al+ AI Seminar No. 7
2018/12/21

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

Devlin et al., 2018 (Google AI Language)

Presenter
Phạm Quang Nhật Minh
NLP Researcher
Alt Vietnam

Outline

- Research context
- Main ideas
- BERT
- Experiments
- Conclusions

Research context

- Language model pre-training has been used to improve many NLP tasks
  - ELMo (Peters et al., 2018)
  - OpenAI GPT (Radford et al., 2018)
  - ULMFit (Howard and Rudder, 2018)
- Two existing strategies for applying pre-trained language representations to downstream tasks
  - *Feature-based*: include pre-trained representations as additional features (e.g., ELMo)
  - *Fine-tunning*: introduce task-specific parameters and fine-tune the pre-trained parameters (e.g., OpenAI GPT, ULMFit)

Limitations of current techniques

- Language models in pre-training are unidirectional, they restrict the power of the pre-trained representations
  - OpenAI GPT used left-to-right architecture
  - ELMo concatenates *forward* and *backward* language models
- **Solution** BERT: Bidirectional Encoder Representations from Transformers

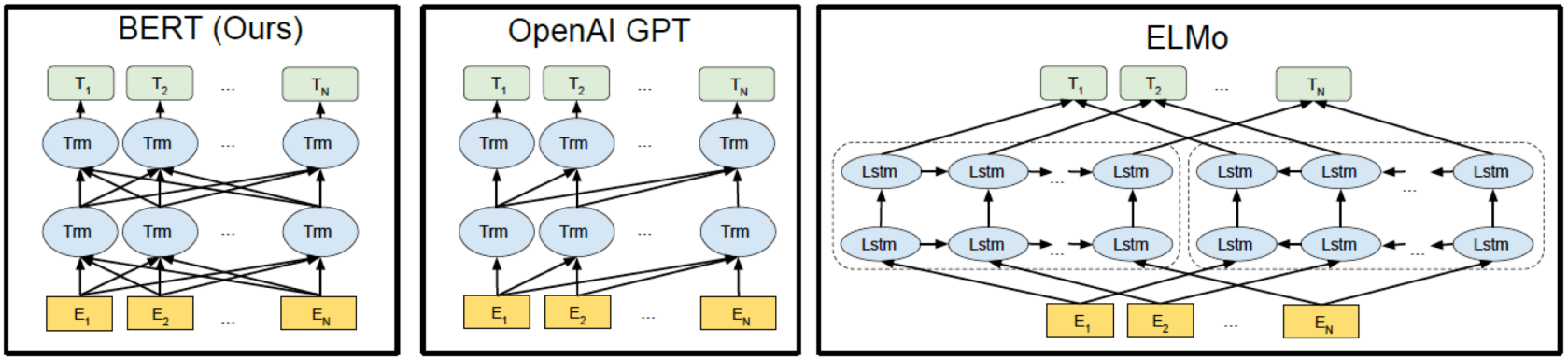
BERT: Bidirectional Encoder Representations from Transformers

- Main ideas
  - Propose a new pre-training objective so that a deep bidirectional Transformer can be trained
    - **The "masked language model" (MLM)**: the objective is to predict the original word of a masked word based only on its context
    - "**Next sentence prediction**"
- Merits of BERT
  - Just fine-tune BERT model for specific tasks to achieve state-of-the-art performance
  - BERT advances the state-of-the-art for eleven NLP tasks

BERT: Bidirectional Encoder Representations from Transformers

Model architecture

- BERT's model architecture is a multi-layer **bidirectional Transformer encoder**
  - (Vaswani et al., 2017) "Attention is all you need"
- Two models with different sizes were investigated
  - $\text{BERT}_{\text{BASE}}$: L=12, H=768, A=12, Total Parameters=110M
    - (*L*: number of layers (Transformer blocks), *H* is the hidden size, *A*: the number of self-attention heads)
  - $\text{BERT}_{\text{LARGE}}$: L=24, H=1024, A=16, Total Parameters=340M

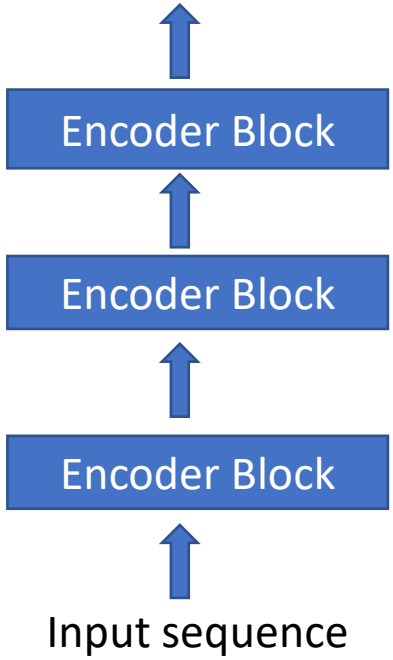


Differences in pre-training model architectures: BERT, OpenAI GPT, and ELMo

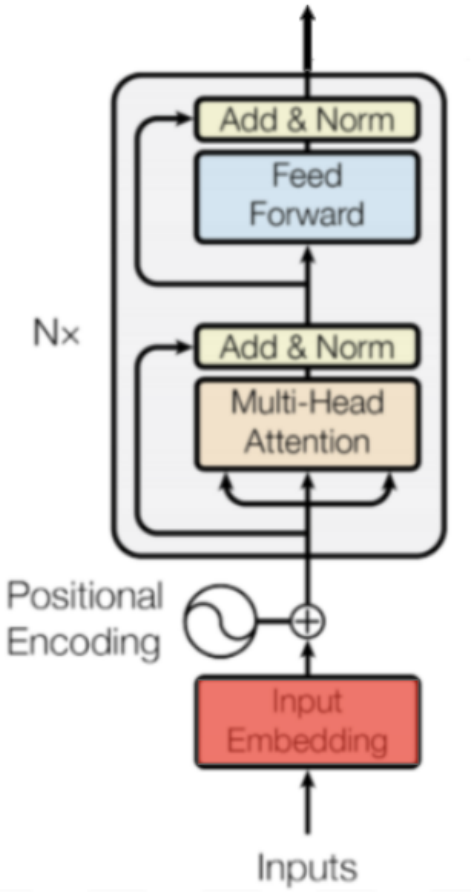
Transformer Encoders

Vaswani et al. (2017) *Attention is all you need*

- Transformer is an attention-based architecture for NLP
- Transformer composed of two parts: **Encoding** component and **Decoding** component
- BERT is a multi-layer bidirectional Transformer **encoder**



Inside an Encoder Block

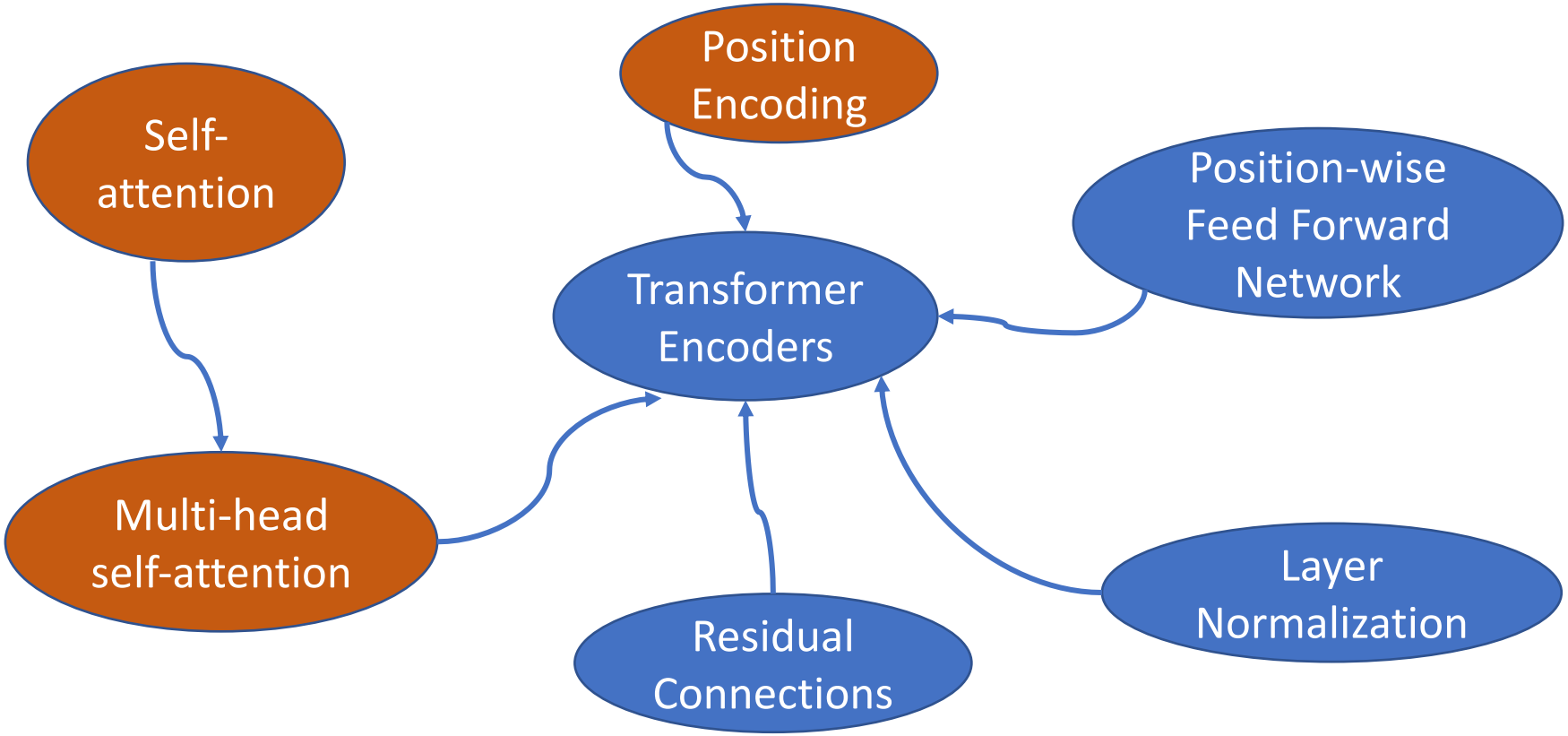


In BERT experiments, the number of blocks N was chosen to be 12 and 24.

Blocks do not share weights with each other

Source: https://medium.com/dissecting-bert/dissecting-bert-part-1-d3c3d495cdb3

Transformer Encoders: Key Concepts



Self-Attention

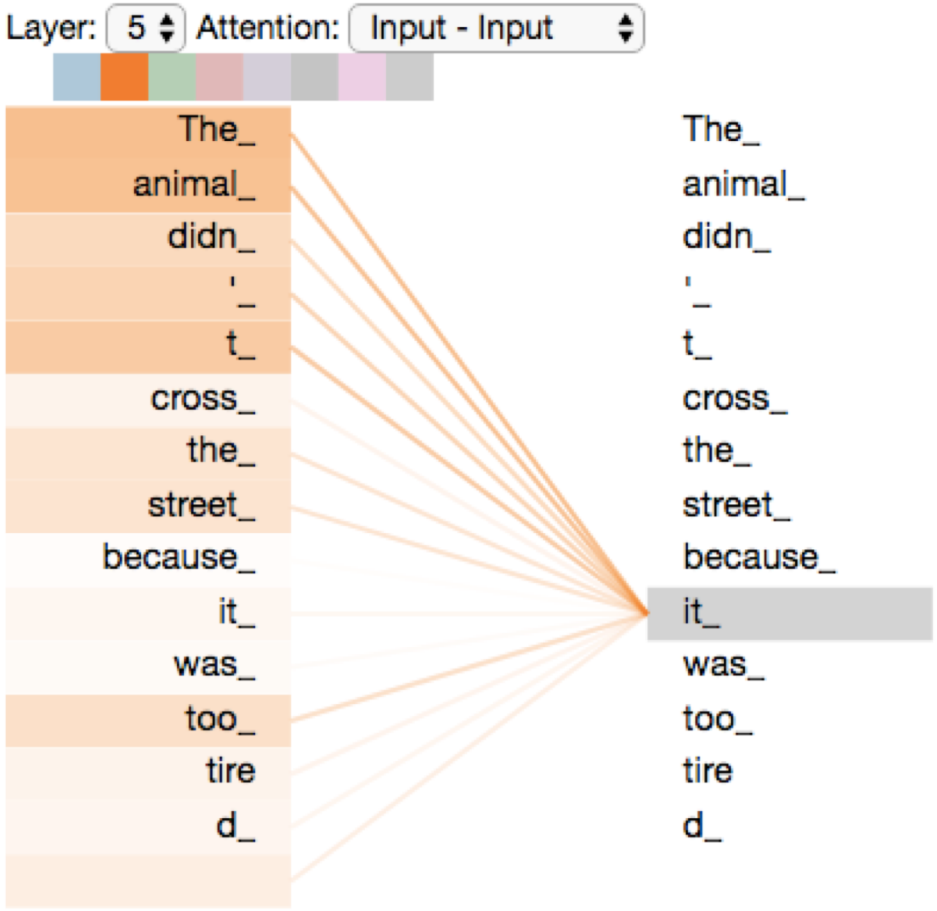
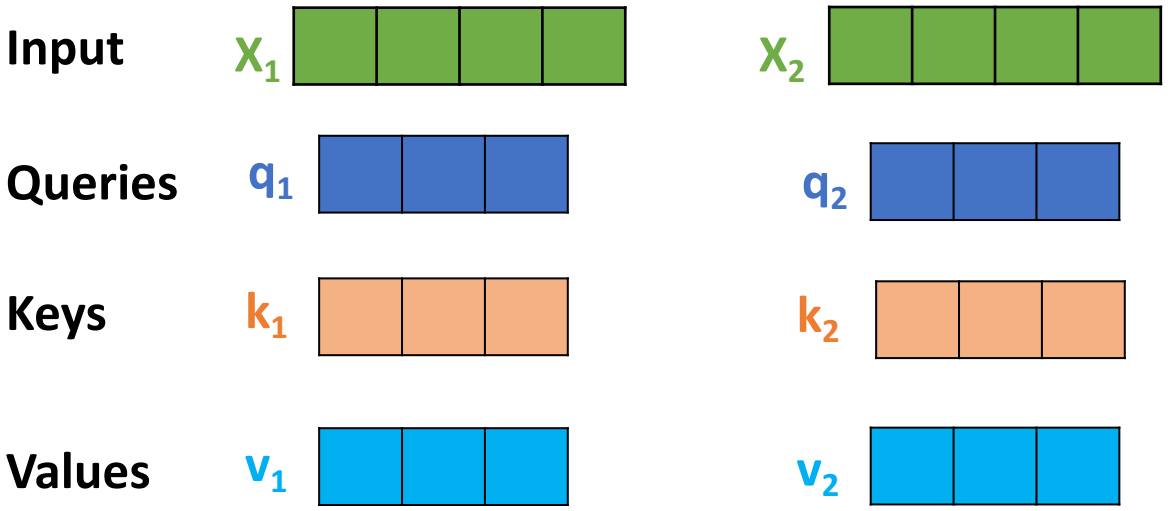


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention in Detail

- Attention maps a query and a set of key-value pairs to an output
  - query, keys, and output are all vectors

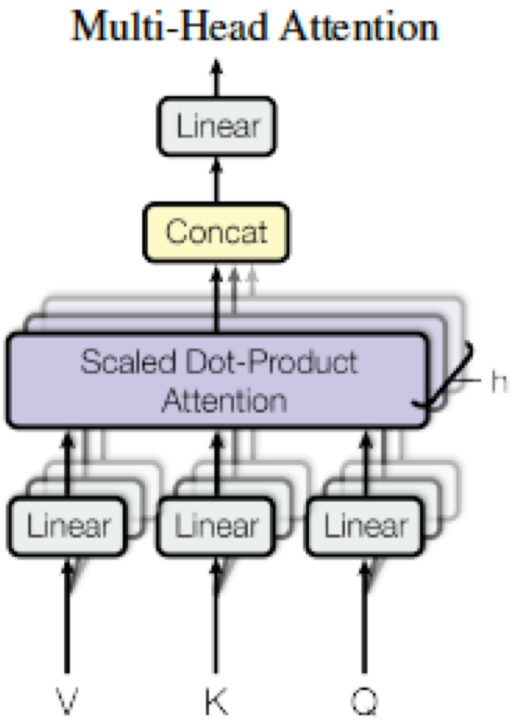


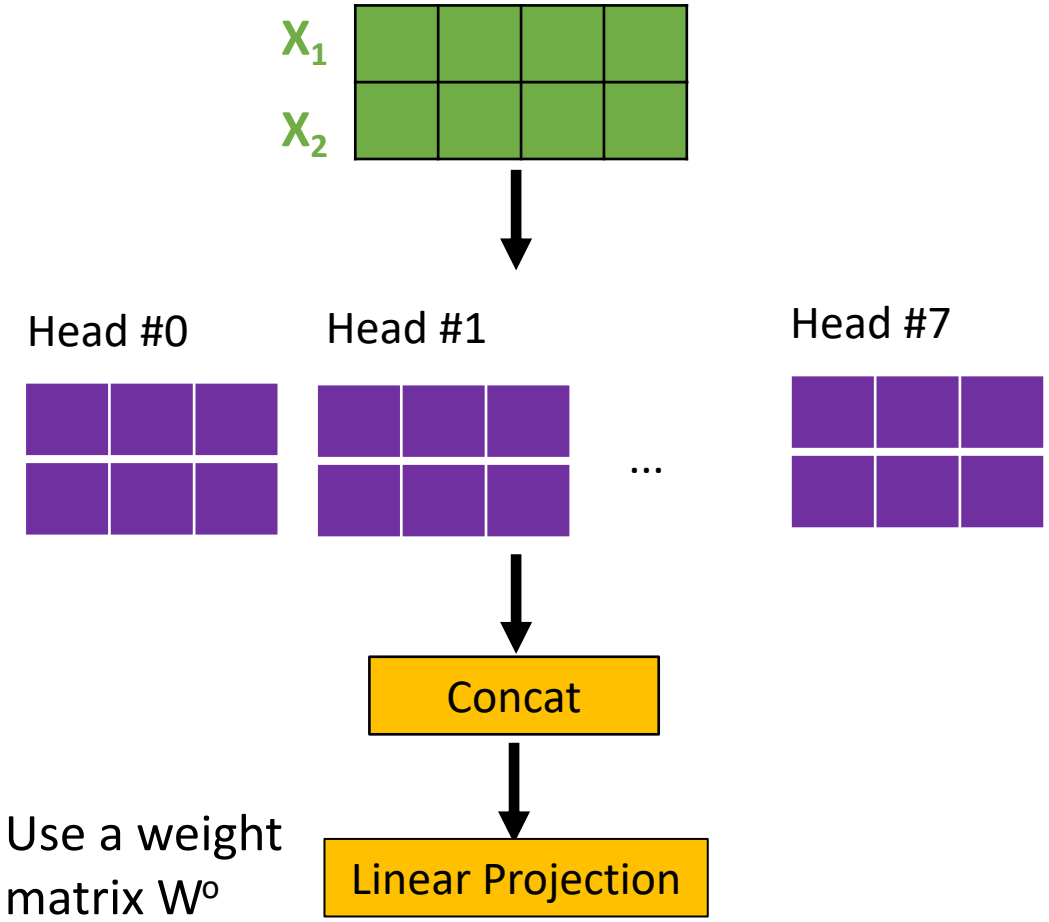
Use matrices W^Q, W^K and W^V to project input into query, key and value vectors

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

d_k is the dimension of key vectors

Multi-Head Attention





Position Encoding

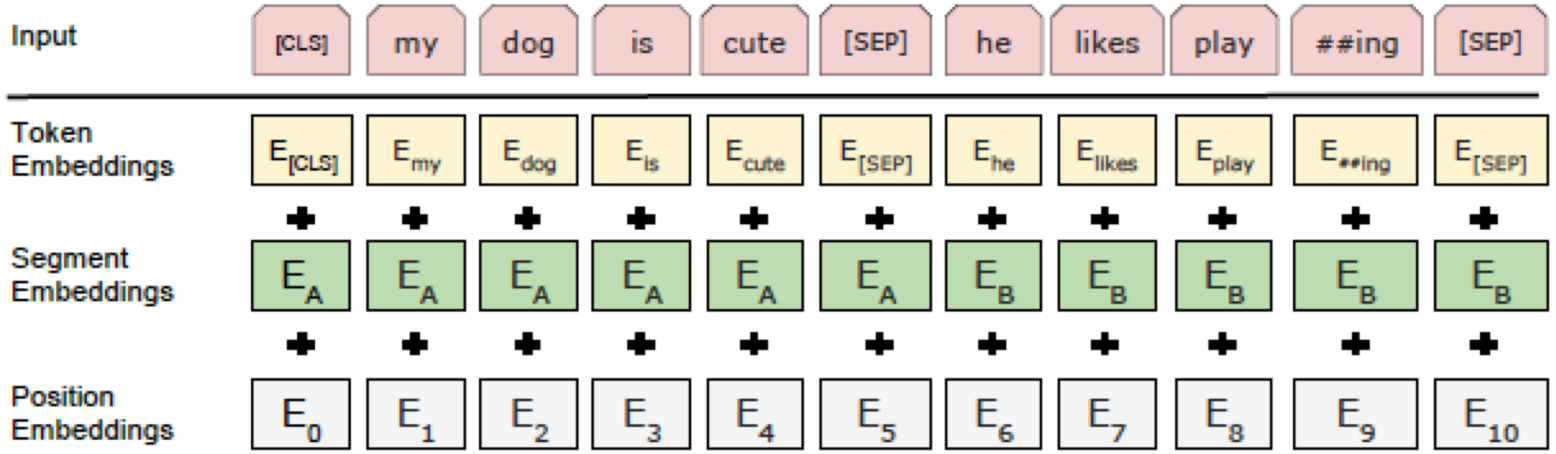
- Position Encoding is used to make use of the order of the sequence
  - Since the model contains no recurrence and no convolution
- In Vawasni et al., 2017, authors used sine and cosine functions of different frequencies

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

  - *pos* is the position and i is the dimension

Input Representation



- Token Embeddings: Use pretrained WordPiece embeddings
- Position Embeddings: Use learned Position Embeddings
  - Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutionalsequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.
- Added sentence embedding to every tokens of each sentence
- Use [CLS] for the classification tasks
- Separate sentences by using a special token [SEP]

Task#1: Masked LM

- 15% of the words are masked at random
  - and the task is to predict the masked words based on its left and right context
- Not all tokens were masked in the same way (example sentence “My dog is hairy”)
  - 80% were replaced by the <MASK> token: “My dog is <MASK>”
  - 10% were replaced by a random token: “My dog is apple”
  - 10% were left intact: “My dog is hairy”

Task#2: Next Sentence Prediction

- Motivation
  - Many downstream tasks are based on understanding the relationship between two text sentences
    - Question Answering (QA) and Natural Language Inference (NLI)
  - Language modeling does not directly capture that relationship
- The task is pre-training binarized next sentence prediction task

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

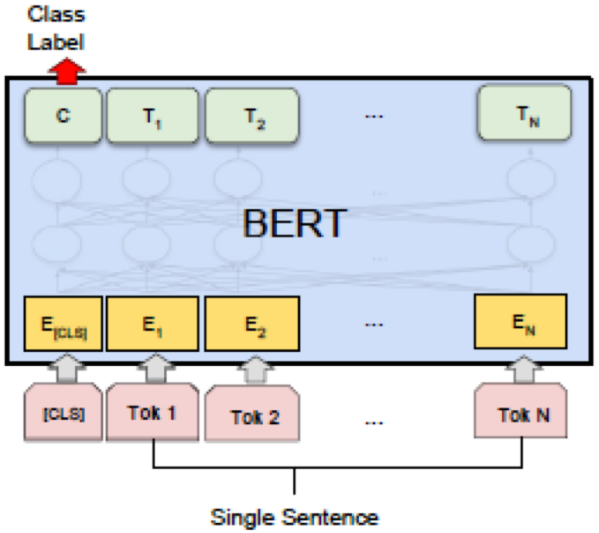
Pre-training procedure

- Training data: BooksCorpus (800M words) + English Wikipedia (2,500M words)
- To generate each training input sequences: sample two spans of text (A and B) from the corpus
  - The combined length is ≤ 500 tokens
  - 50% B is the actual next sentence that follows A and 50% of the time it is a random sentence from the corpus
- The training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood

Fine-tuning procedure

- For sequence-level classification task
  - Obtain the representation of the input sequence by using the final hidden state (hidden state at the position of the special token [CLS]) $C \in R^H$
  - Just add a classification layer and use softmax to calculate label probabilities. Parameters W $\in R^{K \times H}$

$$P = \text{softmax}(CW^T)$$



Fine-tuning procedure

- For sequence-level classification task
  - All of the parameters of BERT and *W* are fine-tuned jointly
- Most model hyperparameters are the same as in pre-training
  - except the batch size, learning rate, and number of training epochs

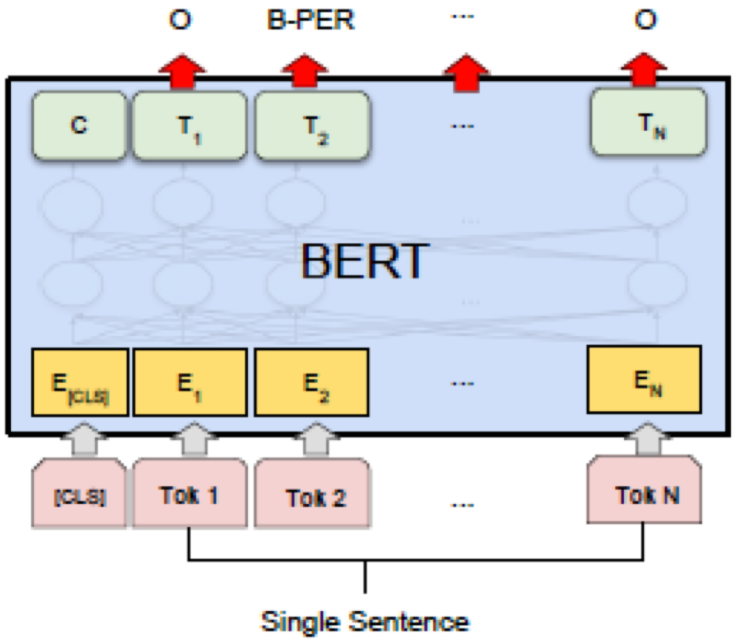
Fine-tuning procedure

- Token tagging task (e.g., Named Entity Recognition)
  - Feed the final hidden representation $T_i \in R^H$ for each token i into a classification layer for the tagset (NER label set)
  - To make the task compatible with WordPiece tokenization

```
Jim   Hen   ##son was a puppet ##eer
I-PER I-PER X     O   O O      X
```

    - Predict the tag for the first sub-token of a word
    - No prediction is made for X

Fine-tuning procedure



Single Sentence Tagging Tasks: CoNLL-2003 NER

Fine-tuning procedure

- Span-level task: SQuAD v1.1
  - Input Question:
    `Where do water droplets collide with ice crystals to form precipitation?`
  - Input Paragraph:
    `.... Precipitation forms as smaller dropletscoalesce via collision with other rain dropsor ice crystals within a cloud. ...`
  - Output Answer:
    `within a cloud`

Fine-tuning procedure

- Span-level task: SQuAD v1.1
  - Represent the input question and paragraph as a single packed sequence
    - The question uses the A embedding and the paragraph uses the B embedding
  - New parameters to be learned in fine-tuning are start vector $S \in \mathbb{R}^H$ and end vector $E \in \mathbb{R}^H$
  - Calculate the probability of word i being the start of the answer span

$$P_i = \frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$

  - The training objective is the log-likelihood the correct and end positions

Comparison of BERT and OpenAI GPT

OpenAI GPT	BERT
Trained on BooksCorpus (800M)	Trained on BooksCorpus (800M) + Wikipedia (2,500M)
Use sentence separater ([SEP]) and classifier token ([CLS]) only at fine-tuning time	BERT learns [SEP], [CLS] and sentence A/B embeddings during pre-training
Trained for 1M steps with a batch-size of 32,000 words	Trained for 1M steps with a batch-size of 128,000 words
Use the same learning rate of 5e-5 for all fine-tuning experiments	BERT choose a task-specific learning rate which performs the best on the development set

Outline

- Research context
- Main ideas
- BERT
- **Experiments**
- Conclusions

Experiments

- GLUE (General Language Understanding Evaluation) benchmark
  - Distribute canonical Train, Dev and Test splits
  - Labels for Test set are not provided
- Datasets in GLUE:
  - MNLI: Multi-Genre Natural Language Inference
  - QQP: Quora Question Pairs
  - QNLI: Question Natural Language Inference
  - SST-2: Stanford Sentiment Treebank
  - CoLA: The corpus of Linguistic Acceptability
  - STS-B: The Semantic Textual Similarity Benchmark
  - MRPC: Microsoft Research Paraphrase Corpus
  - RTE: Recognizing Textual Entailment
  - WNLI: Winograd NLI

GLUE Results

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	79.9	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/.

SQuAD v1.1

System	Dev		Test	
	EM	F1	EM	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.

Reference: https://rajpurkar.github.io/SQuAD-explorer

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.

SWAG

- The Situations with Adversarial Generations (SWAG)

A girl is going across a set of monkey bars. She

(i) jumps up across the monkey bars.

(ii) struggles onto the bars to grab her head.

(iii) gets to the end and stands on a wooden plank.

(iv) jumps up and does a back flip.

- The only task-specific parameters is a vector $V \in \mathbb{R}^H$
- The probability distribution is the softmax over the four choices

$$P_i = \frac{e^{V \cdot C_i}}{\sum_{j=1}^{4} e^{V \cdot C_i}}$$

SWAG Result

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.

Ablation Studies

- To understand
  - Effect of Pre-training Tasks
  - Effect of model sizes
  - Effect of number of training steps
  - Feature-based approach with BERT

Ablation Studies

- Main findings:
  - “Next sentence prediction” (NSP) pre-training is important for sentence-pair classification task
  - Bidirectionality (using MLM pre-training task) contributes significantly to the performance improvement

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

Table 5: Ablation over the pre-training tasks using the $\text{BERT}_{\text{BASE}}$ architecture. “No NSP” is trained without the next sentence prediction task. “LTR & No NSP” is trained as a left-to-right LM without the next sentence prediction, like OpenAI GPT. “+ BiLSTM” adds a randomly initialized BiLSTM on top of the “LTR + No NSP” model during fine-tuning.

Ablation Studies

- Main findings
  - Bigger model sizes are better even for small-scale tasks

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

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

Conclusions

- Unsupervised pre-training (pre-training language model) is increasingly adopted in many NLP tasks
- Major contribution of the paper is to propose a deep *bidirectional* architecture from Transformer
  - Advance state-of-the-art for many important NLP tasks

Links

- TensorFlow code and pre-trained models for BERT: https://github.com/google-research/bert
- PyTorch Pretrained Bert: https://github.com/huggingface/pytorch-pretrained-BERT
- BERT-pytorch: https://github.com/codertimo/BERT-pytorch
- BERT-keras: https://github.com/Separius/BERT-keras

Remark: Applying BERT for non-English languages

- Pre-trained BERT models are provided for more than 100 languages (including Vietnamese)
  - https://github.com/google-research/bert/blob/master/multilingual.md
- Be careful with tokenization!!
- For Japanese (and Chinese): “spaces were added around every character in the CJI Unicode rage before applying WordPiece” => Not a good way to do
  - Use SentencePiece: https://github.com/google/sentencepiece
  - We may need to pre-train BERT model

References

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.
2. Vaswani et al. (2017). Attention Is All You Need. arXiv preprint arXiv:1706.03762. https://arxiv.org/abs/1706.03762
3. The Annotated Transformer: http://nlp.seas.harvard.edu/2018/04/03/attention.html, by harvardnlp.
4. Dissecting BERT: https://medium.com/dissecting-bert