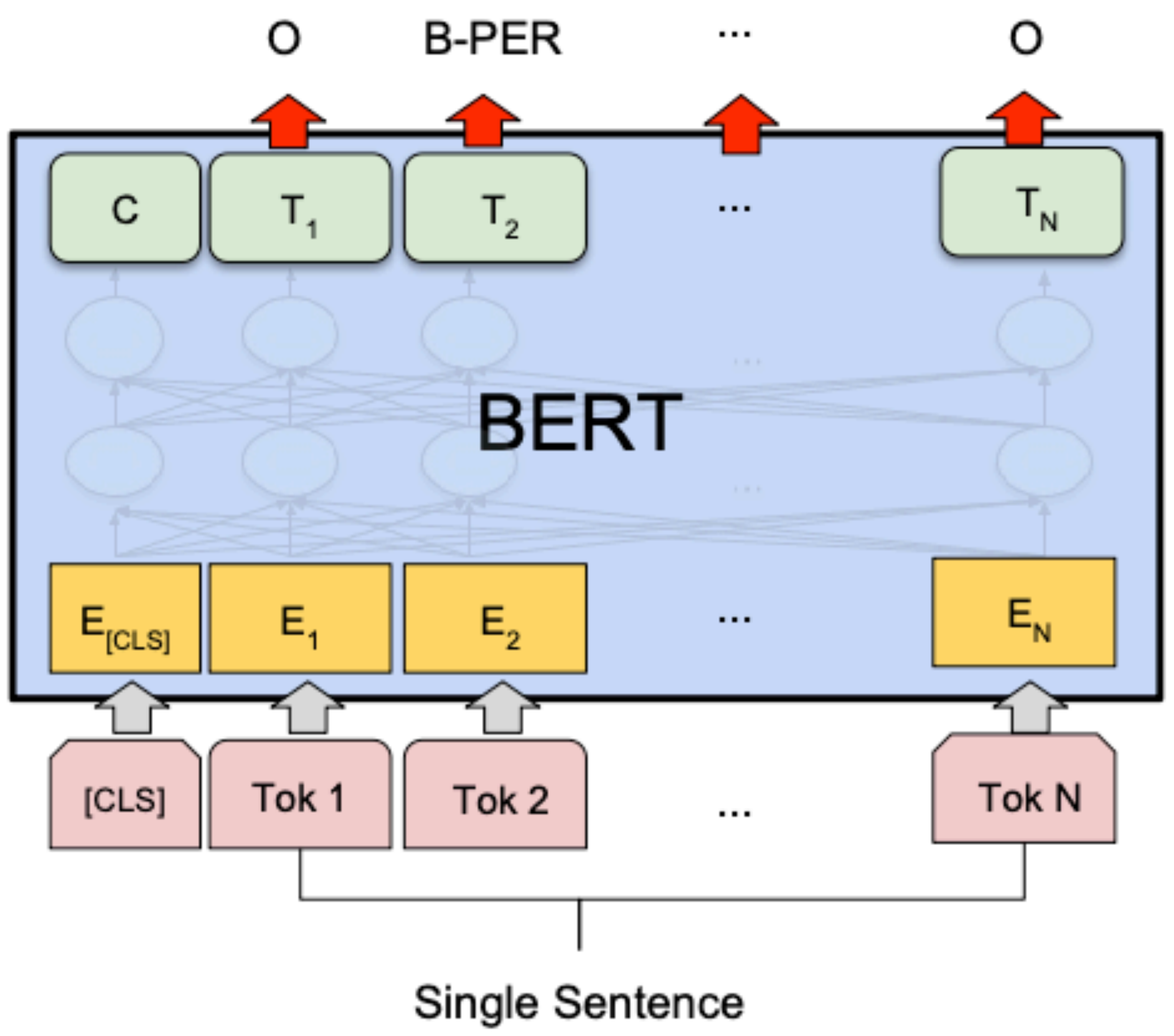
BERT fine-tuning

"Pretrain once, finetune many times."

token-level tasks



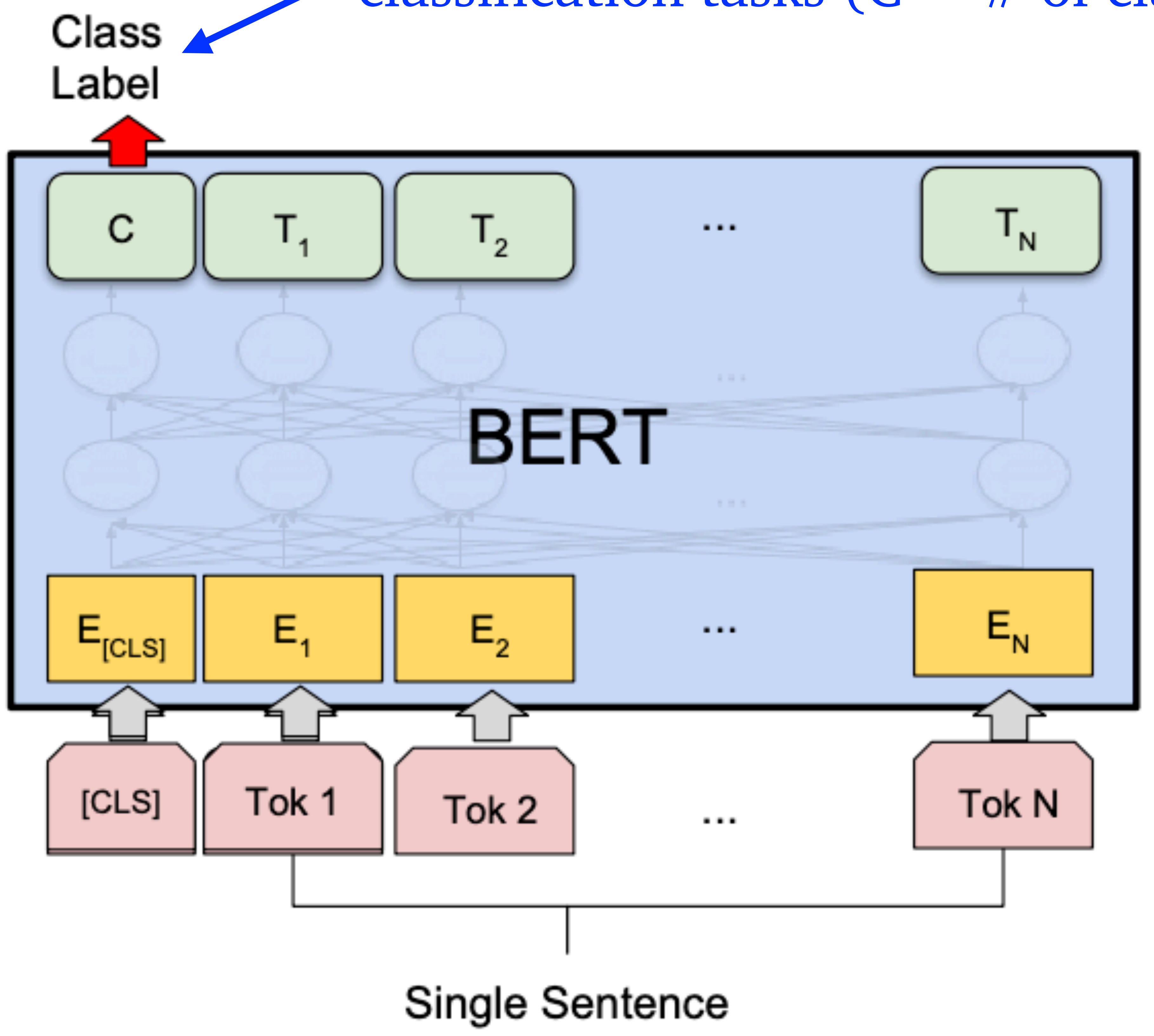
(c) Question Answering Tasks:
SQuAD v1.1



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

Example: sentiment classification

We just need to introduce $C \times h$ parameters for classification tasks (C = # of classes, h = hidden size)!



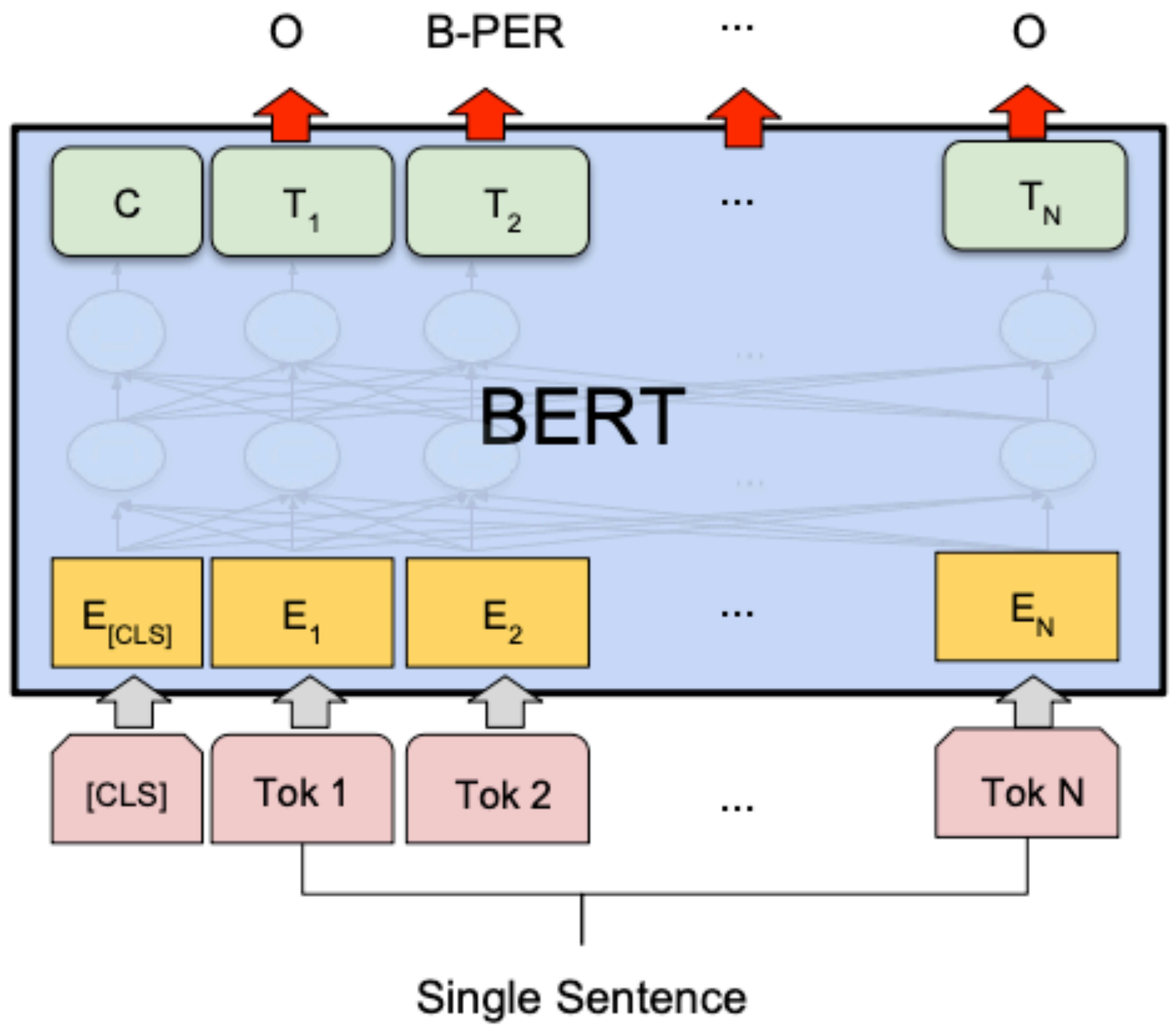
$$P(y = k) = softmax_k(\mathbf{W}_o \mathbf{h}_{[CLS]})$$

$$\mathbf{W}_o \in \mathbb{R}^{C \times h}$$

All the parameters will be learned together (original BERT parameters + new classifier parameters)

Example: named entity recognition (NER)

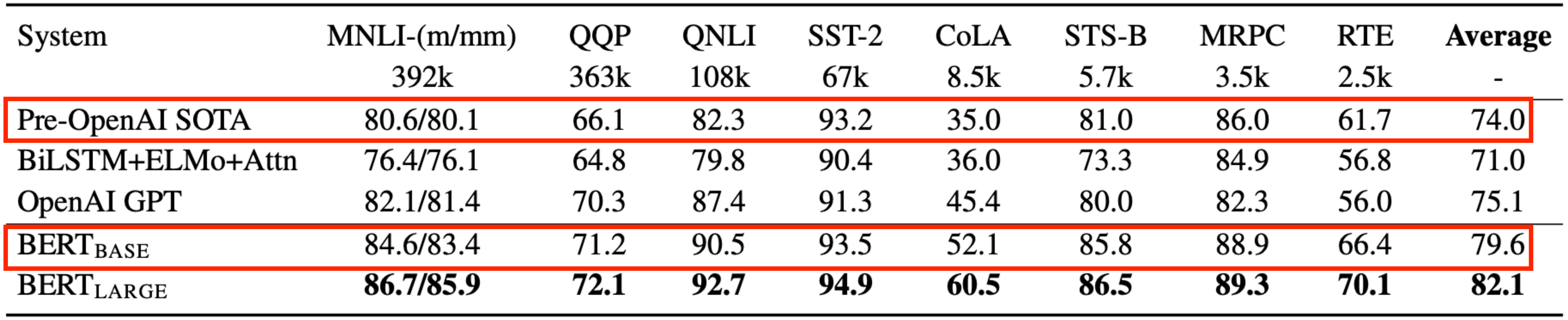
We just need to introduce $C \times h$ parameters for classification tasks (C = # of classes, h = hidden size)!



$$P(y_i = k) = softmax_k(\mathbf{W}_o \mathbf{h}_i)$$

$$\mathbf{W}_o \in \mathbb{R}^{C \times h}$$

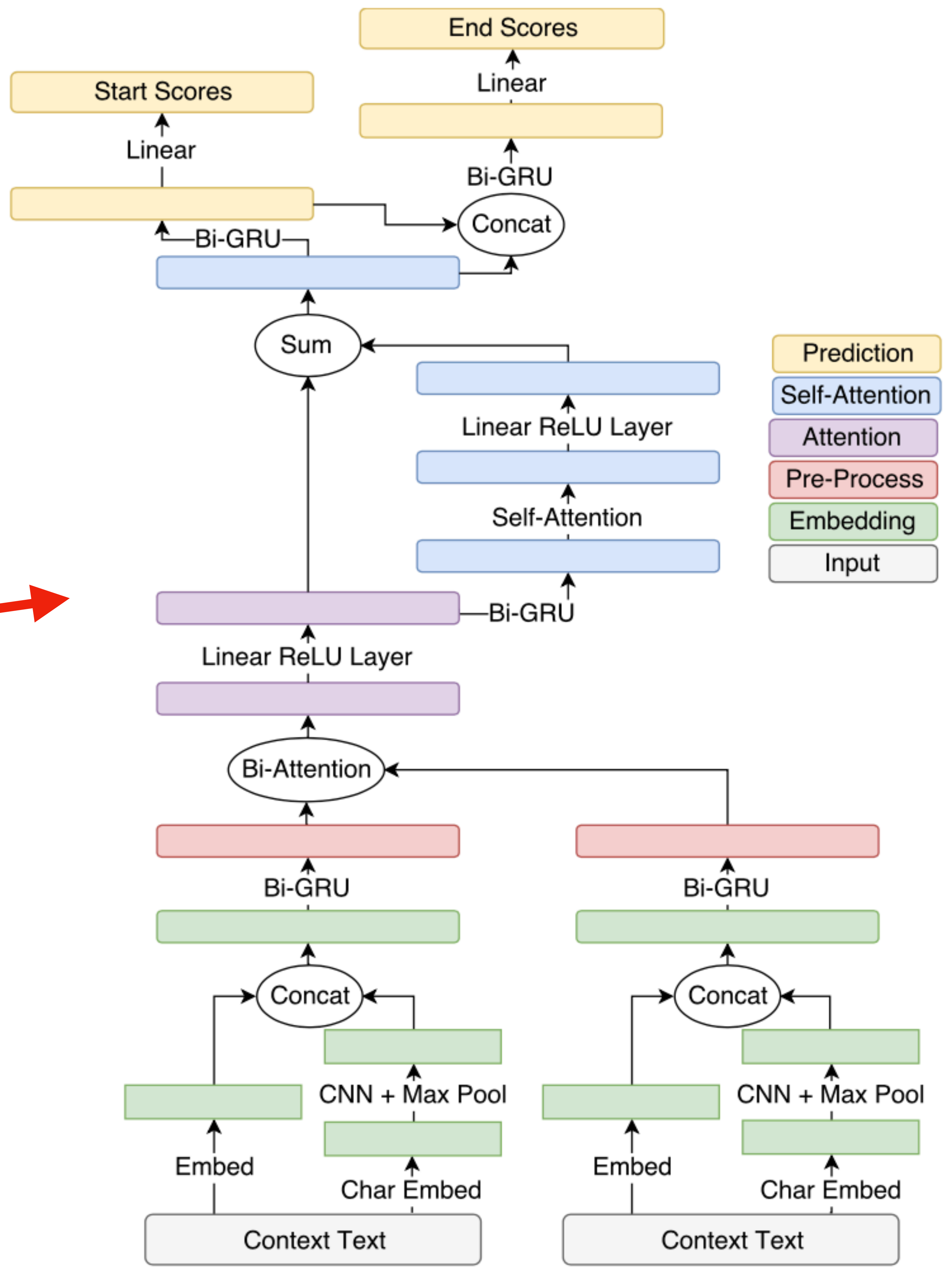
Experimental results: GLUE



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**

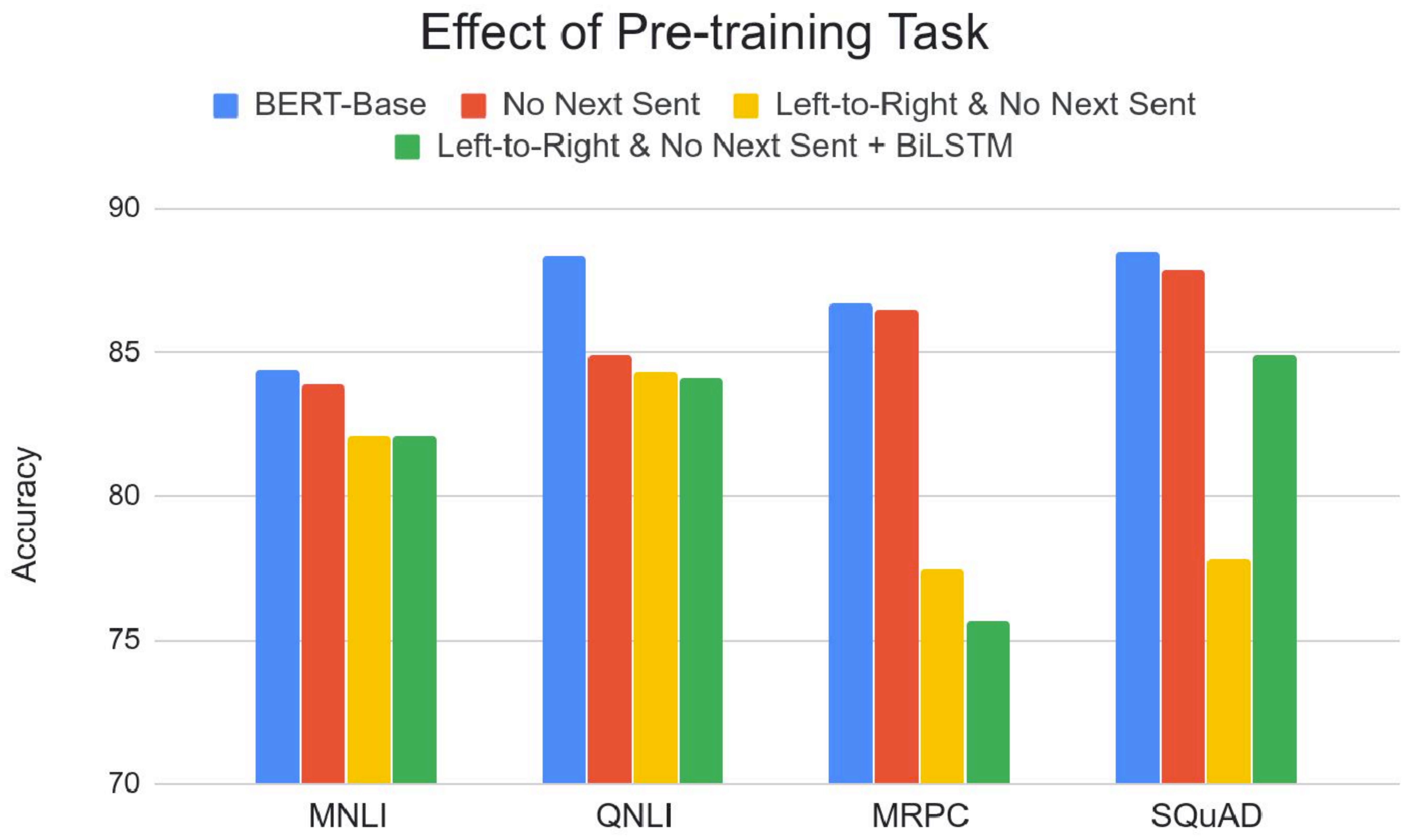
Experimental results: SQuAD

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**



SQuAD = Stanford Question Answering dataset

Ablation study: pre-training tasks



- MLM >> left-to-right LMs
- NSP improves on some tasks
- Note: later work (Joshi et al., 2020; Liu et al., 2019) argued that NSP is not useful

Ablation study: model sizes

# layers

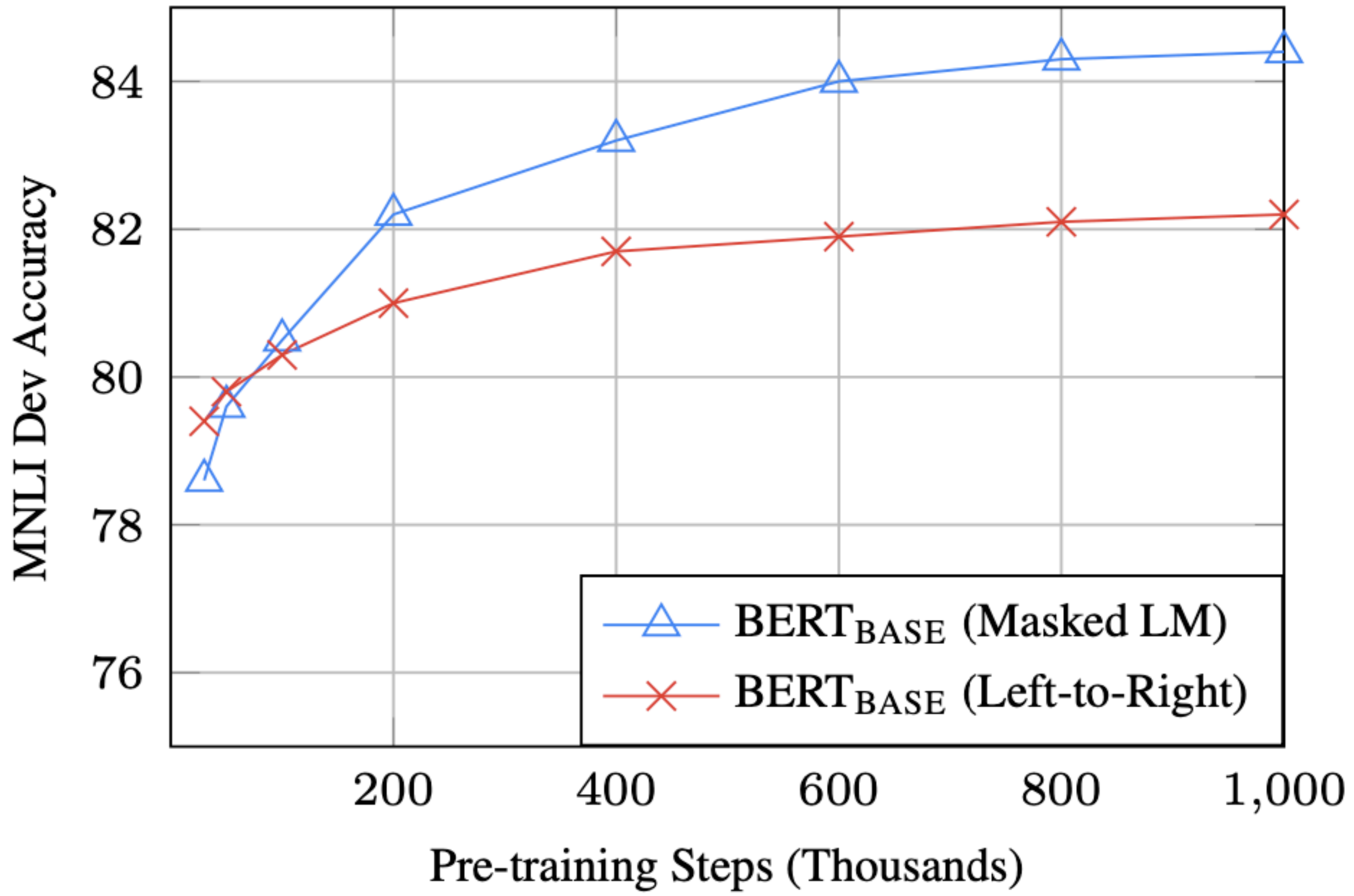
hidden size

# of heads

Hyperparams			Dev Set Accuracy			
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

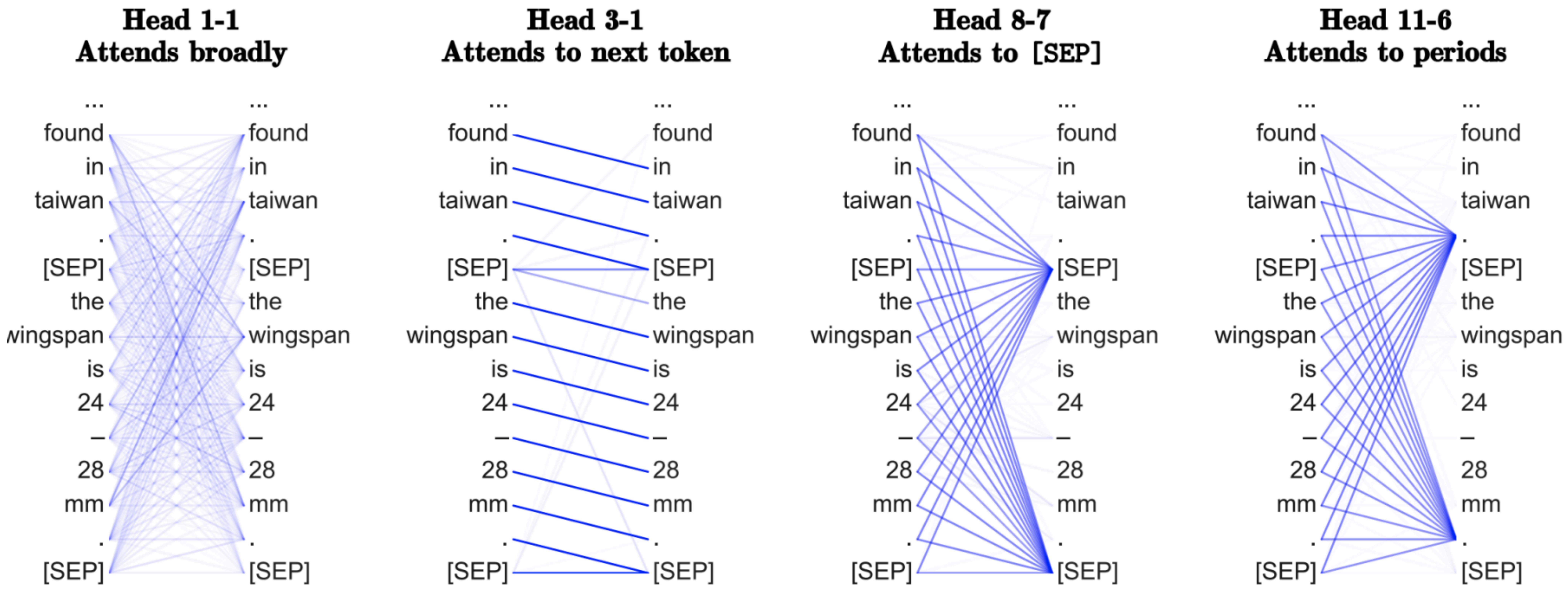
The bigger, the better!

Ablation study: training efficiency



MLM takes slightly longer to converge because it only predicts 15% of tokens

What does BERT learn?



(Clark et al., 2019) What Does BERT Look At? An Analysis of BERT's Attention

Limitations of pre-trained encoders

- Why not use pre-trained encoders for everything?
- If your task involves generating sequences, BERT and other pre-trained encoders don’t naturally lead to nice autoregressive (1-word-at-a-time) generation methods.
- Might want to use a pre-trained decoder

Slide credit: John Hewitt, Stanford CS224N

Pre-training for three types of architectures

- The neural architecture influences the type of pre-training and natural use cases:
  - **Encoders**: Gets bidirectional context
  - **Encoder-decoders**: Gets good parts of encoders and decoders?
  - **Decoders**: Language models!

Objective: Span corruption!

Replace different length spans from the input with placeholders; decode the spans that were removed.

Inputs: Thank you <X> me to your party <Y> week

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

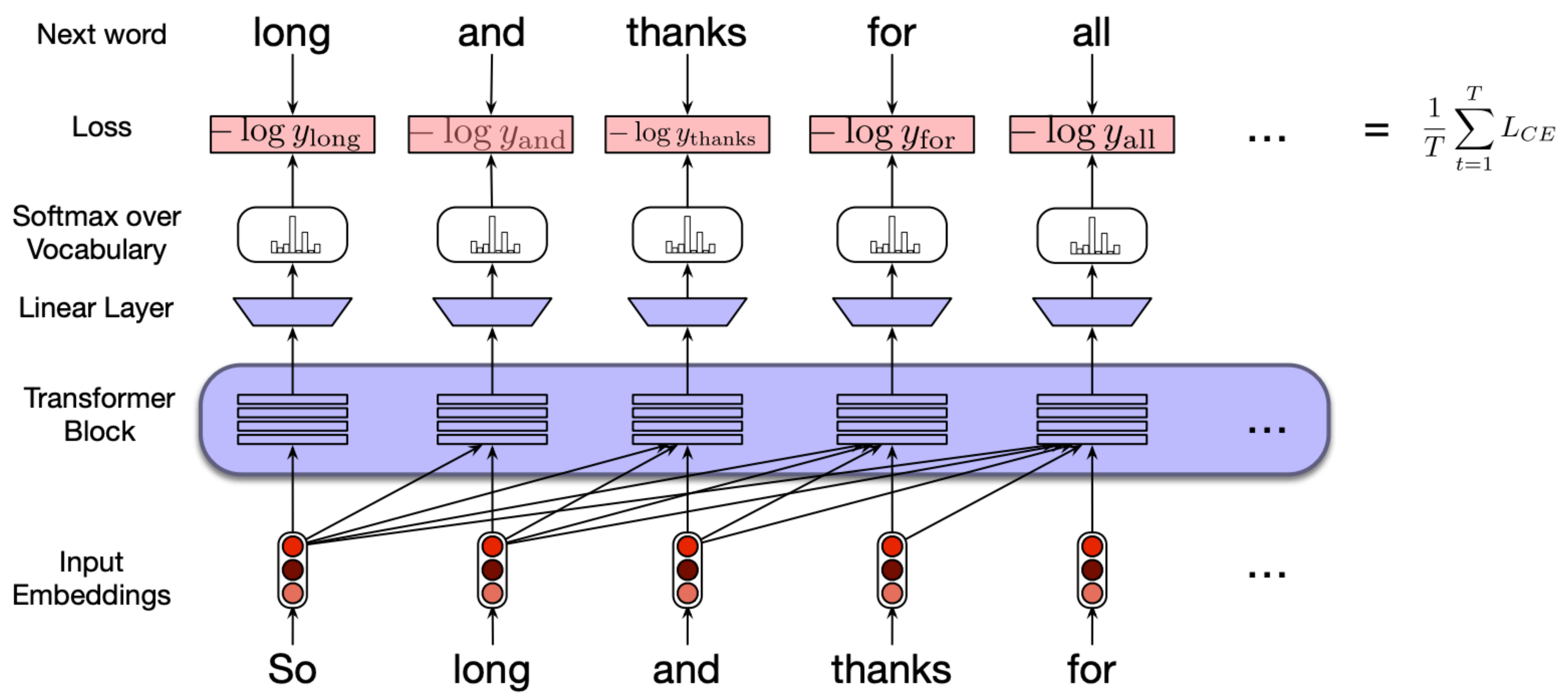
Pre-training for three types of architectures

- The neural architecture influences the type of pre-training and natural use cases:
  - **Encoders**: Gets bidirectional context
  - **Encoder-decoders**: Gets good parts of encoders and decoders?
  - **Decoders**: Language models!

Generative Pre-Training (GPT)

(Released in 2018/6)

- Use a **Transformer decoder** (unidirectional; left-to-right) instead of LSTMs
- Use **language modeling** as a pre-training objective
- Trained on longer segments of text (**512 BPE tokens**), not just single sentences

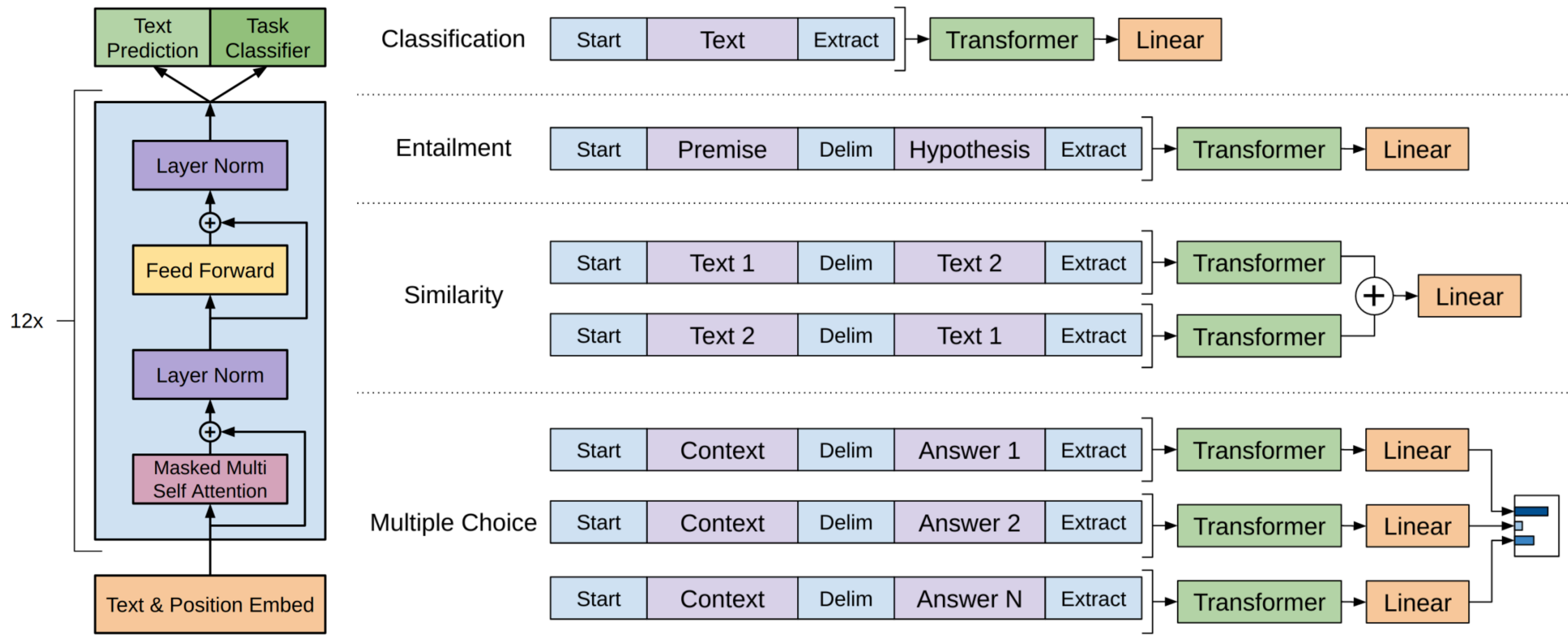


(Radford et al, 2018): Improving Language Understanding by Generative Pre-Training

Generative Pre-Training (GPT)

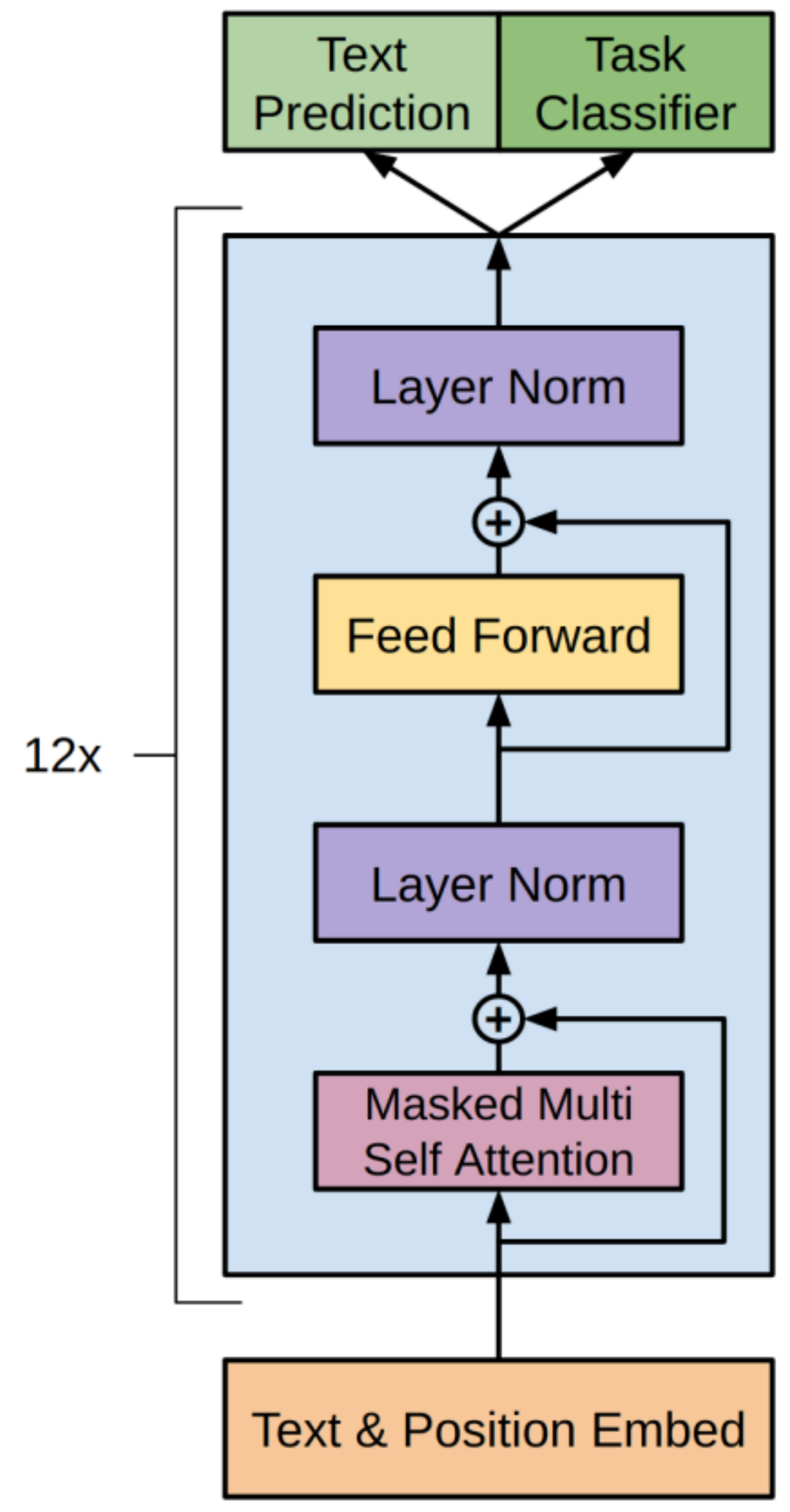
(Released in 2018/6)

- "Fine-tune" the entire set of model parameters on various downstream tasks



(Radford et al, 2018): Improving Language Understanding by Generative Pre-Training

GPT: More details



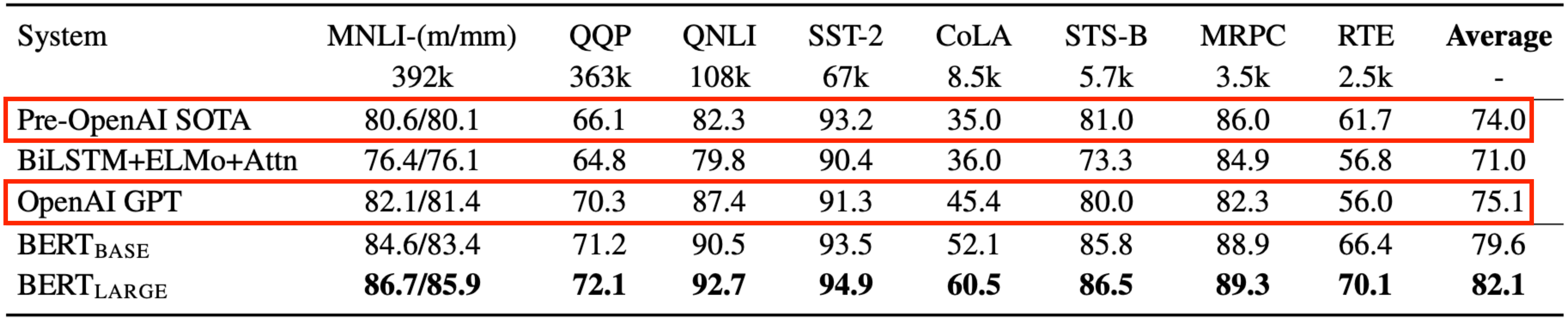
- 12 layers, 768 hidden size, 12 attention heads, 110M parameters

Same as BERT-base

Recall: BERT was trained on this + Wikipedia!

- Training corpus: BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens
- Trained for 100 epochs, batch size 64

Experimental results: GLUE



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**