Introduction to NLP

CSE5321/CSEG321

Lecture 13. Pretraining

Hwaran Lee ( )

Materials are referenced from Stanford CS224n

Lecture Plan

Lecture 13: Pre-training

1. Contextualized word embeddings

2. Pre-training and fine-tuning

3. GPT, ELMo, BERT

Contextualized Word Embeddings

Contextualized Word Embeddings

Limitations of word2vec

- One vector for each word type
  - (a.k.a. “static word embedding”)

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

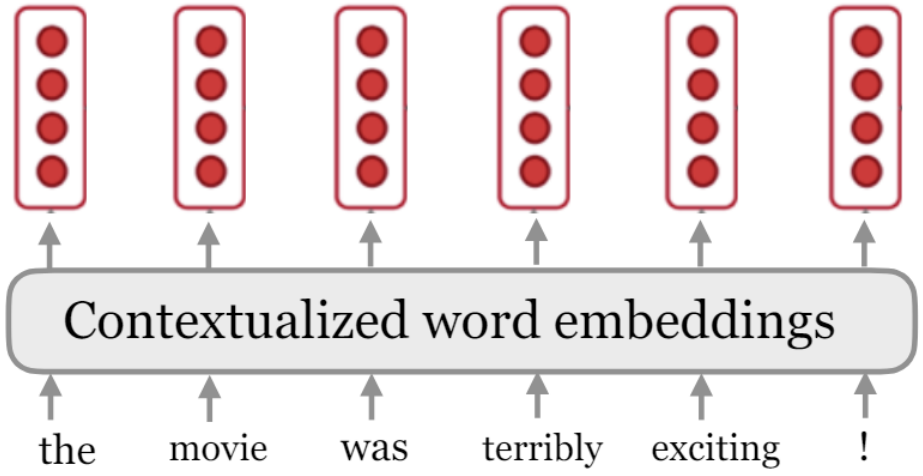
- Complex characteristics of word use: syntax and semantics

- Polysemous words, e.g., bank, mouse
  - $mouse^1$: a mouse controlling a computer system in 1968.
  - $mouse^2$: a quite 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

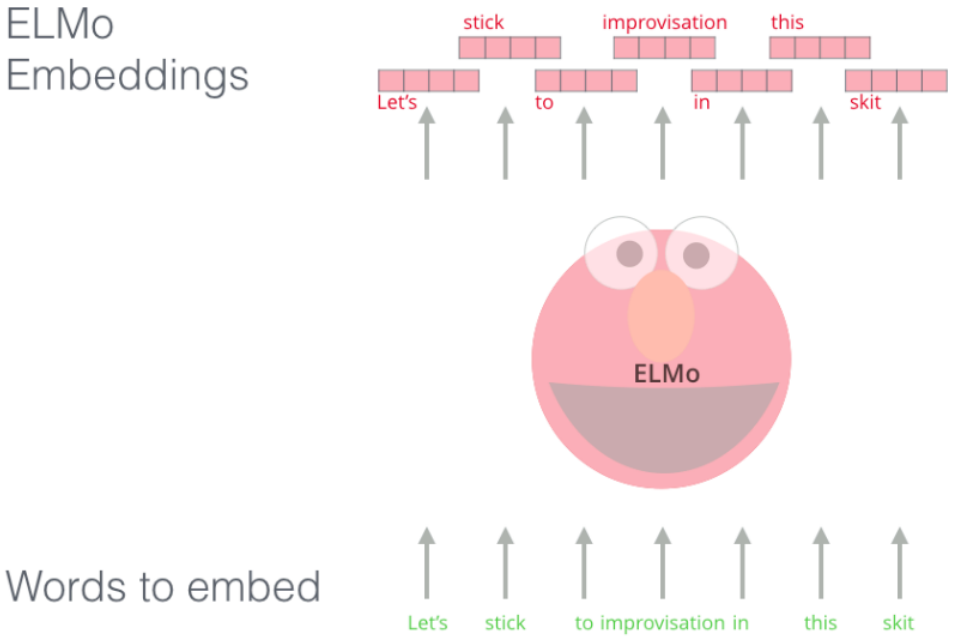
Contextualized Word Embeddings

Motivation

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



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



Contextualized Word Embeddings

Motivation

- Sent #1: 그 사람은 미국으로 가버렸다. $\mathbf{e}(\text{가버렸다}) = ?$
- Sent #2: 그 양반 갈 때도 예술로 가버렸다. $\mathbf{e}(\text{가버렸다}) = ?$
- Sent #3: 그 오래된 와인은 맛이 가버렸다. $\mathbf{e}(\text{가버렸다}) = ?$
- Sent #4: 그녀의 사연이 이해가 가버렸다. $\mathbf{e}(\text{가버렸다}) = ?$
- Sent #5: 결국 그는 먼 곳으로 가버렸다. $\mathbf{e}(\text{가버렸다}) = ?$

Contextualized Word Embeddings

Motivation

- Sent #1: 그 사람은 미국으로 가버렸다.

- Which of the following $\mathbf{e}(\text{가버렸다})$ is expected to have the most similar vector to the first one?

  - (a) Sent #2: 그 양반 갈 때도 예술로 가버렸다.

  - (b) Sent #3: 그 오래된 와인은 맛이 가버렸다.

  - (c) Sent #4: 그녀의 사연이 이해가 가버렸다.

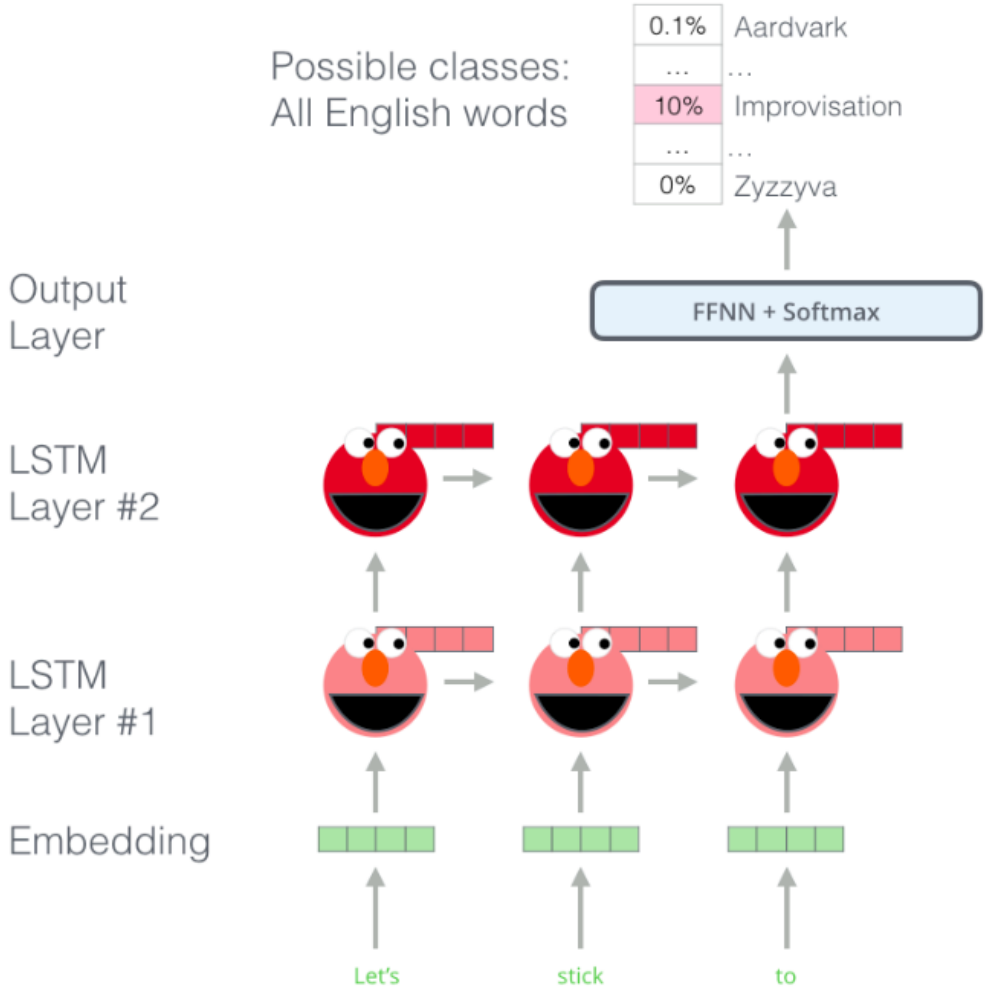
  - (d) Sent #5: 결국 그는 먼 곳으로 가버렸다.

ELMo: Embeddings from Language Models

The key idea of ELMo (Peters et al., 2018)

- 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



ELMo: Embeddings from Language Models

How does ELMo work?

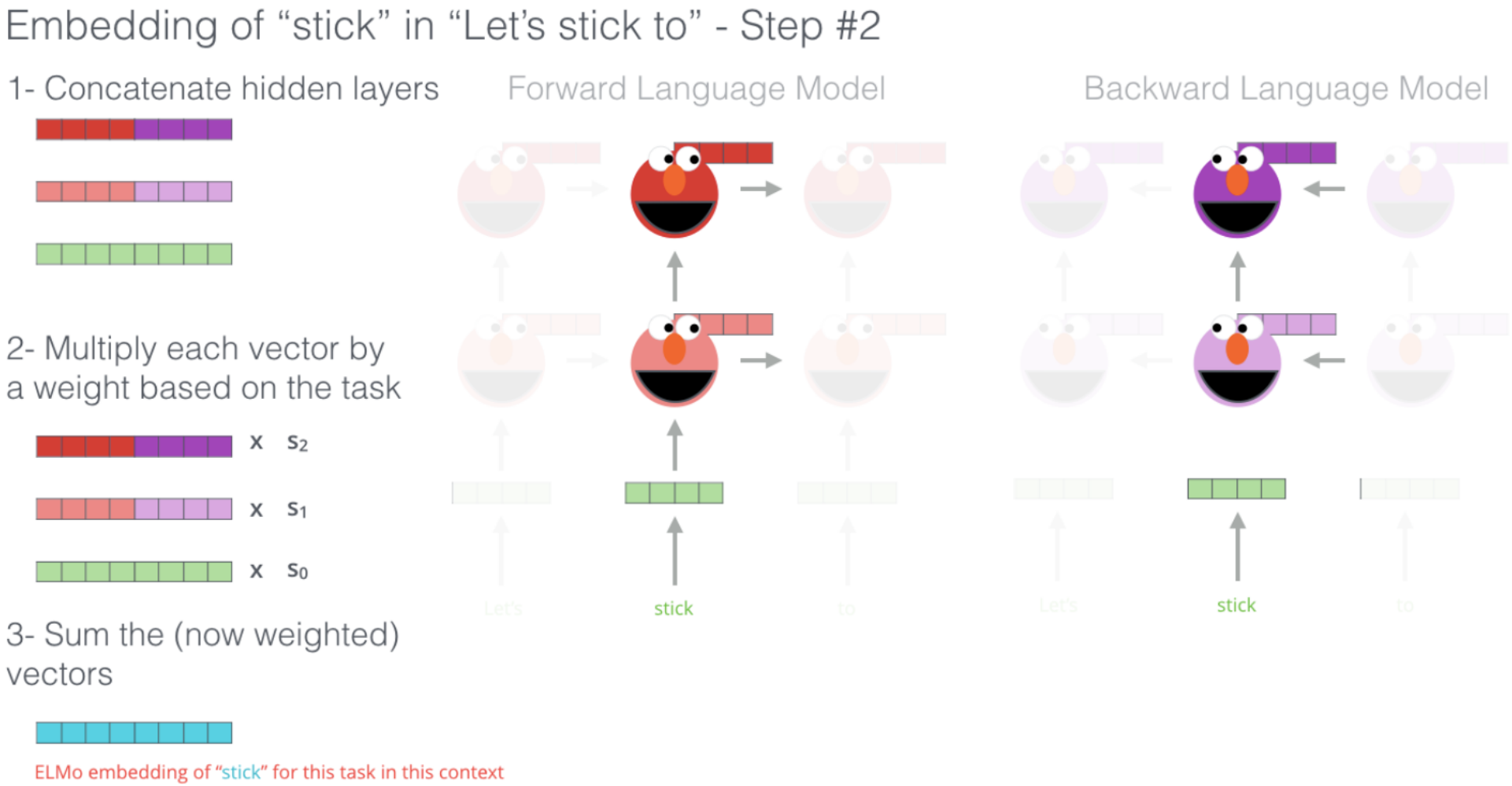
- Contextualized word embeddings = The weighted average of input embeddings + all hidden representations (j)

$$\mathrm{ELMo}_k^{task} = \mathbf{E}\left(R_k; \theta^{task}\right) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{LM}$$

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

ELMo: Embeddings from Language Models

How does ELMo work?

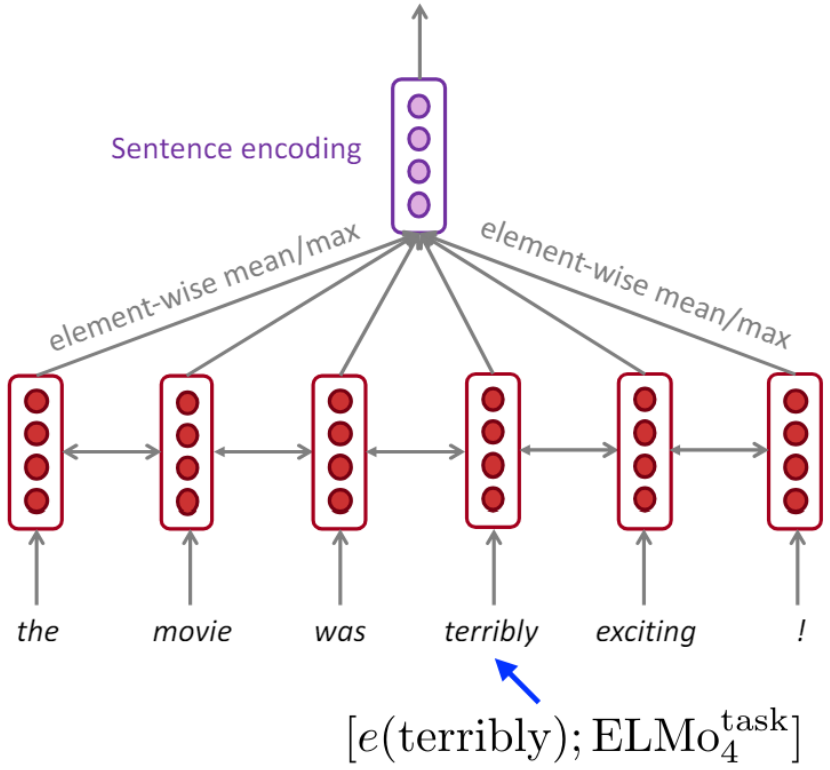


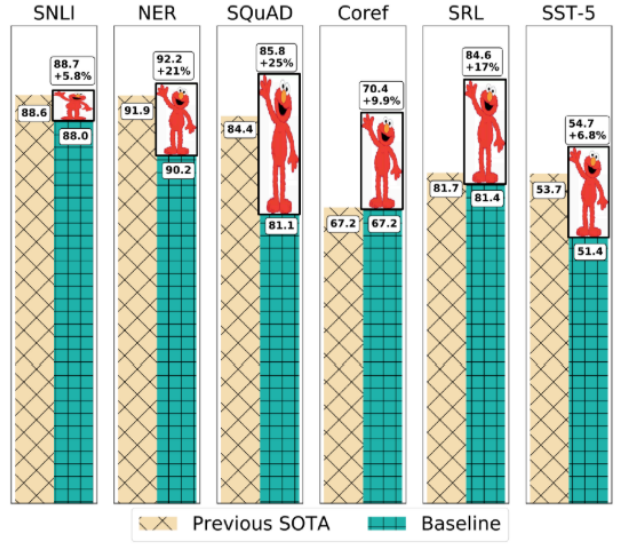
ELMo: Embeddings from Language Models

ELMo: pre-training and the use

- Data: 10 epochs on 1B Word Benchmark
- Training time: 2 weeks on 3 NVIDIA GTX 1080 GPUs

Example use: A BiLSTM model for sentiment classification





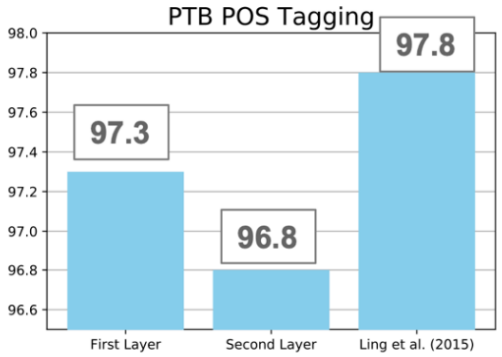
ELMo: Embeddings from Language Models

ELMo: some take-aways

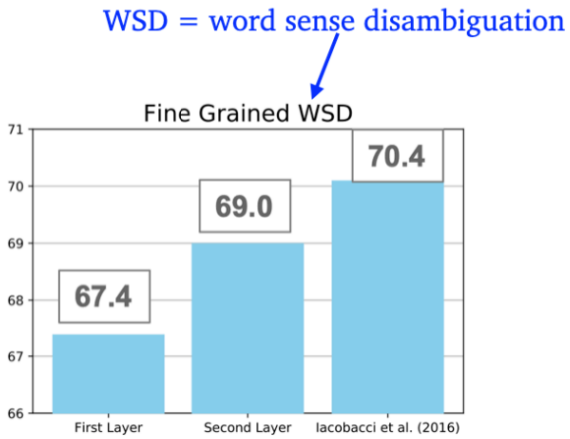
- Why use both forward and backward language models?
  - Because it is important to model both left and right context

  - Bidirectionality is very important in language understanding tasks

- 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



second layer > first layer

Pre-training and Fine-tuning

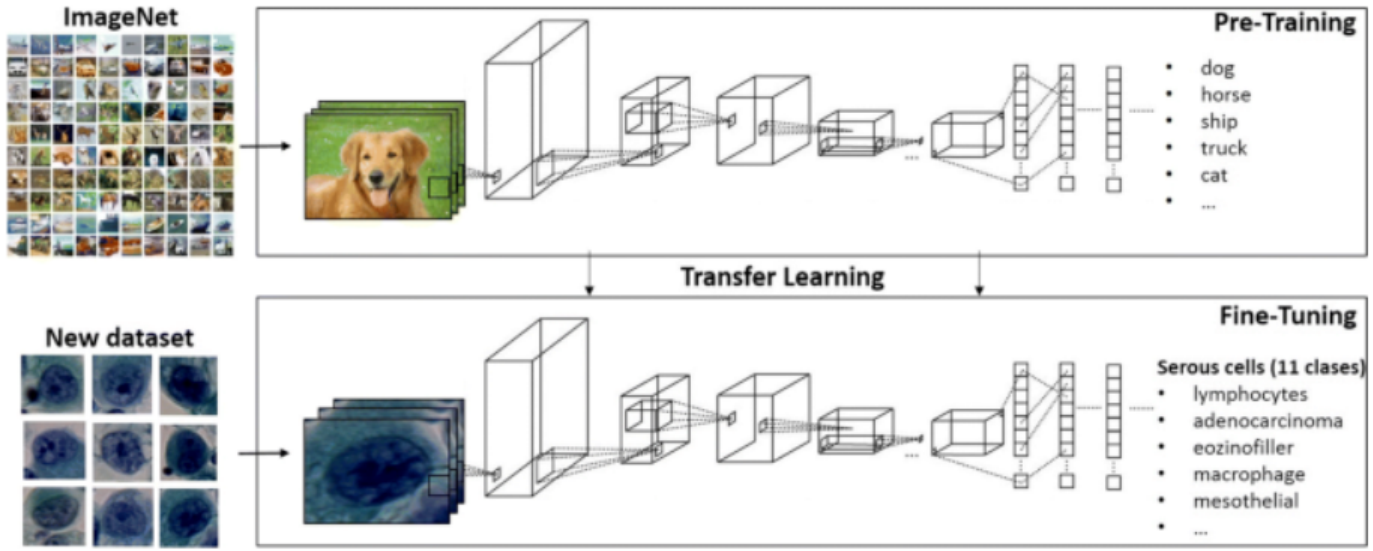
Pre-training and Fine-tuning

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

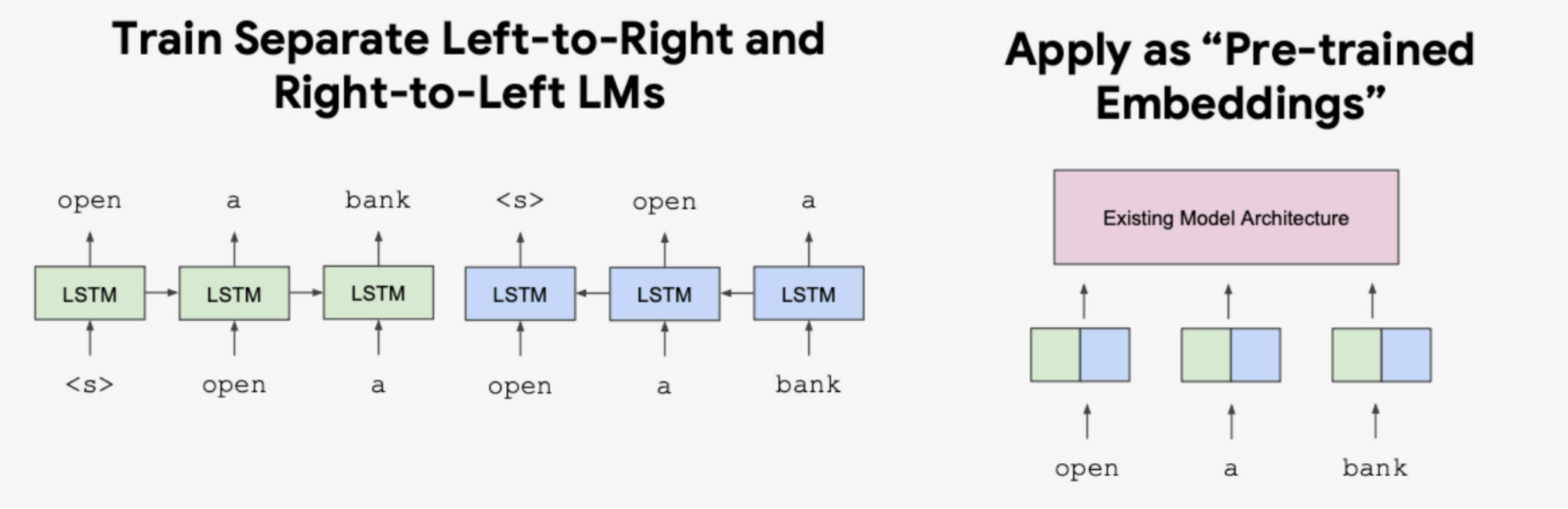
Can we find some task X that can be useful for a wide range of downstream tasks Y?



Pre-training and Fine-tuning

Feature-based vs fine-tuning approaches

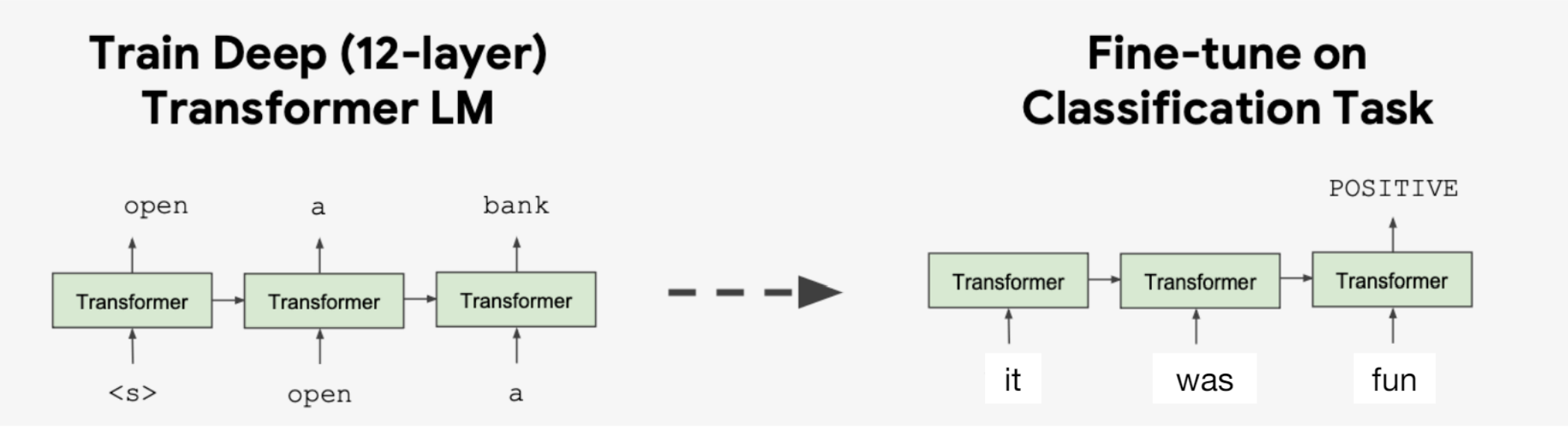
- ELMo is a feature-based approach which only produces word embeddings that can be used as input representations of existing neural models



Pre-training and Fine-tuning

Feature-based vs fine-tuning approaches

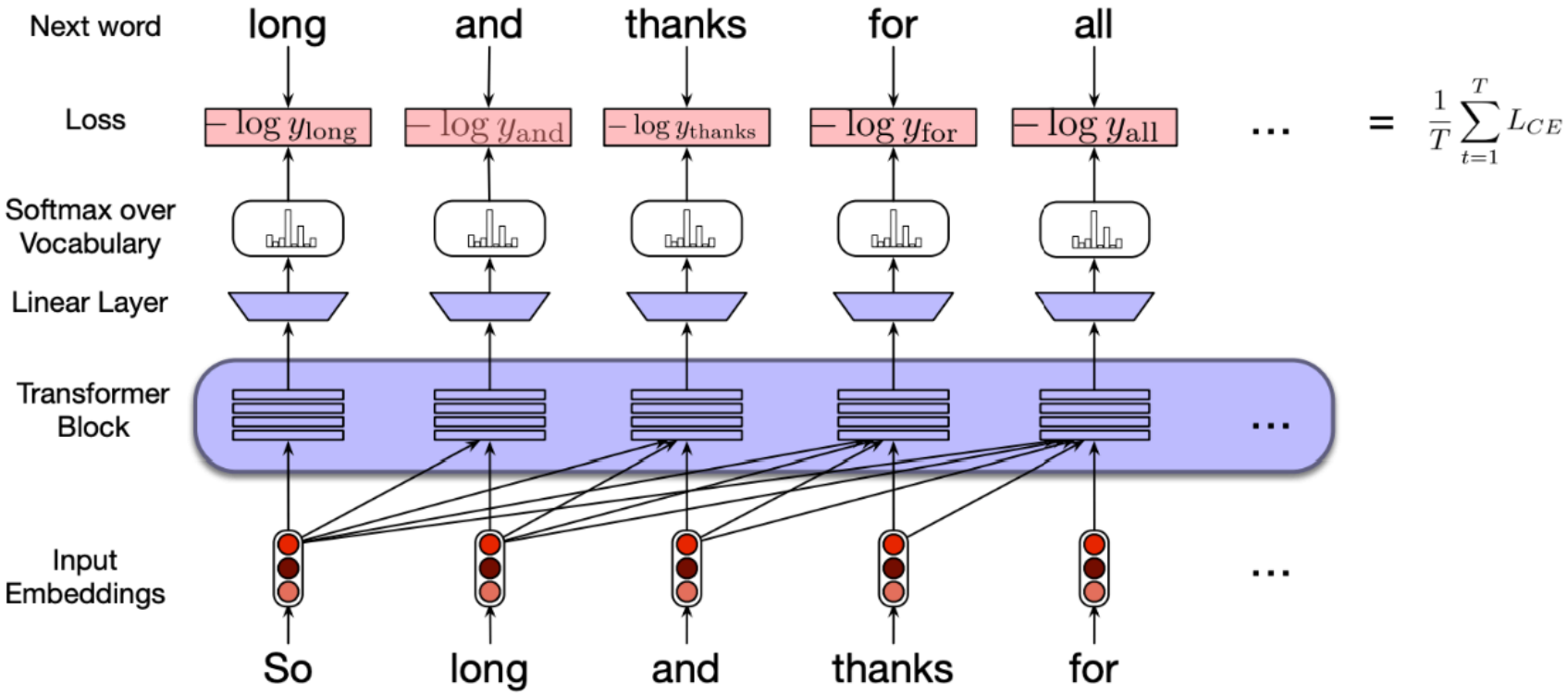
- GPT / BERT (and most of following models) are fine-tuning approaches
  - Almost all models weights will be re-used, and only a small number of task-specific weights will be added for downstream tasks



Pre-training and Fine-tuning

Generative Pre-Training (GPT) (Radford et al., 2018)

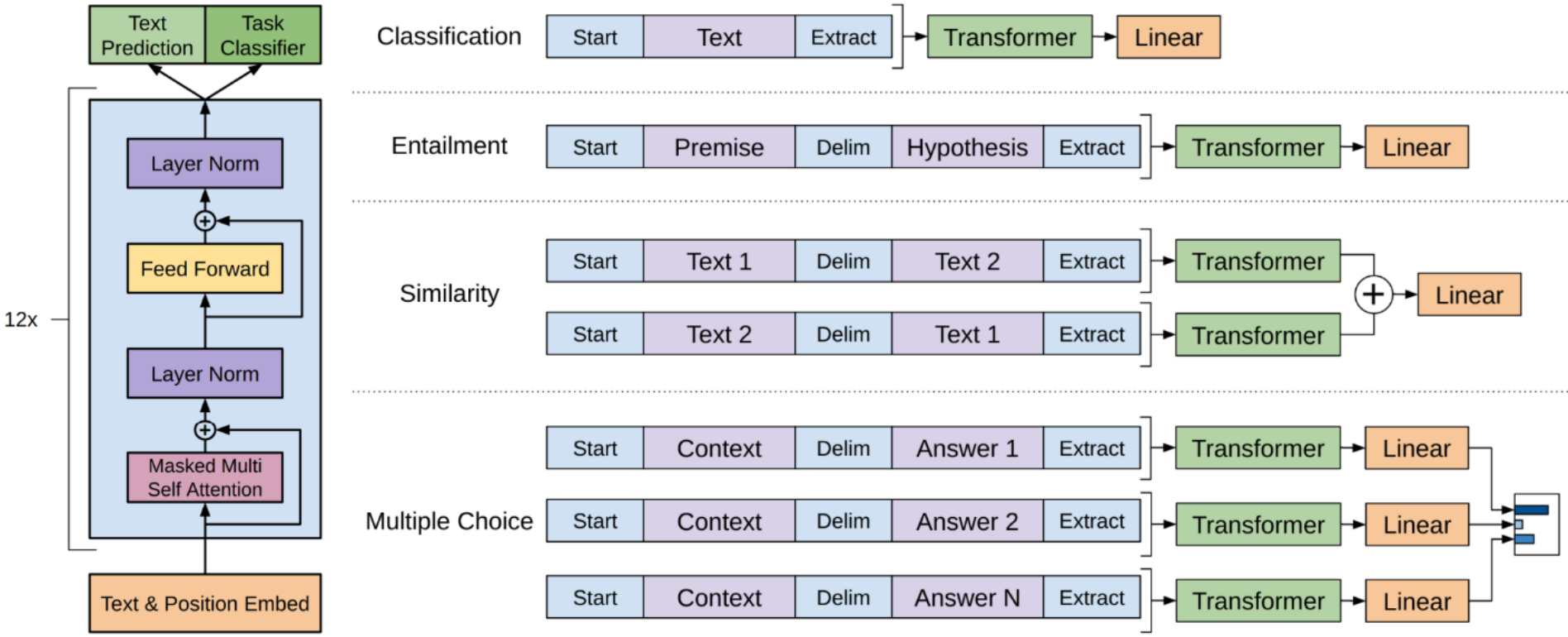
- Use a Transformer decoder (unidirectional; left-to-right) instead of LSTMs
- Use language models as a pre-training objective
- Trained on longer segments of text (512 BPE tokens), not just single sentences



Pre-training and Fine-tuning

Generative Pre-Training (GPT) (Radford et al., 2018)

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



Pre-training and Fine-tuning

BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2019)

- 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 (MLLM)
  - Next sentence prediction (NSP) – but later work shows that NSP hurts performance

Pre-training and Fine-tuning

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

Pre-training and Fine-tuning

MLM: 80-10-10 corruption

- For the 15% predicted words,
  - 80% of the training time, they replace it with [MASK] token
    - went to the store -> went to the [mask]
  - 10% of the training time, they replace it with a random word in the vocabulary
    - went to the store -> went to the running
  - 10% of the time, they keep it unchanged
    - went to the store -> went to the store

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

Pre-training and Fine-tuning

Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, question-answering)

- 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`

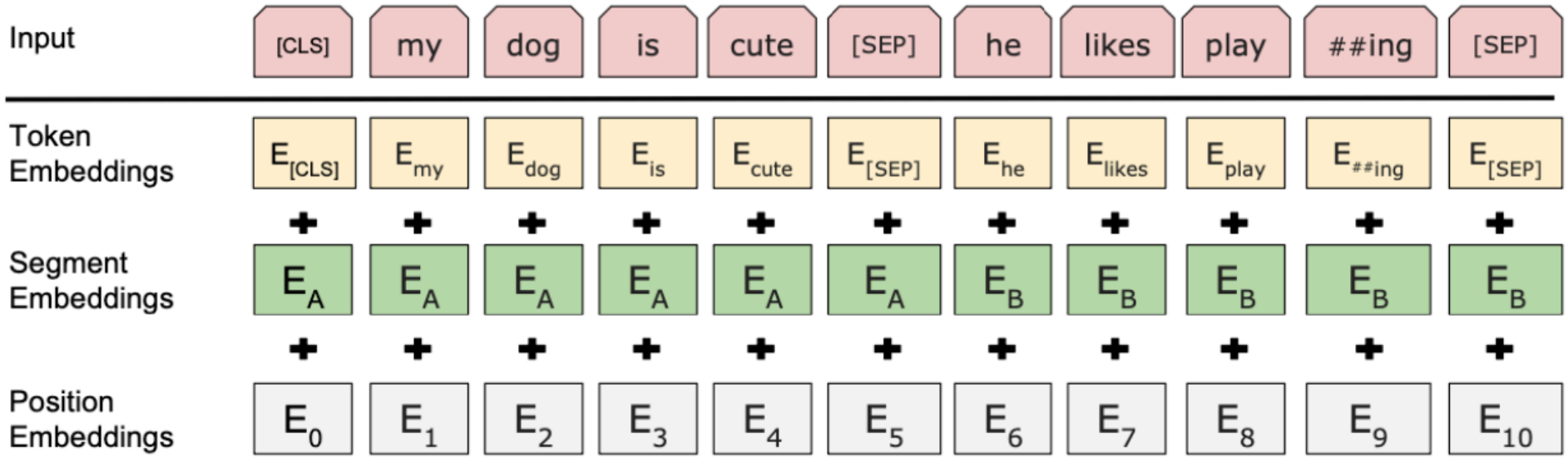
They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

Pre-training and Fine-tuning

BERT pre-training

- Vocabulary size: 30,000 wordpieces (common sub-word units)

- Input embeddings:



Pre-training and Fine-tuning

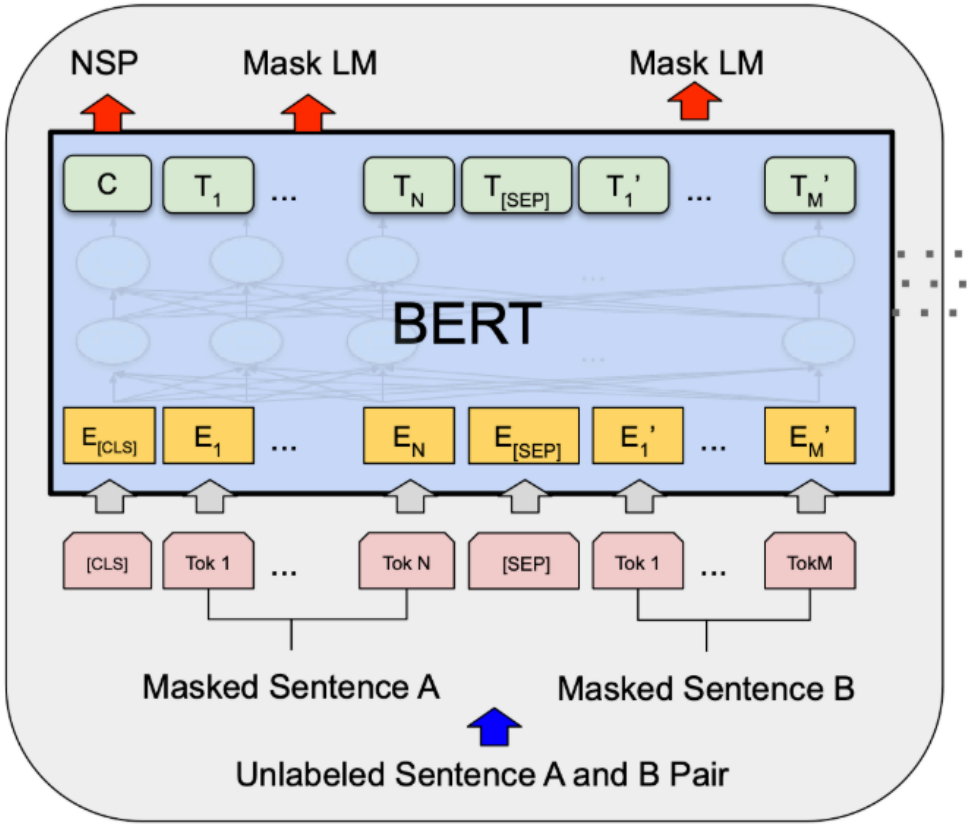
BERT pre-training

- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters (= same as OpenAI GPT)
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters
- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B) (OpenAI GPT was trained on BooksCorpus only)
- Max sequence size: 512 wordpiece tokens
- Trained for 1M steps, batch size 128K

Pre-training and Fine-tuning

BERT pre-training

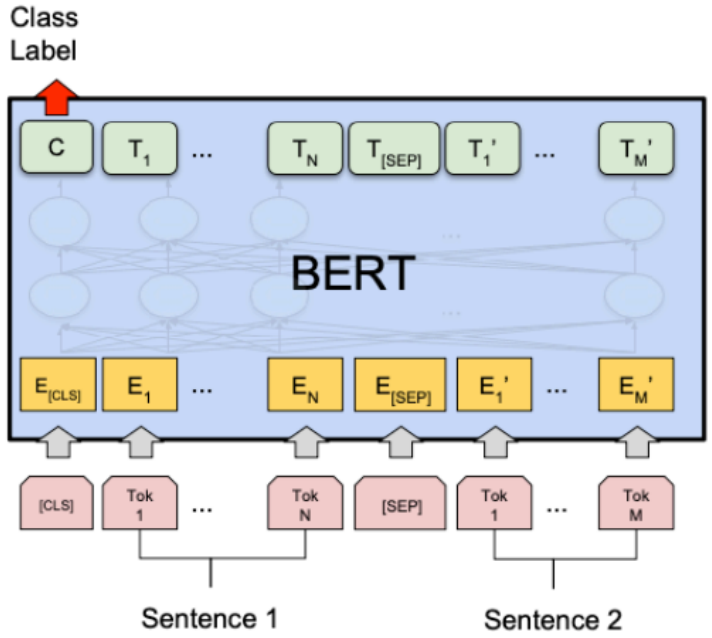
- MLM and NSP are trained together
- [CLS] is pre-trained for NSP
- Other token representations are trained for MLM



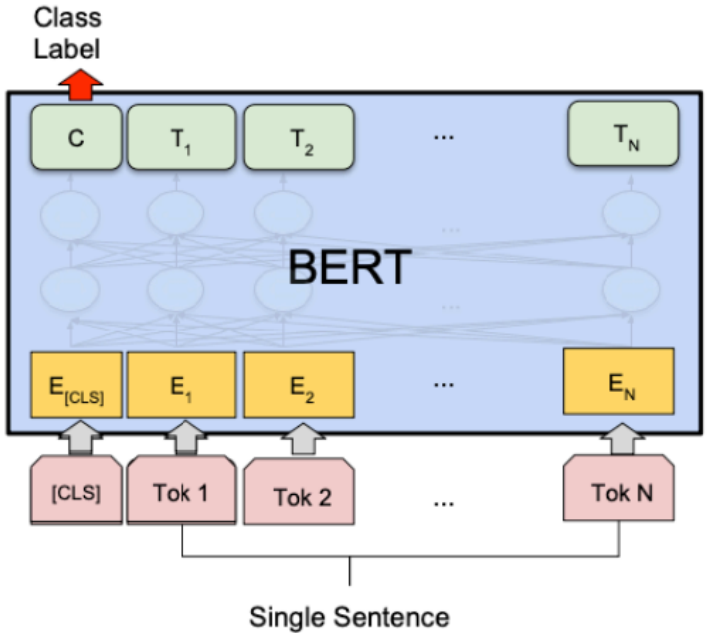
Pre-training and Fine-tuning

BERT pre-training

- Pre-train once, fine-tune many times (sentence-level tasks)



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

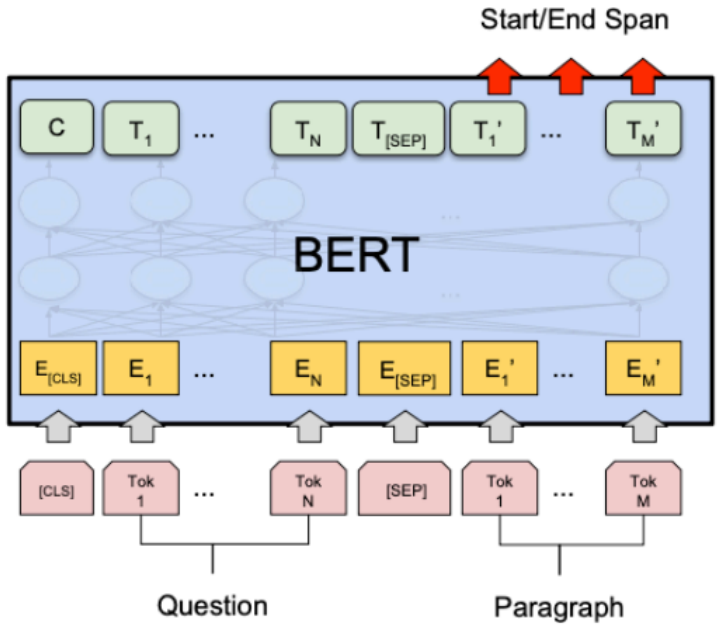


(b) Single Sentence Classification Tasks: SST-2, CoLA

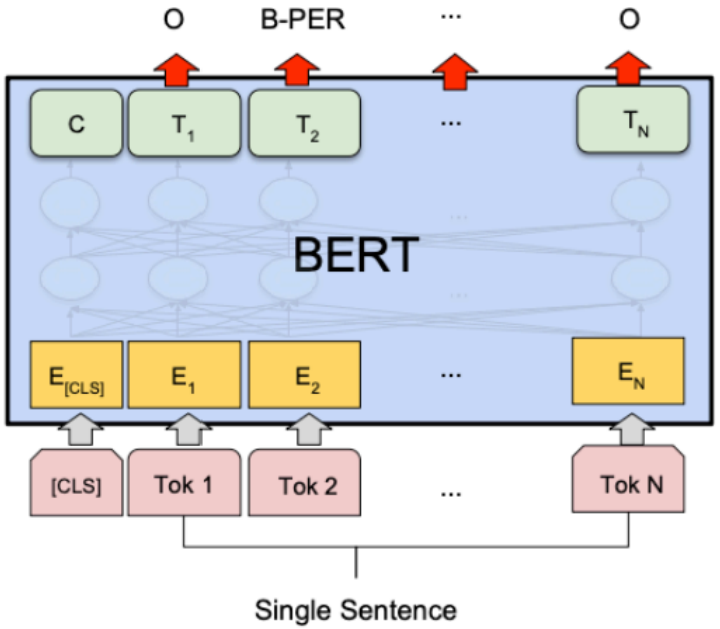
Pre-training and Fine-tuning

BERT pre-training

- Pre-train once, fine-tune many times (token-level tasks)



(c) Question Answering Tasks: SQuAD v1.1



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

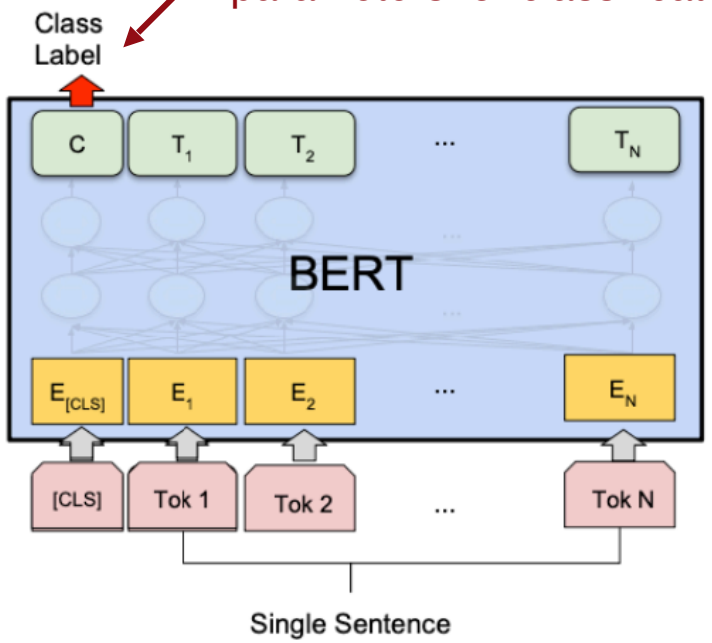
Pre-training and Fine-tuning

Example: sentiment classification

- $P(y = k) = softmax_k(\mathbf{W}_o \mathbf{h}_{[CLS]}),\ \mathbf{W}_o \in \mathbb{R}^{C \times h}$

- All parameters will be learned together
  - original BERT parameters + new classifier parameters

We just need to introduce $\mathscr{C} \times h$ parameters for classification tasks



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

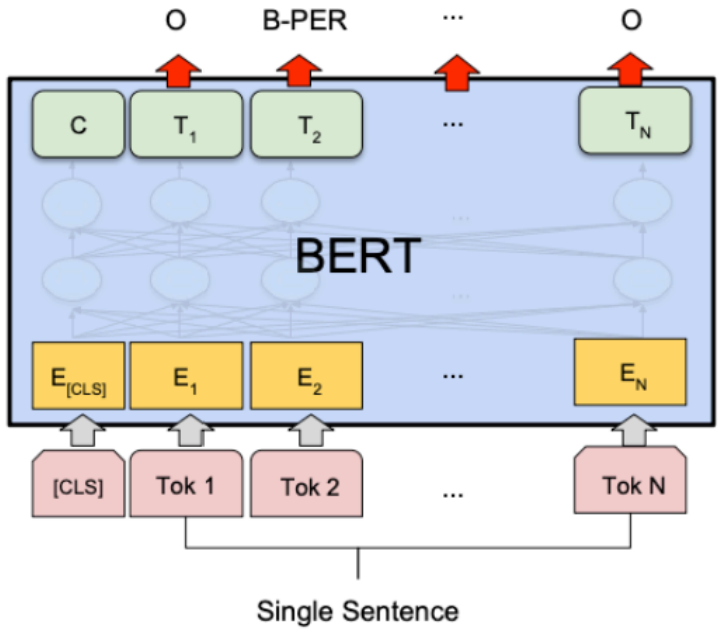
Pre-training and Fine-tuning

Example: named entity recognition (NER)

- $P(y_i = k) = softmax_k(\mathbf{W}_o \mathbf{h}_i),\ \mathbf{W}_o \in \mathbb{R}^{C \times h}$

- All parameters will be learned together
  - original BERT parameters + new classifier parameters

We just need to introduce $\mathscr{C} \times h$ parameters for classification tasks



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

Pre-training and Fine-tuning

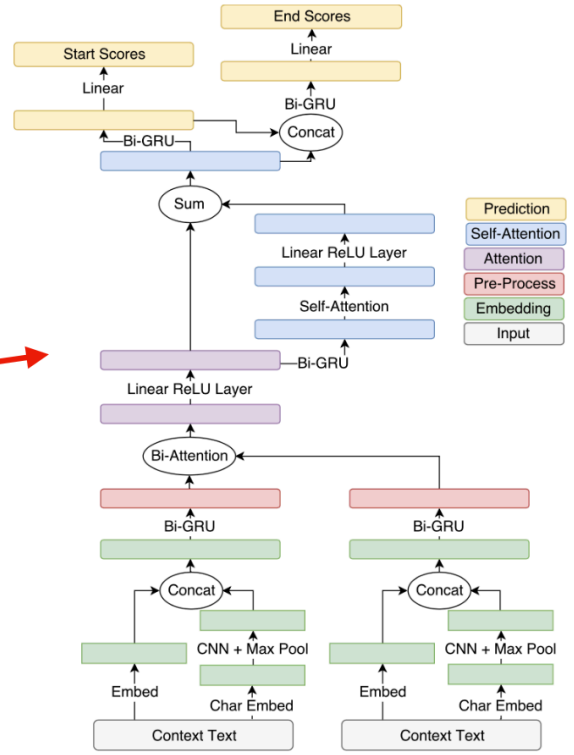
Experimental results: GLUE

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**

Pre-training and Fine-tuning

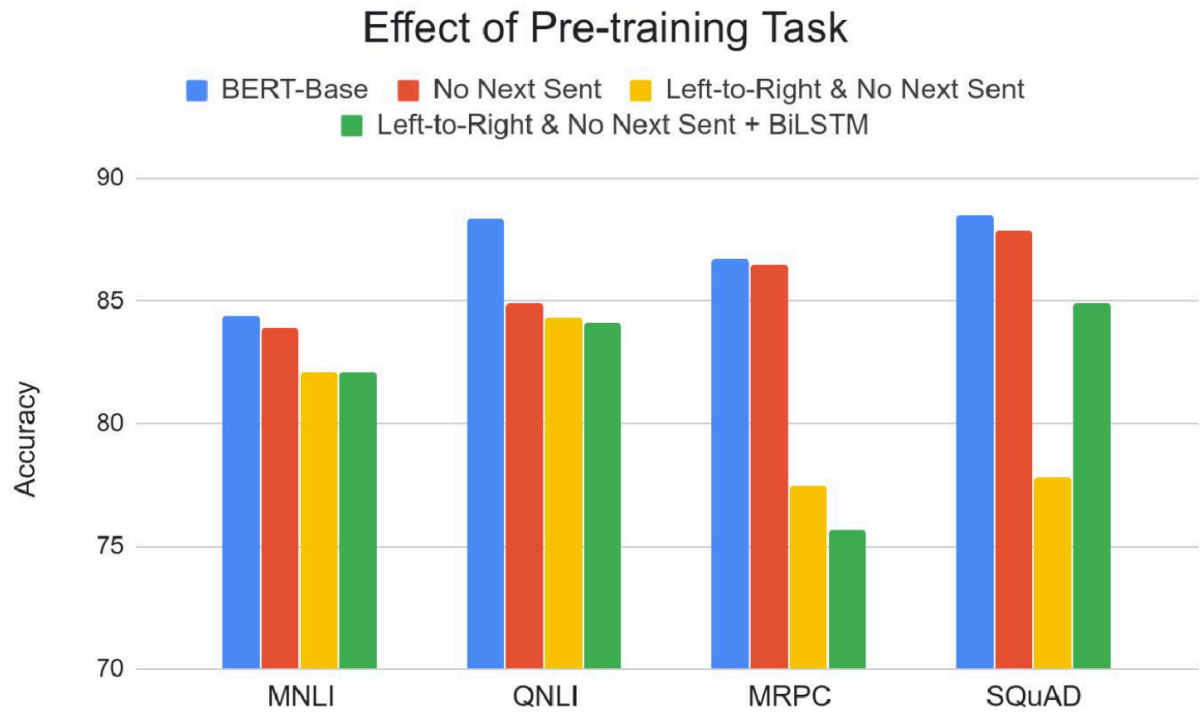
Experimental results: SQuAD

System	Dev EM	Dev F1	Test EM	Test 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				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**



Pre-training and Fine-tuning

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

Pre-training and Fine-tuning

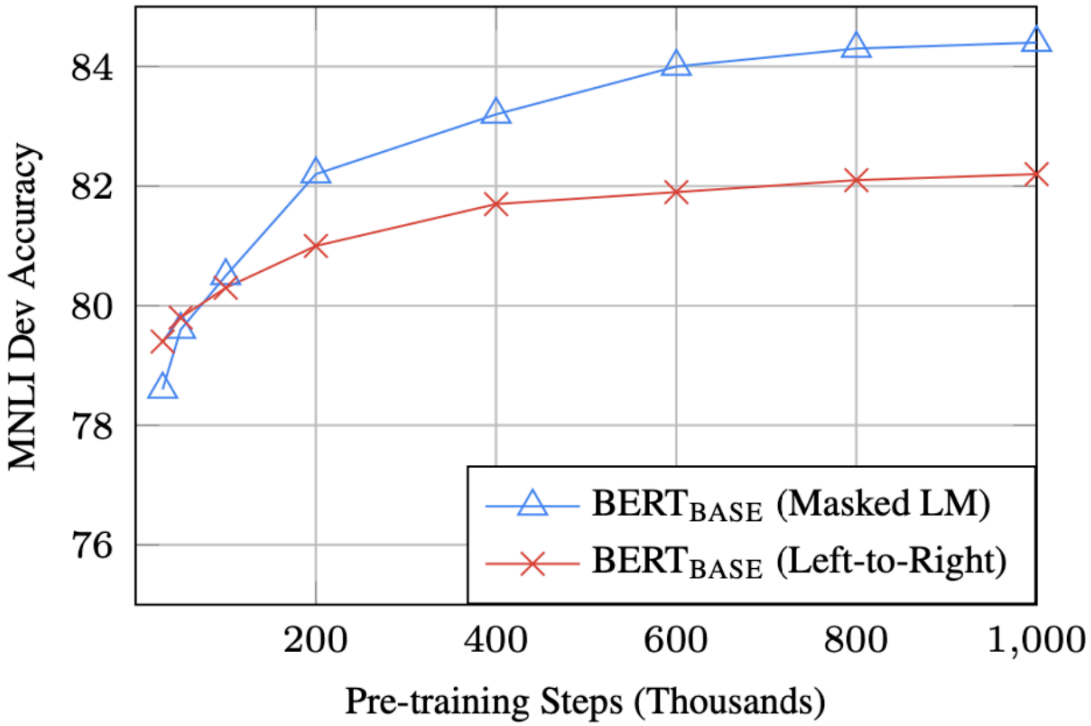
Ablation study: model sizes

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!

Pre-training and Fine-tuning

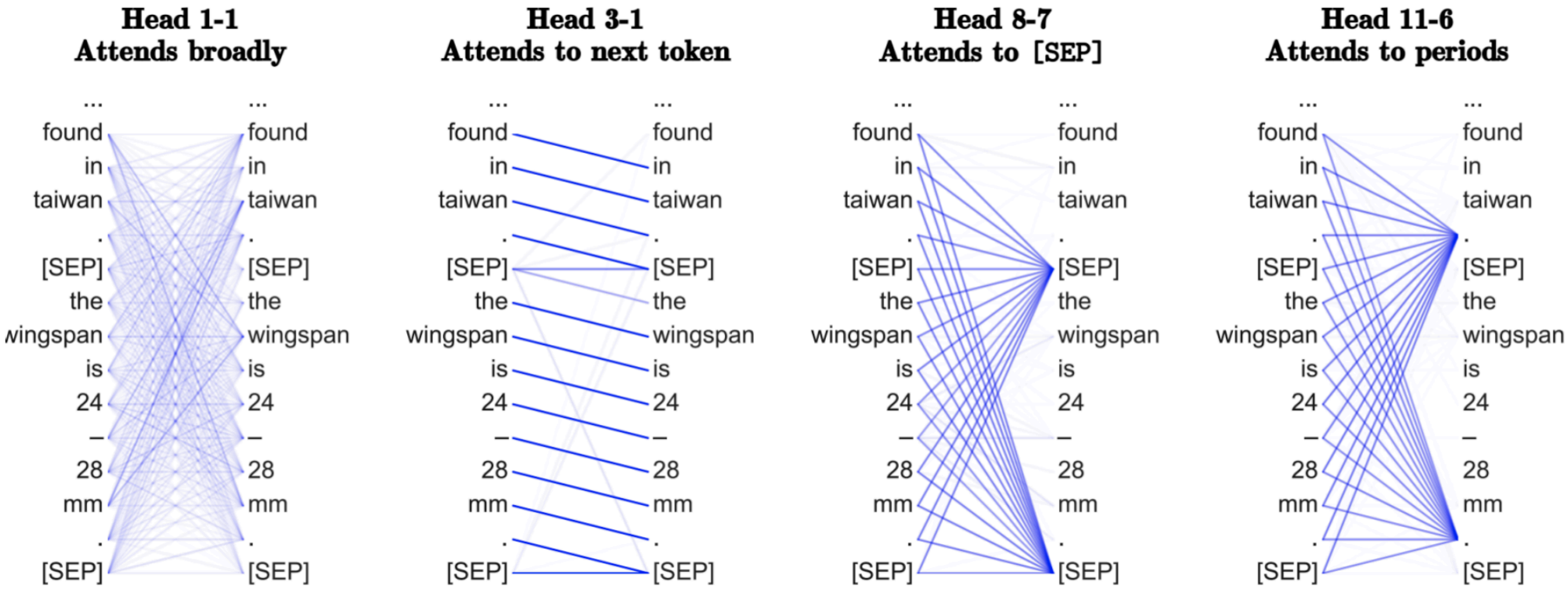
Ablation study: training efficiency



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

Pre-training and Fine-tuning

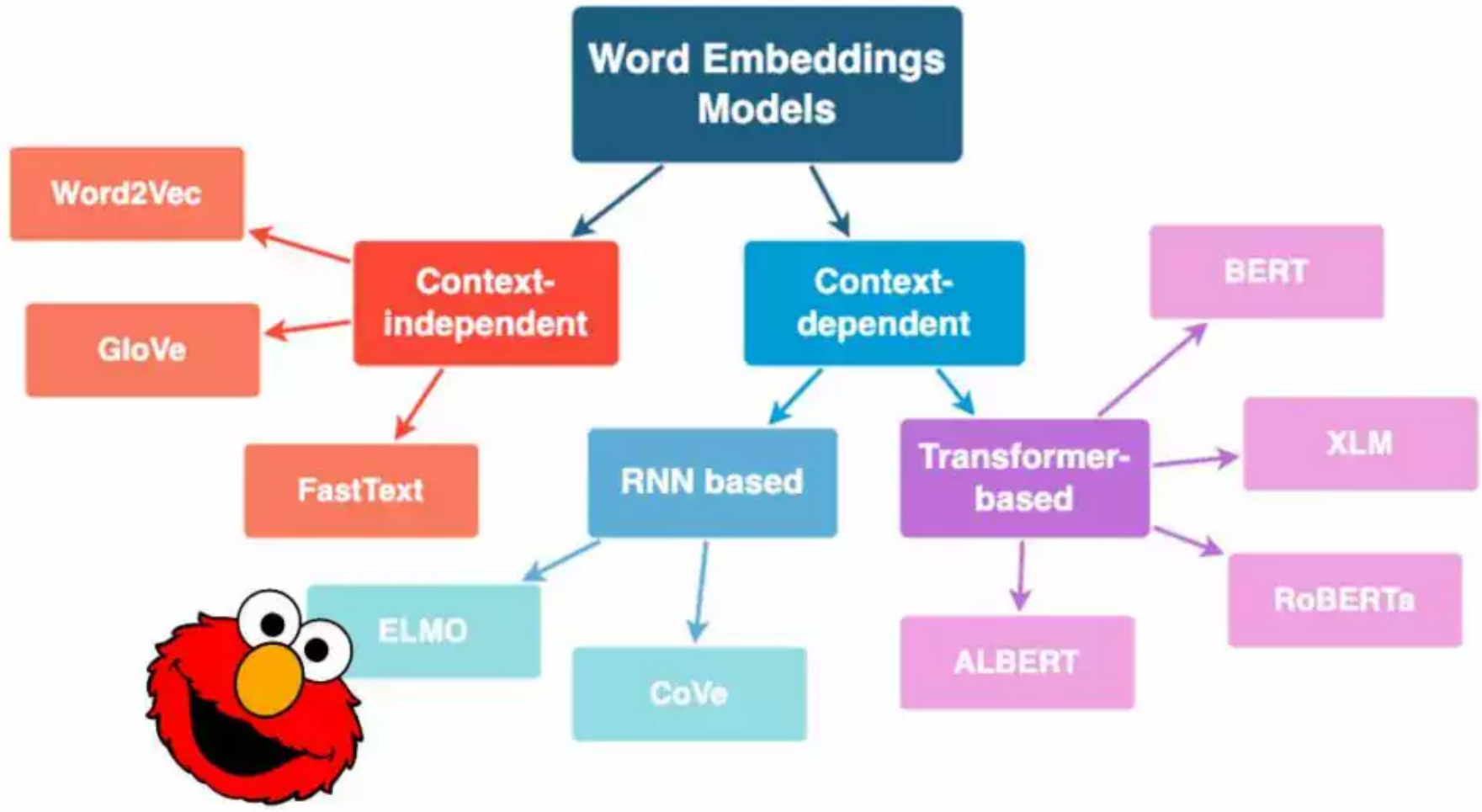
What does BERT learn?



Pre-training and Fine-tuning

ELMo vs GPT vs BERT

- Which of the following statements is INCORRECT?
  - (a) BERT was trained on more data than ELMo
  - (b) BERT builds on Transformer encoder, and GPT builds on Transformer decoder
  - (c) ELMo requires different model architectures for different tasks
  - (d) BERT was trained on data with longer contexts compared to GPT


Word Embeddings Models
Word2Vec
GloVe
Context-independent
FastText
Context-dependent
RNN based
ELMO
CoVe
Transformer-based
BERT
XLM
RoBERTa
ALBERT