BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova

Google AI Language

All4Nothing
sd081104@gmail.com

BERT

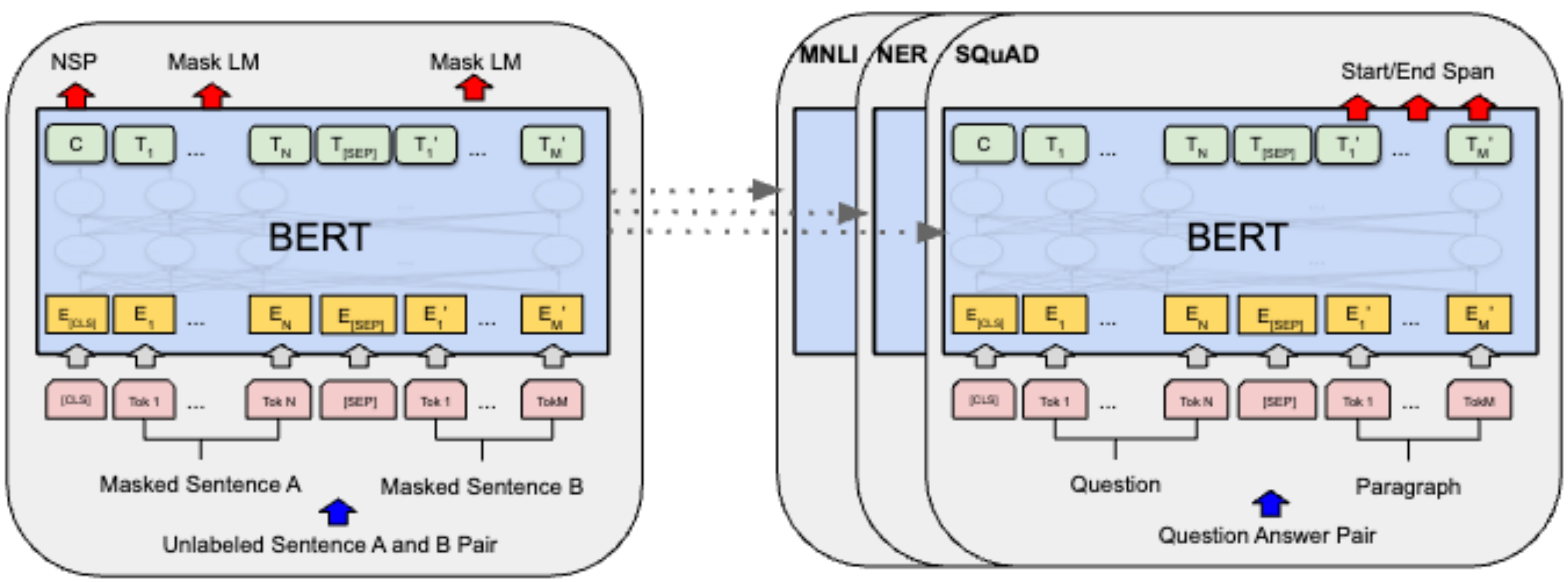
Bidirectional Encoder Representations from Transformers

- Designed to pre-train deep bidirectional representation from unlabeled text by jointly conditioning on both left and right context in all layers
- Masked Language Model (MLM)
- Next Sentence Prediction (NSP)
- Pre-trained BERT model can be fine-tunes with just one additional output layer to create SOTA models for a wide range of NLP tasks (QA, NER, Sentiment Analysis, etc.)

BERT

Background

- Motivation
  - Language models only use left context or right context, but language understand bi-directional



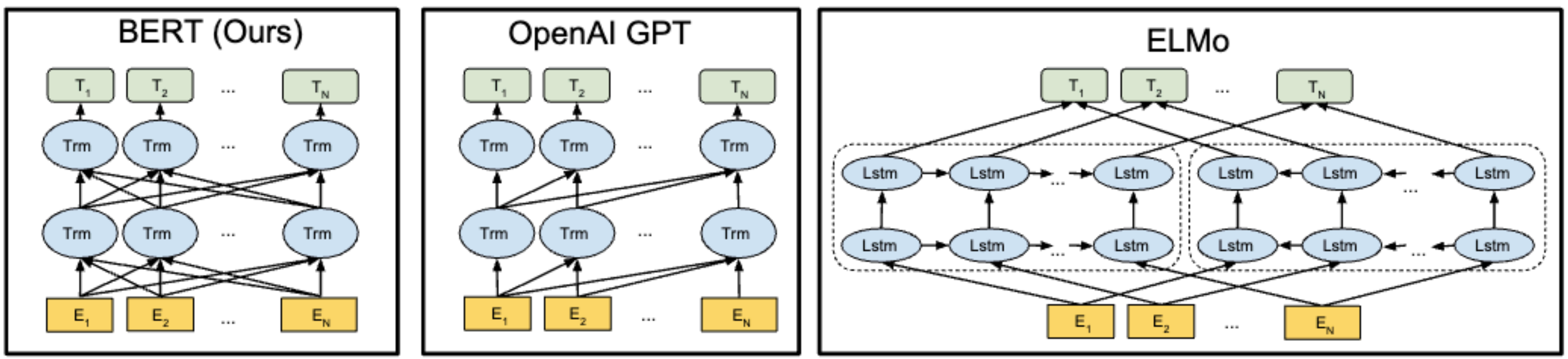
Pre-training

Fine-Tuning

BERT

Background

- Differences in pre-training model architectures

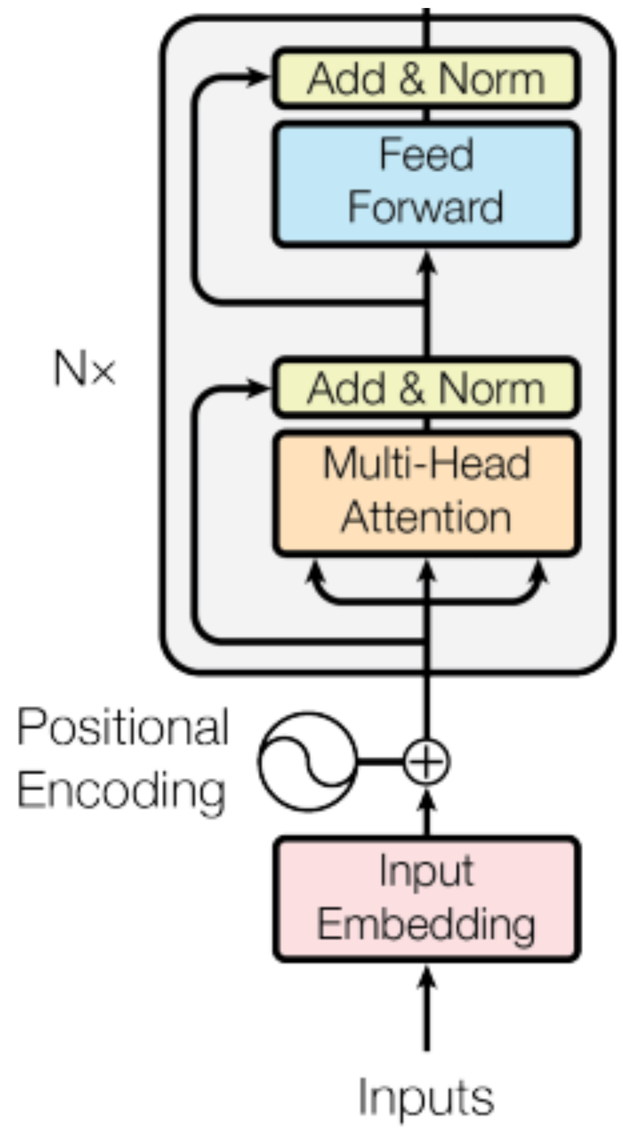


- BERT uses a bidirectional Transformer
- OpenAI GPT uses a left-to-right Transformer
- ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM

BERT

Model Architecture

- Multi-layer bidirectional Transformer Encoder
  - L: number of layers (Transformer block)
  - H: hidden size
  - A: number of self-attention heads
- BERT$_{\text{BASE}}$
  - L = 12, H = 768, A = 12
  - Total parameters = 110M
  - Same model size as OpenAI GPT
- BERT$_{\text{LARGE}}$
  - L = 24, H = 1024, A = 16
  - Total parameters = 340M

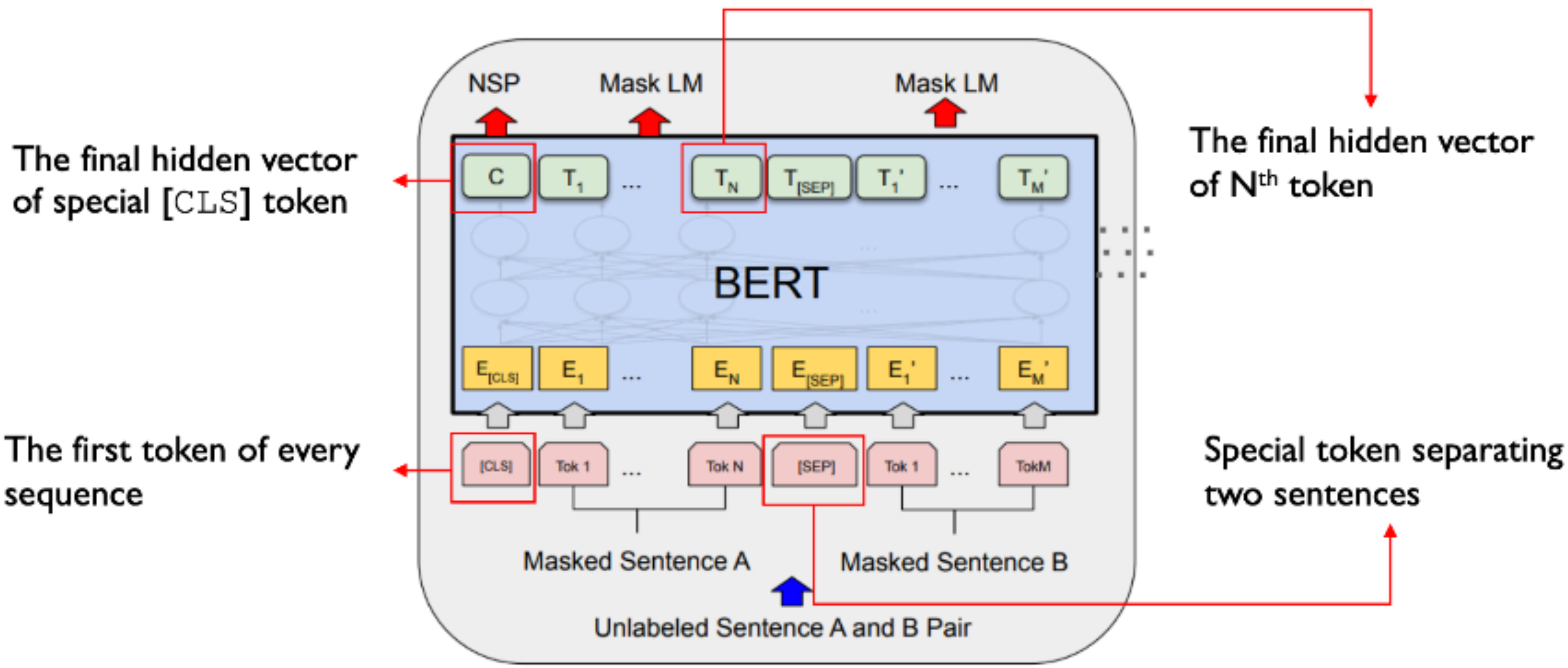


Transformer Encoder

BERT

Input/Output Representations

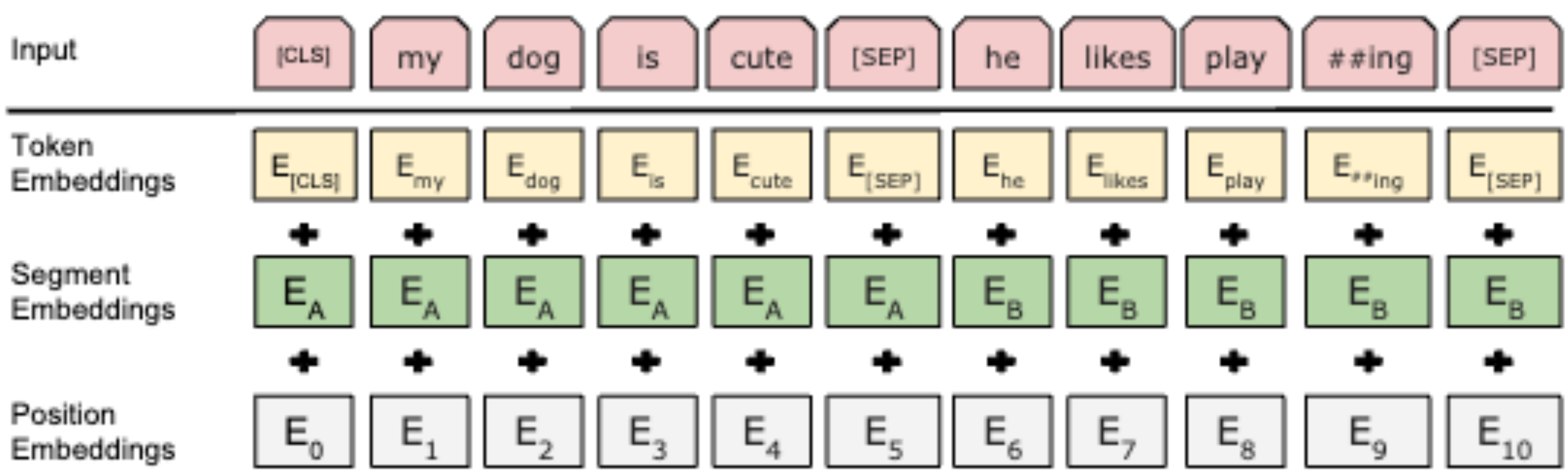
- To make BERT handle a variety of down-stream tasks, the input representation is able to unambiguously represent both a single sentence and a pair of sentences (ex: Question-Answer)



BERT

Input/Output Representations

- Input representation is the sum of
  - Token embedding: WordPiece embeddings with a 30,000 token vocabulary
  - Segment embedding
  - Position embedding: same as in the Transformer



BERT

Pre-training BERT

- Masked Language Model (MLM)
  - Mask some percentage of the input tokens at random, and then predict those masked tokens
    - 15% of each sequence are replaced with a [MASK] token
    - Predict the masked words

BERT

Pre-training BERT

- Problem
  - Mask token never seen during fine-tuning
- Solution
  - 15% of the words to predict, but don’t replace with [MASK] 100% of the time. Instead:
    - 80% of the time: Replace the word with the [MASK] token
      - my dog is hairy → my dog is [MASK]
    - 10% of the time: Replace the word with a random word
      - my dog is hairy → my dog is apple
    - 10% of the time: Keep the word unchanged
      - my dog is hairy → my dog is hairy

BERT

Pre-training BERT

- Ablation over different masking strategies

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI Fine-tune	NER Fine-tune	NER Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

BERT

Pre-training BERT

- Next Sentence Prediction (NSP)
  - Many important downstream tasks such as QA and NLI are based on understanding the relationship between two sentences, which is not directly captured by language modeling
  - Predict whether Sentence B is an actual sentence that proceeds Sentence A, or a random sentence

BERT

Pre-training BERT

- Next Sentence Prediction (NSP)
  - A Binarized next sentence prediction task that can be trivially generated from any monolingual corpus is trained
    - 50% of the time B is the actual next sentence that follows A (IsNext)
    - 50% of the time it is a random sentence from the corpus (NotNext)

BERT

Pre-training BERT

- Next Sentence Prediction (NSP)
  - Despite its simplicity, pre-training towards this task is very beneficial both QA and NLI

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

Label = `IsNext`

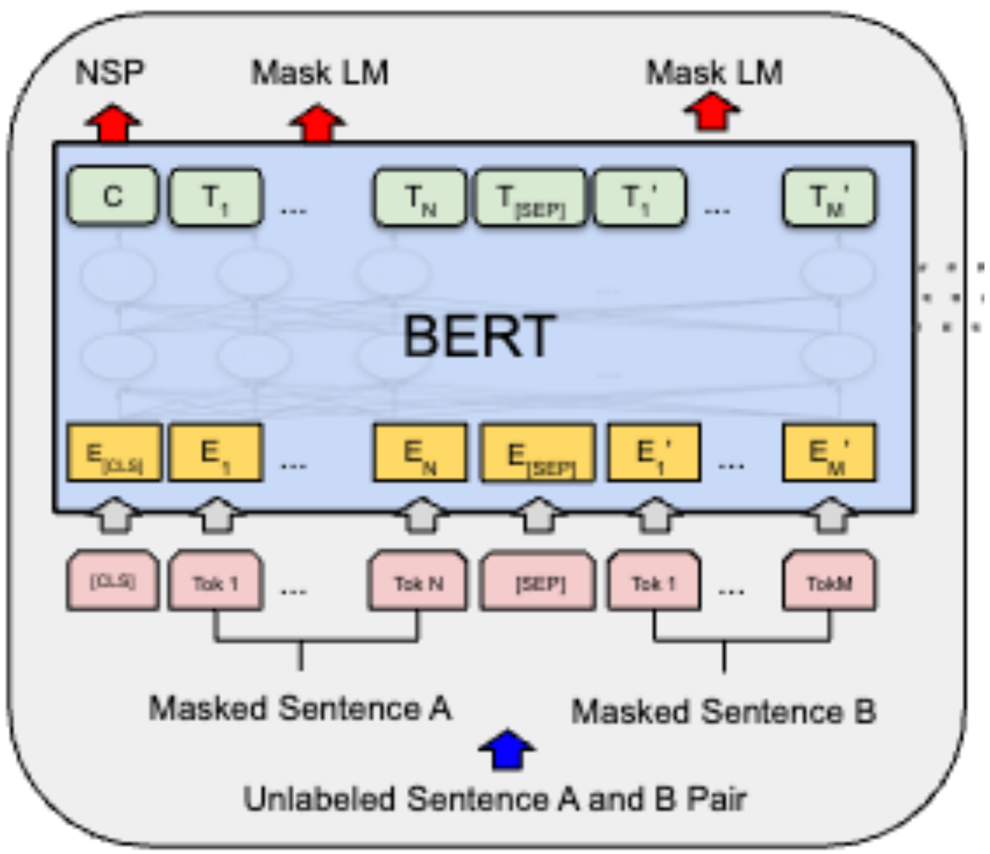
Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`

Label = `NotNext`

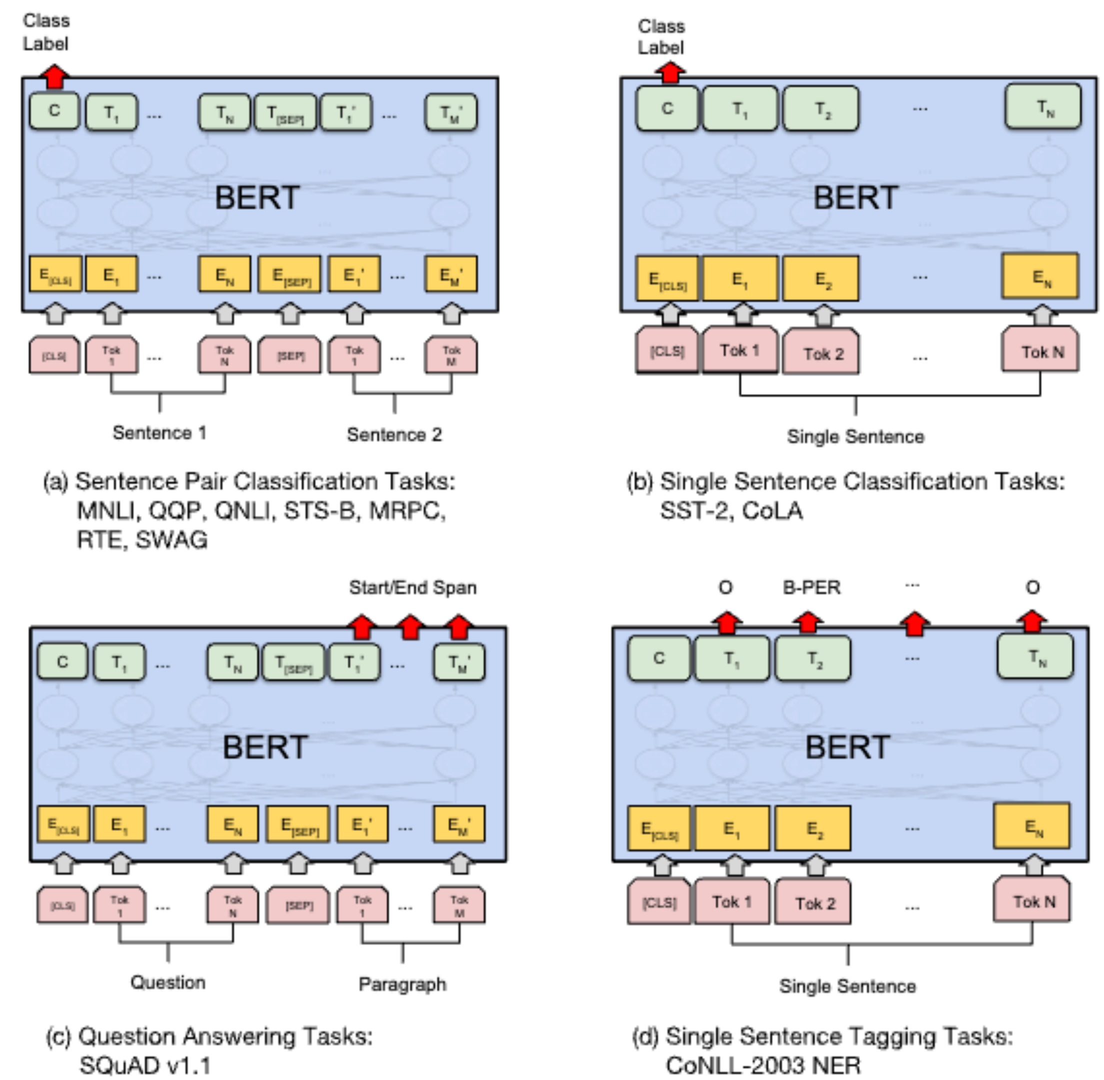
BERT

Fine-tuning

- Transfer Learning



Pre-training



BERT

BERT vs GPT-1

- Comparison of BERT and GPT-1
  - Training-data size
    - GPT is trained on BookCorpus(800M words), BERT is trained on the BookCorpus and Wikipedia(2,500M words)
  - Training special tokens during training
    - BERT learns [SEP], [CLS] and sentence A/B embedding during pre-training
  - Task-specific fine-tuning
    - GPT uses the same learning rate of 5e-5 for all fine-tuning experiments, BERT chooses a task-specific fine-tuning learning rate

BERT

Results

- GLUE Benchmark Results

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**

BERT

Ablation Studies

- Effect of Pre-training Tasks

Tasks	Dev Set MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
$\text{BERT}_{\text{BASE}}$	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

BERT

Ablation Studies

- Effect of Model Size

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

BERT

Ablation Studies

- Feature-based Approach with BERT
  - Not all tasks can be easily represented by a Transformer encoder architecture
    → Require task-specific model architecture
  - Computational benefits to pre-compute an expensive representation of the training data

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$BERT_{LARGE}$	96.6	92.8
$BERT_{BASE}$	96.4	92.4
Feature-based approach ($BERT_{BASE}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

BERT

Any Questions?