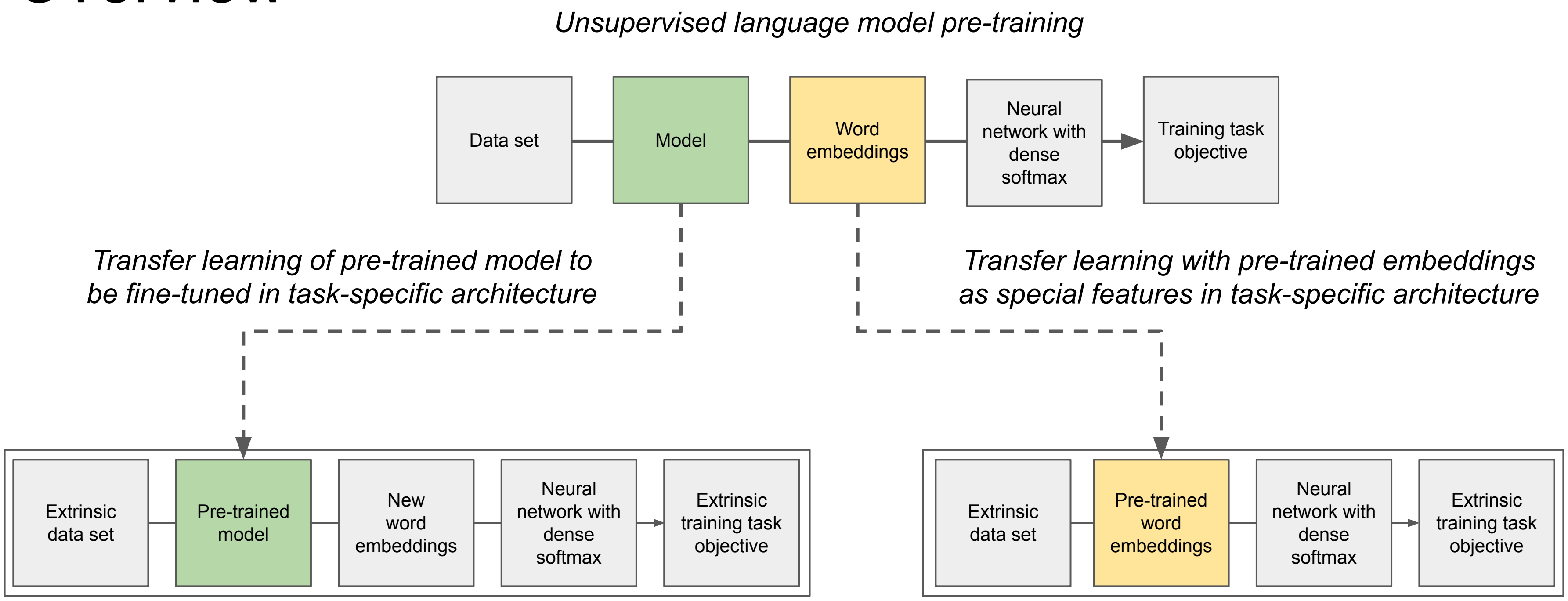
Presentation on “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”

LING 575 F/G, Spring 2019: Text Representation Learning
Assignment 5
May 20, 2019

Thomas Phan and Avijit Vajpayee

Overview



- OpenAI GPT
- Google BERT

- Brown Clusters, Word2Vec, GloVe (Word Level)
- Skip-thought, Doc2Vec (Sentence / Doc Level)
- ELMo: context-sensitive word embeddings from LM

BERT

- Problem:
  - Unigram LM useful for only generation purpose
  - Generated embeddings only represent one meaning context
  - Need to take both previous and later word tokens <u>at the same time</u>

The dogs bark at noon.

The tree bark is brittle.

Unigram language model generates only one embedding of *bark* for both meanings.

Bidirectional language model looks at left and right of *bark* and generates different embeddings for it given the sentence.

BERT

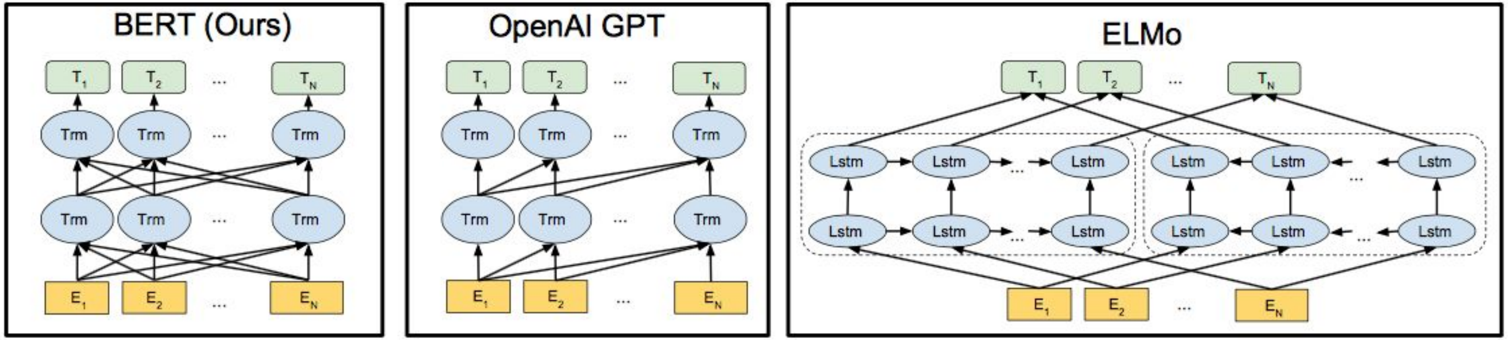


Figure 1: Differences in pre-training model architectures. BERT uses a bidirectional Transformer. OpenAI GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTM to generate features for downstream tasks. Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

- Previous approaches:
  - *OpenAI GPT*: Masked transformer architecture for Left to Right LM
  - *ELMo*: Shallow independently-trained bidirectional LSTM
- Authors try out:
  - Bidirectional Transformer LM to force learning how to use entire sequence
  - New Training Objectives: MLM (Masked Language Modelling), Next Sentence Prediction

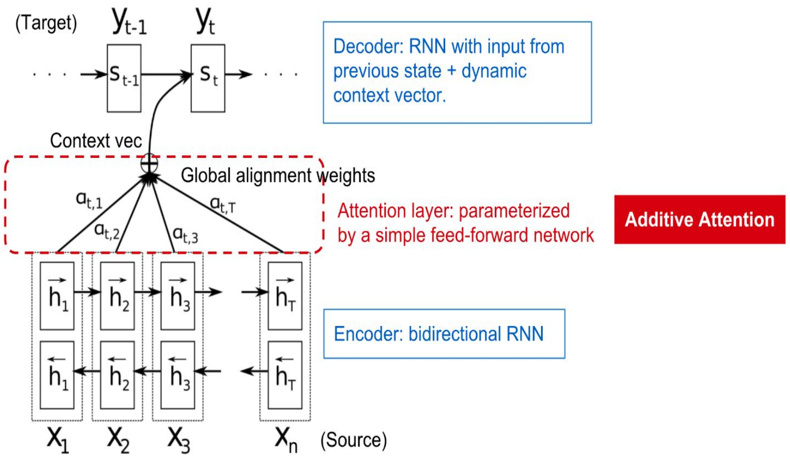
Seq2Seq

- RNN Encoder-Decoder Architectures
  - **Issues:**
    - Sequential nature. Harder to parallelize
    - Long Range Dependencies (Gated Architectures like LSTM/GRU help)
    - Cramming meaning of whole sentence into single vector

- Transformer Architecture ("Attention is all you need" - Vaswani et. al. 2017)
  - Seq2Seq with NO RECURRENT CONNECTIONS
    - Very fast on GPU
  - Use multi-headed self-attention and encoder-decoder attention mechanism
  - NOT a single sweep of attention

Attention

- Broadly, vector of importance weights
  - Give decoder access to all of encoder's states
  - Decoder "chooses" which hidden state to use and which to ignore by weights
- [Bahdanau et. al. 2015]
  - NMT
  - Encoder Hidden State : h_i
  - Decoder Hidden State : s_t
    - Function of s_{t-1} and c_t
  - Dynamic Context Vector : c_t
    - "Additive Attention" $c_t = \sum_i^n \alpha_{t,i} h_i$
  - Captures Input-Output Dependencies



Self-Attention

- Self-Attention (*Intra-Attention*) :
  - Relating different positions of a single sequence in order to compute a representation of the same sequence
  - Capture within-input and within-output dependencies
  - For e.g. The pen is mightier than the sword

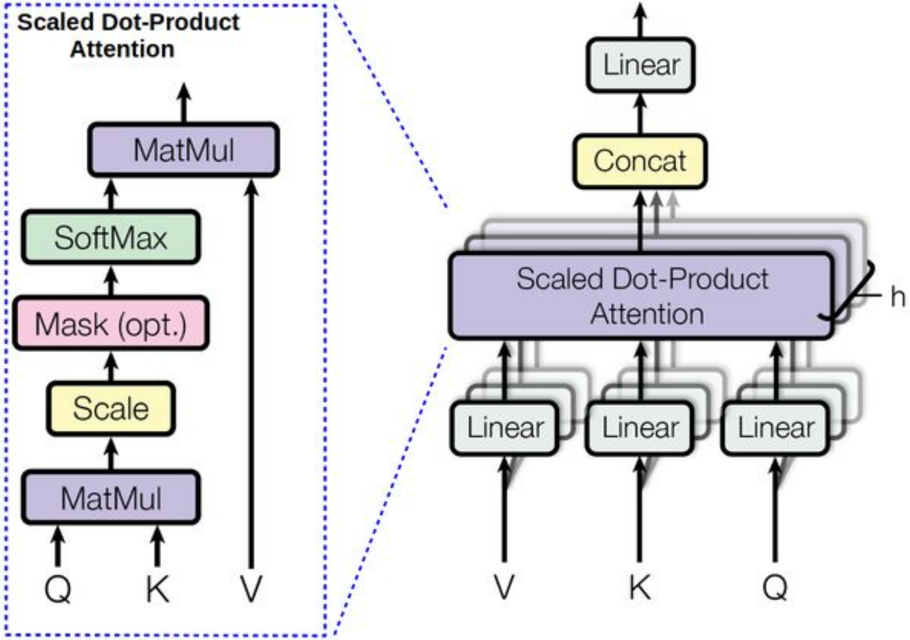
- Key-Value Attention :
  - Separate form from function
  - Split to 3 Vectors : Key (K), Value (V), Query (Q)
  - For Self-Attention : K, V, Q all linear transformations of input embedding
  - For Encoder-Decoder Attention
    - K, V from Encoder Outputs
    - Q from target embedding

Multi-Head Attention

- Multiple heads in parallel

- *Intuition* - Ensembling

- Independent attention outputs concatenated and linearly transformed



Transformer

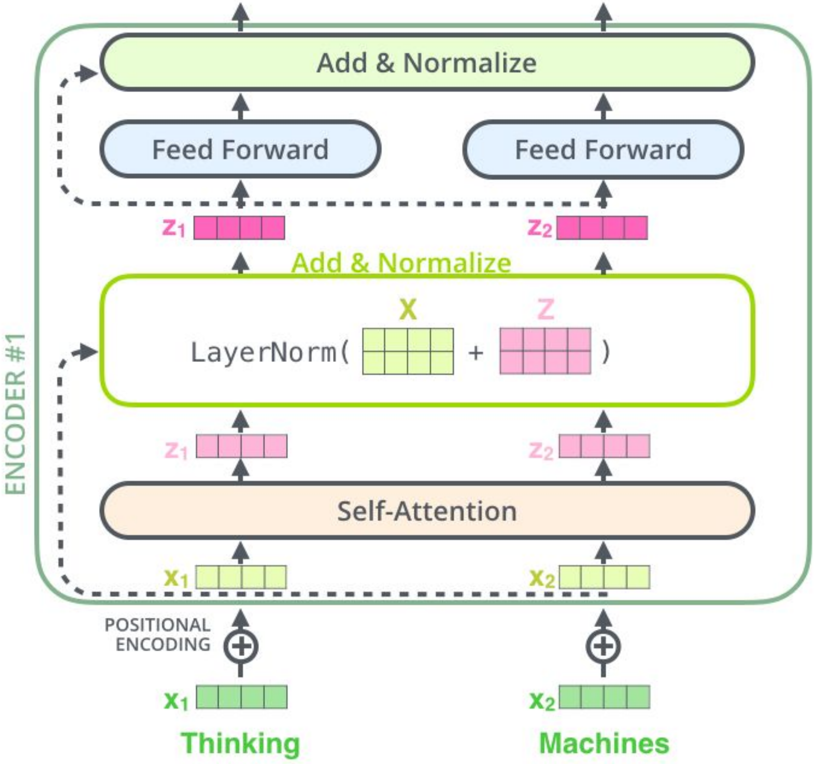
- Encoder-Decoder Architecture
    - Without Recurrent Connections
    - Multi-Head Scaled Dot Product Self-Attention
    - Tries to capture 3 types of dependencies
        - Input to Output
        - Within Input
        - Within Output
- As no sequence, CANNOT naturally make use of position of words in sequence
    - Augment word embeddings with positional embeddings

Encoder

- Multiple stacked Single Encoder blocks (6 single blocks)

- Single Block 2 sublayers
  - Multi-Head Self Attention
  - Feedforward Network

- Optimization Tricks
  - Layer Normalization
  - Residual Connections

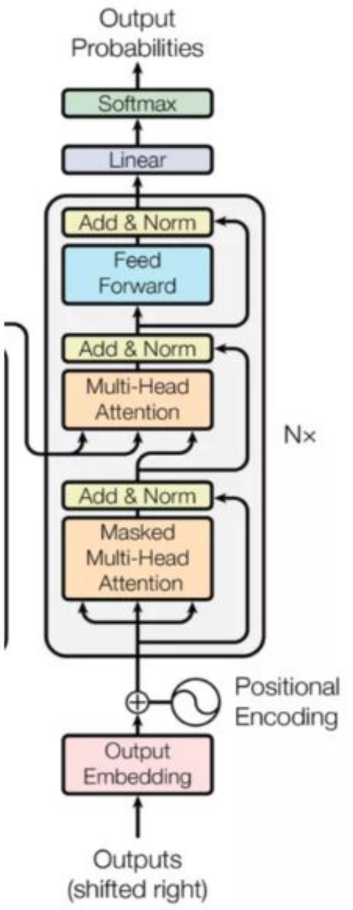


Decoder

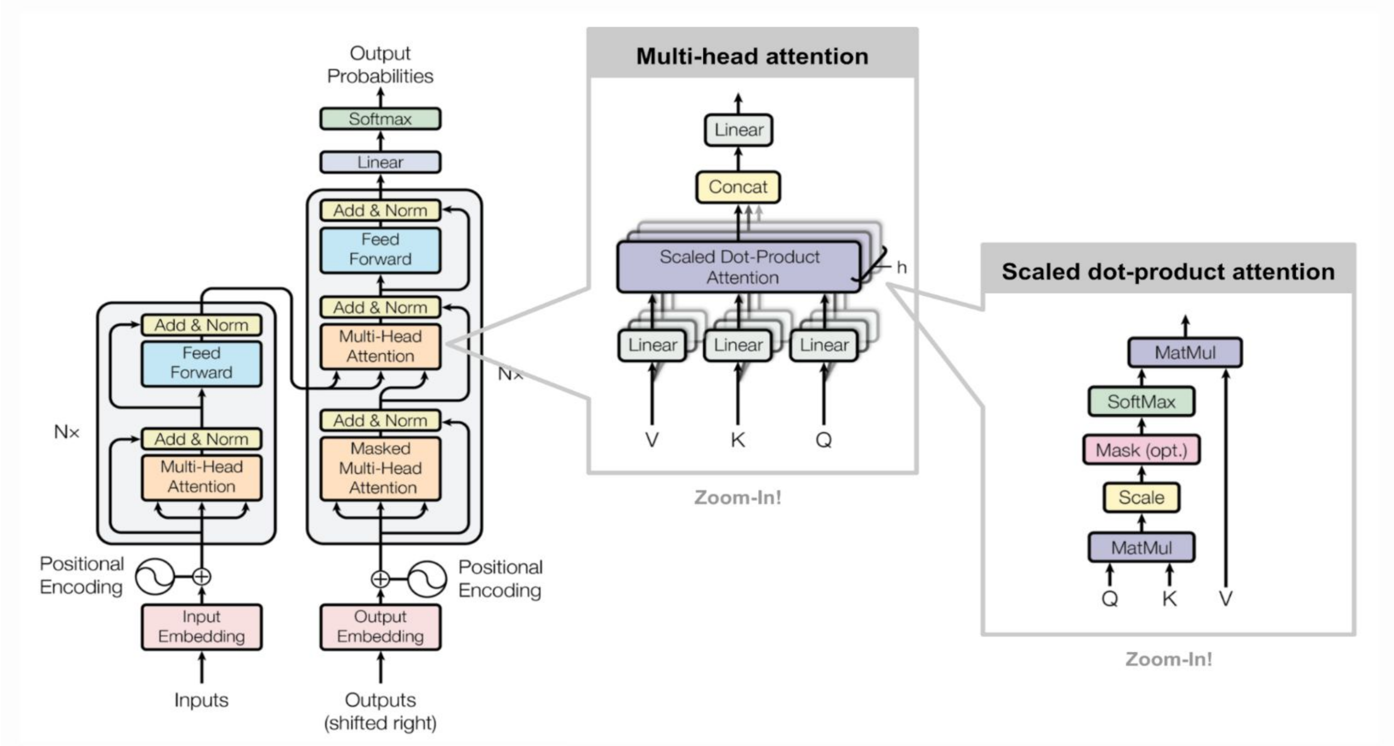
- Multiple stacked single Decoder blocks (6 single blocks)

- Single Decoder Block 3 sub-layers:
  - Previous Outputs Self-Attention (masked)
  - Encoder-Decoder Attention
  - Feed-forward network

- Optimization Tricks -
  - Layer Normalization
  - Residual Connections



Putting it all together



References

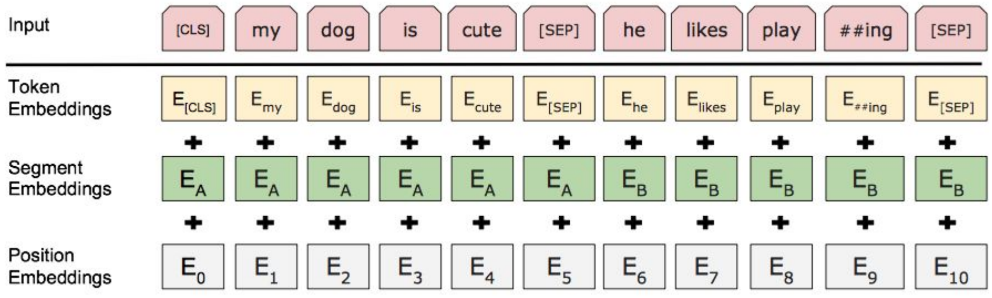
- Excellent blog with visualization of Transformer
- Blog on various attention mechanisms in NLP
- Deep Learning for NLP Best Practices

1. Overview
2. Transformer Architecture
3. Pre-Training
4. Fine-Tuning
5. Performance
6. Discussion

Input Representations

Can unambiguously represent single or pair of sentences as sequence

- Token Embeddings from WordPiece
    - Reduces vocab size and data sparsity
    - Smallest units that carry meaning
- Segment Embeddings
    - When using sentence pairs as sequence
- Positional Embeddings



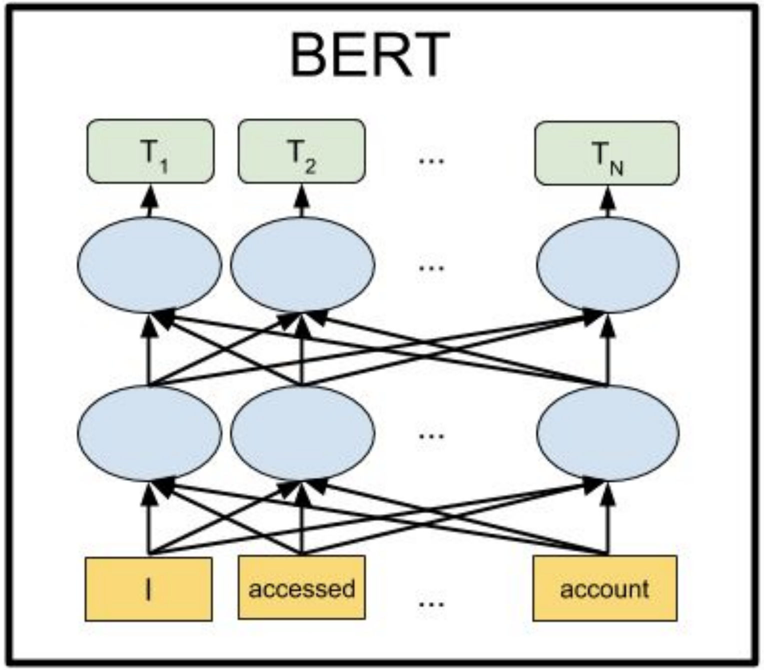
- Prepend each sequence with special [CLS] token
    - Used as final representation of sequence when fine-tuning for classification tasks

Architecture

- Multiple stacked layers of Transformer Encoder

3 Hyper-parameters

1. L = Number of Layers (12) (Transformer Encoder Blocks)
2. H = Hidden Size (768)
3. A = Attention Heads (12)
   a. Feedforward Filter Size = 4 A (48)



Pre-Training

2 objectives -

1. Masked LM
    a. Instead of left to right prediction, randomly mask words in sequence and use both context (left, right) to jointly predict masked
2. Next Sentence Prediction
    a. Important for tasks that require learning relation between two sentences (NLI, QA)
    b. Creating a sequence of two sentences
        i. 50% chance second sentence is next sentence
        ii. 50% chance randomly sampled
    c. Binary Classification - *next / notNext*

Data :

Book Corpus + English Wikipedia (3.3 Billion Tokens)

Training Loss :

Mean masked LM likelihood + Mean next sentence prediction likelihood

Training Time :

4 Days (Multiple cloud TPUs)

1. Overview
2. Transformer Architecture
3. Pre-Training
4. Fine-Tuning
5. Performance
6. Discussion

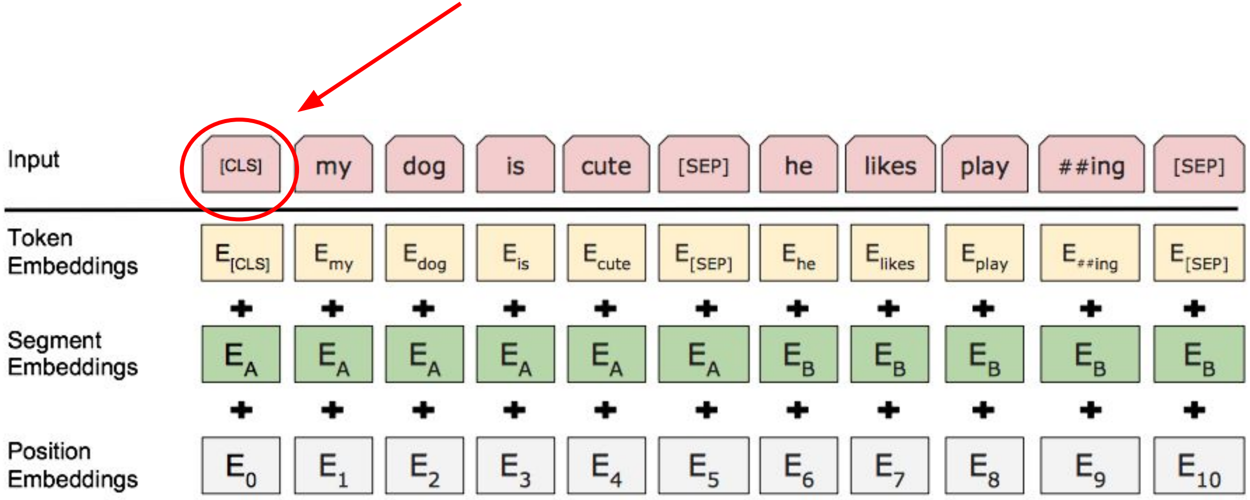
Fine-tuning

- Model is fine-tuned to downstream extrinsic tasks

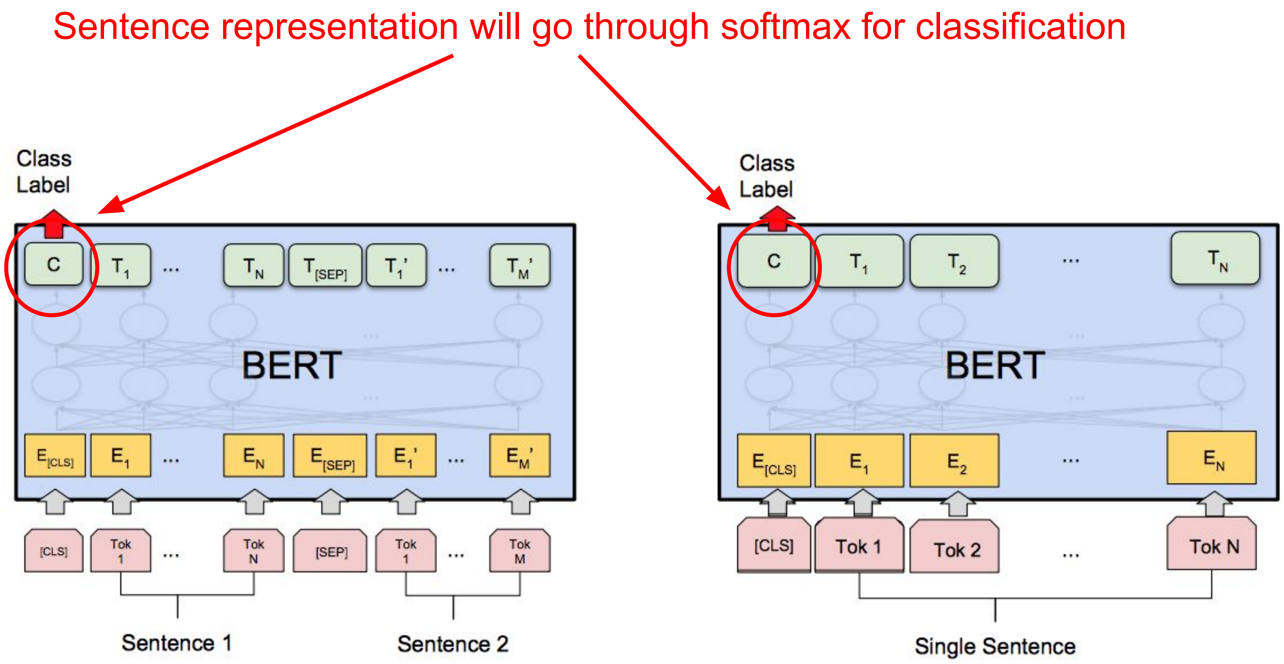
Benchmark / suite	Name	Task type
GLUE (General Language Understanding Evaluation) benchmark suite	MNLI (Multi-Genre Natural Language Inference)	Natural language inferencing
	QQP (Quora Question Pairs	Semantic equivalence classification
	QNLI (Question Natural Language Inference)	Natural language inferencing
	SST-2 (Stanford Sentiment Treebank)	Sentiment analysis classification
	CoLA (Corpus of Linguistic Acceptability)	Sentence classification
	STS-B (Semantic Textual Similarity Benchmark)	Sentence similarity
	MRPC (Microsoft Research Paraphrase Corpus)	Paraphrase classification
	RTE (Recognizing Textual Entailment)	Natural language inferencing
	WNLI (Winograd NLI)	Natural language inferencing
SQuAD v1.1	Stanford Question-Answering Dataset	Question answering using Wikipedia
CoNLL 2003 Shared Task	Conference on Computational Natural Language Learning 2003	Named entity recognition
SWAG	Situations With Adversarial Generations	Sentence pair inference

Fine-tuning

- Model is fine-tuned to downstream extrinsic tasks
- Classification is performed on (1) entire sentence or on (2) tokens
- Entire sentence is represented by final hidden state vector from the Transformer output of the [CLS] token (first token of a sentence)



Fine-tuning

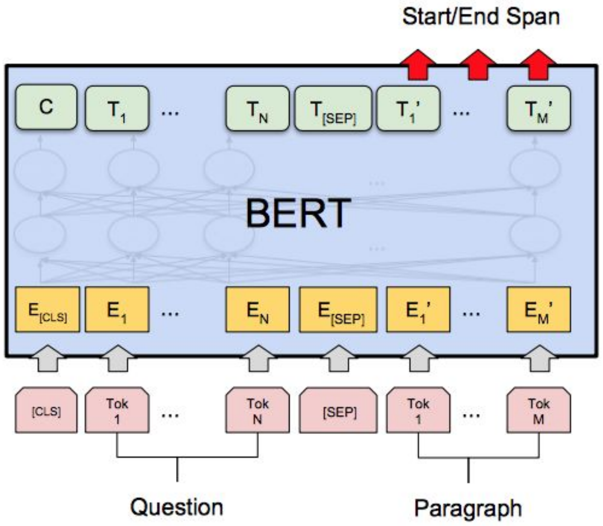


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

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

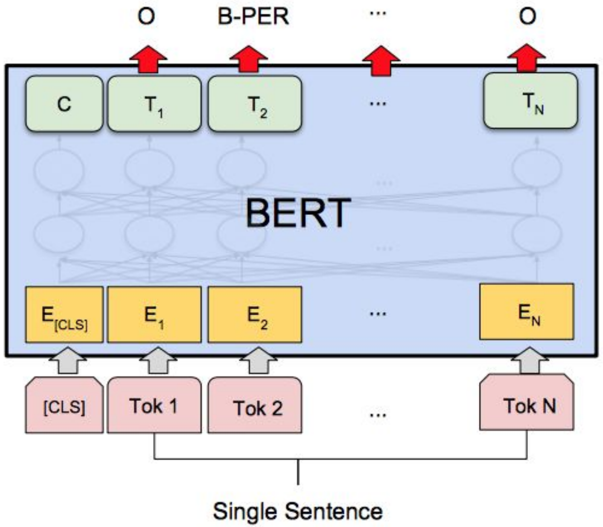
Fine-tuning

Classify each token as start/end of Q or A span



(c) Question Answering Tasks:
SQuAD v1.1

Classify each token as one of the NER tags



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

Performance - GLUE benchmark suite

OpenAI GPT and BERT-BASE have same configuration

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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Table 1: GLUE Test results, scored by the GLUE evaluation server. The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set. OpenAI GPT = (L=12, H=768, A=12); $\text{BERT}_{\text{BASE}}$ = (L=12, H=768, A=12); $\text{BERT}_{\text{LARGE}}$ = (L=24, H=1024, A=16). BERT and OpenAI GPT are single-model, single task. All results obtained from https://gluebenchmark.com/leaderboard and https://blog.openai.com/language-unsupervised/.

Performance

Question-Answering

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
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**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Named Entity Recognition

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

Table 3: CoNLL-2003 Named Entity Recognition results. The hyperparameters were selected using the Dev set, and the reported Dev and Test scores are averaged over 5 random restarts using those hyperparameters.

Aside: best Test F1 performance from **original CoNLL 2003 shared task competition** was 88.7 using combination of MaxEnt and other algorithms

Sentence pair inference

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. Test results were scored against the hidden labels by the SWAG authors. †Human performance is measure with 100 samples, as reported in the SWAG paper.

Better than single human expert!

Performance - Generate embeddings w/o fine-tuning

How to generate embeddings without fine-tuning?

Concatenating last four hidden layers performs best for CoNLL 2003 NER

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

Table 7: Ablation using BERT with a feature-based approach on CoNLL-2003 NER. The activations from the specified layers are combined and fed into a two-layer BiLSTM, without backpropagation to BERT.

1. Overview
2. Transformer Architecture
3. Pre-Training
4. Fine-Tuning
5. Performance
6. Discussion

Design Decisions

BERT design decision	Benefit
Language model pre-training	Transfer learning of model.
Transformer architecture	Seq2Seq without recurrent cells. Easier to parallelize.
Bidirectional transformer	Forces model to use entire sequence, taking more context into account.
Masked language model	Allows transformer to be trained bidirectionally.
Next sentence prediction	Essential for tasks which involve relation between 2 sentences

Discussion

- Is Masked LM more effective than sequential LM ? Yes

- Is Next Sentence Prediction necessary ? Yes (for a subset)

- Does a larger model always help ? Yes

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

Inference Tasks

NSP does NOT help much

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

Masked LM vs. Sequential LM

- Masked LM: only predict a subset of tokens (15%)
- Sequential LM: predict all tokens

Masked LM converges slower but better accuracy with same number of steps.