RoBERTa

A Robustly Optimized BERT Pretraining Approch

ROBERTA

By Facebook AI (fairseq Team)

A Robustly Optimized BERT Pretraining Approach

View on Github

Open on Google Colab

```
import torch
roberta = torch.hub.load('pytorch/fairseq', 'roberta.large')
roberta.eval()  # disable dropout (or leave in train mode to finetune)
```

RoNLPa : my Rapid Optimized NLP approch

① Attention is all you need

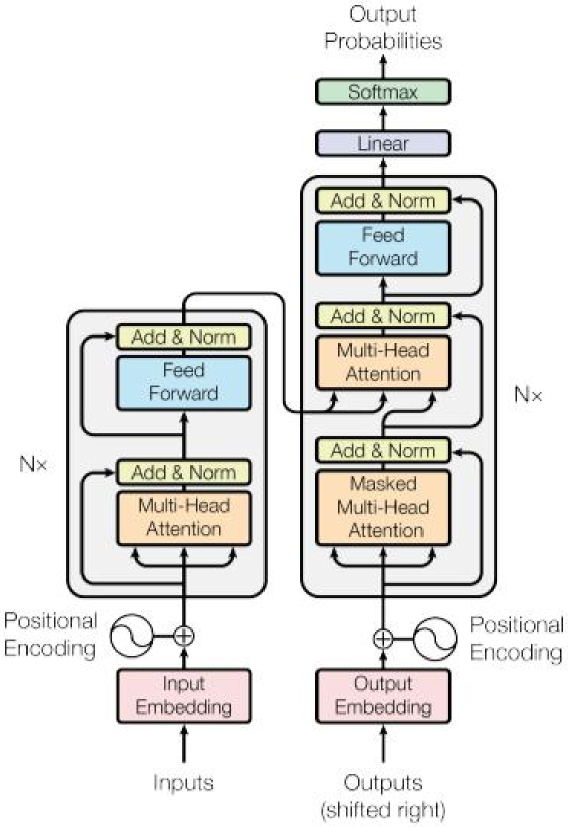
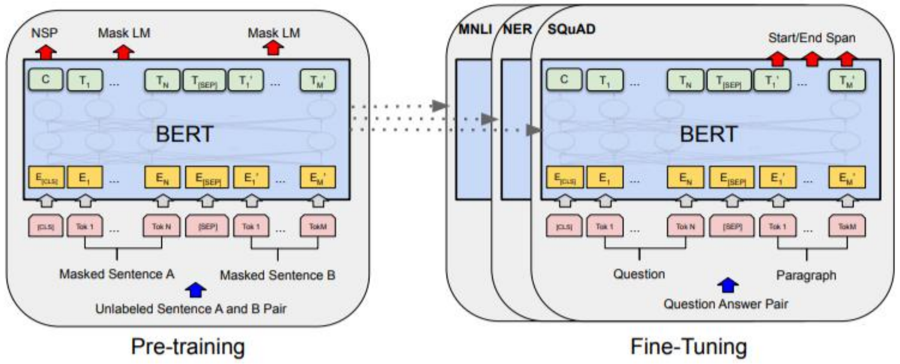
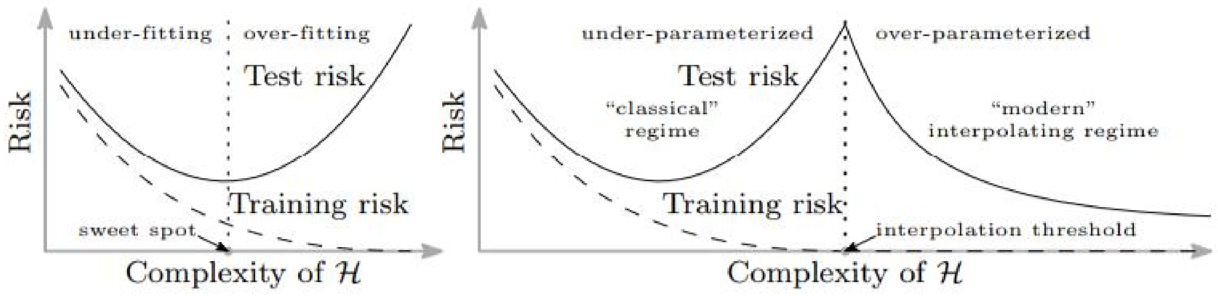


Figure 1: The Transformer - model architecture.

② BERT



③ RoBERTa



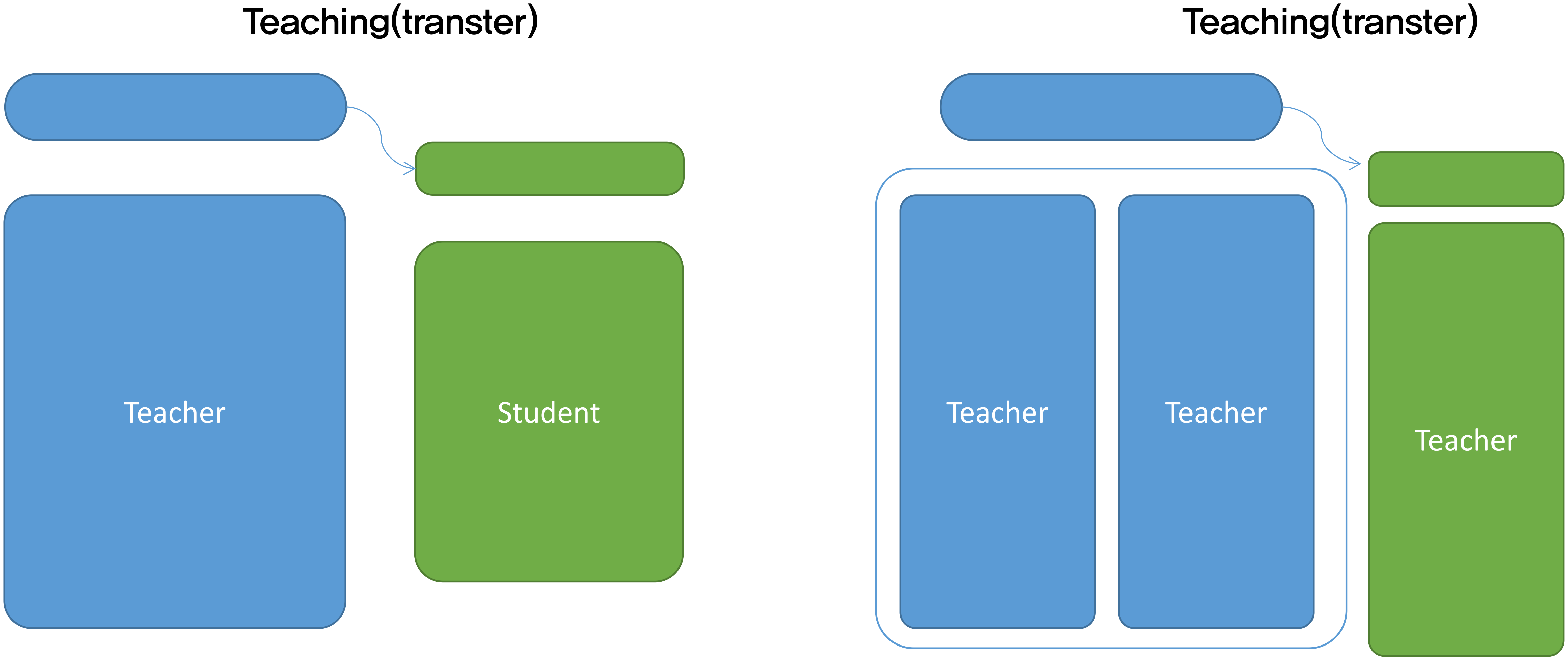
(a) U-shaped "bias-variance" risk curve

(b) "double descent" risk curve

Reconciling modern machine learning and the bias-variance trade-off

https://arxiv.org/pdf/1812.11118v1.pdf

Knowledge Distillation : Model Compression



BERT was undertrainded

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

1) Training the model longer with bigger batches over more data(+CC-News)

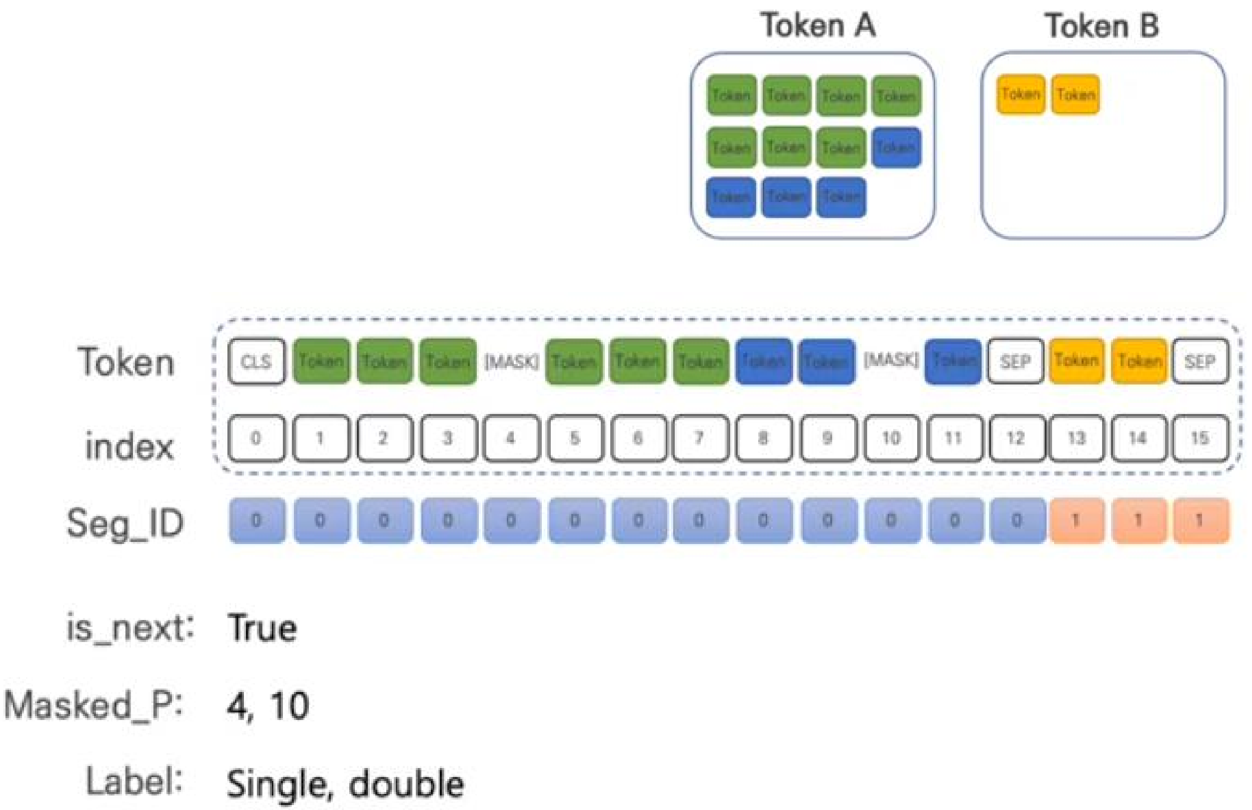
2) Removing the next sentence prediction objective

3) Training on longer sequence

4) Dynamically changed mask positions

BERT$_{base}$

- [CLS] Task-specific한 정보를 줌
- [SEP] Token A, B를 구분하는 token
- [SEP] Token A, B의 끝을 알리는 token
- [MASK] 예측할 마스크 토큰



1) 2개의 이상의 문장을 결합한 입력 데이터

2) 2개 이상의 문장이 1개의 문장처럼 Token과 함께 입력됨 [CLS], X1, ... ,Xn [SEP], Y1, ... ,Ym, [EOS]

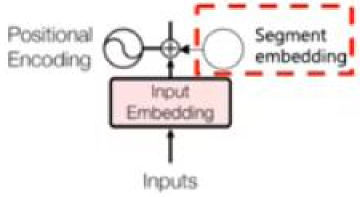
3) M + N 〈 T, 훈련하는 동안 문장 길이를 조절 (T = 512)

4) Word Piece embedding, Position embedding

5) 16GB data

BERT$_{base}$

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]



looking x_j

Vocab size(Encoder)

<PAD>	<s>	</s>	<UNK>	am	for	···	i	looking	···	w	···	z
0	0	0	0	0	0	0	0	1	0	0	0	0

Vocab size(Encoder)

A	B
1	0

Position vocab

0	1	2	3	4	5
0	0	1	0	0	0

- Encoder
  - Sent1: [CLS] i am looking for happiness [SEP], B type sentence [SEP]

looking

1. Token Embedding

0.2	0.1	0.7	0.2

+

2. Token Type Embedding

0.1	0.1	0.1	0.3

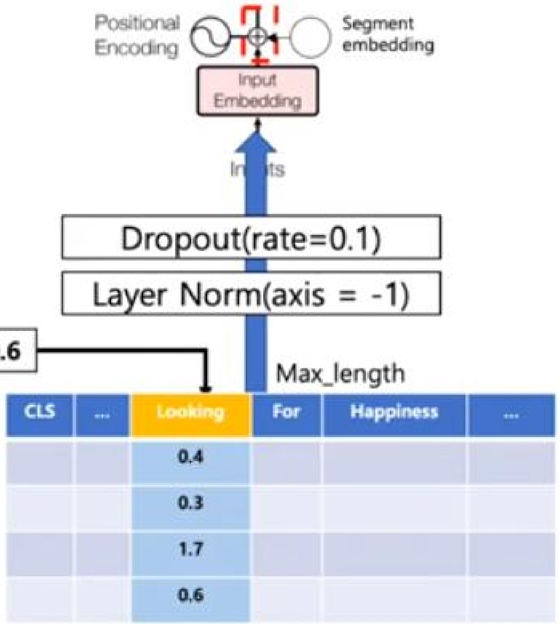
=

0.4	0.3	1.7	0.6

+

3. Position embeddings

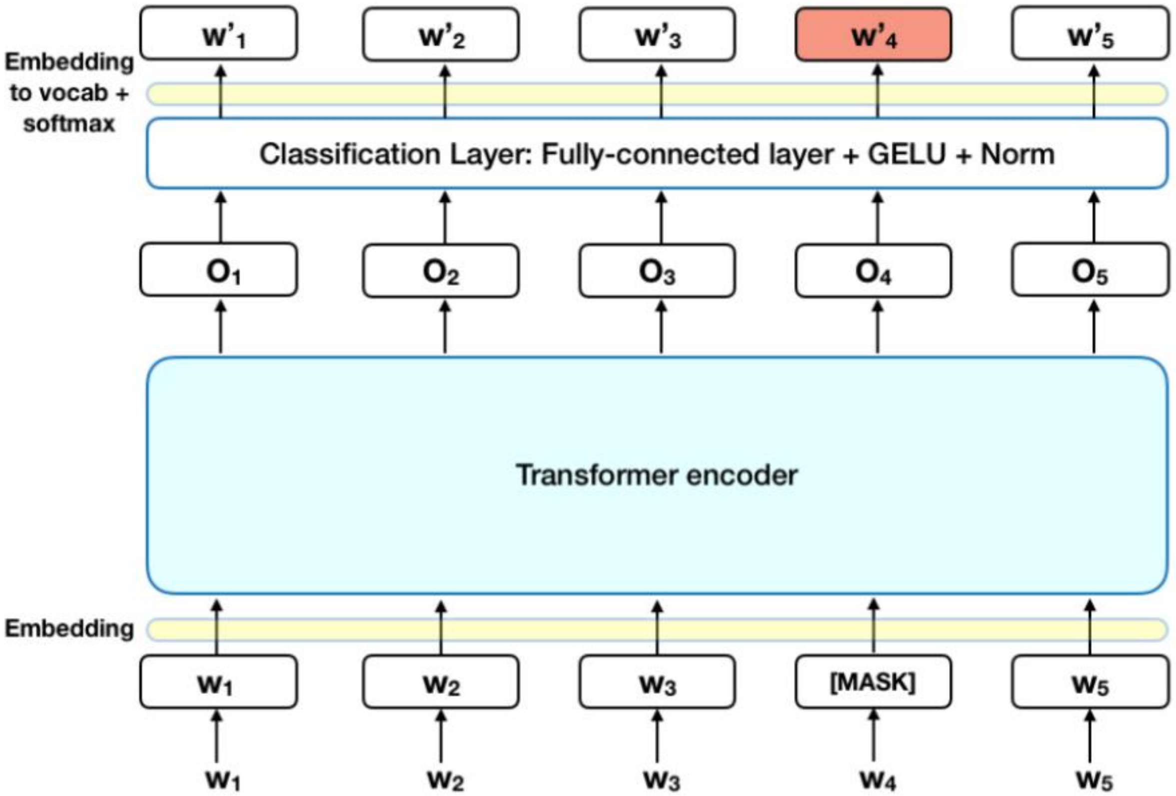
0.1	0.1	0.9	0.1



Embed_size

https://www.youtube.com/watch?v=xhY7m8QVKjo&t=6125s

BERT$_{base}$



	Before	After
80%	My dog is **hairy**	My dog is **[MASK]**
10%	My dog is **hairy**	My dog is **apple**
10%	My dog is **hairy**	My dog is **hairy**

1) Using only Transformer's encoder

2) BERT_large : L layer=24, Hidden deimension=1024,
self–attention head=16, Parameters = 340M

BERT_base : L=12, H=768, A=12, Parameters = 110M

3) Training obectives

– masked langauge model : Cross – entropy loss (15% of Input Tokens, masking 80%, changing 10%)

– next sentence prediction : Binary classification loss (from same or different documents)

Adam Optimizer

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

1) Adam Optimizer : $\beta_1 = 0.9$, $\beta_2 = 0.99 \rightarrow \beta_2 = 0.98$, $\varepsilon = 1e^{-6}$

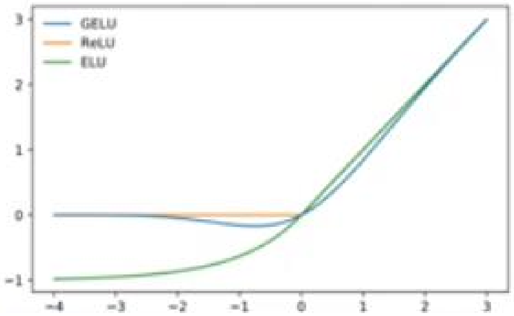
2) L2 weight decay of 0.01

3) Learning rate = 10,000 step $1e^{-4}$ than linealy decayed

4) dropout 0.1 on all layer, attention weight

5) GELU

6) 1,000,000 updates, 256 minibatches



More and More Data

1) Book Corpus + English wikipedia (16GB)
2) Common Crawl News(76GB) : 영문 기사 (2016.09 ~2019.02)
3) Open Web Text (38GB) : 적어도 3개 이상 upvote된 Redis의 web context를 추출함

Static vs Dynamic Masking

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

1) Base Bert는 전처리 단계에서 한번 Masking 처리를 함

- 동일한 문장이 epoch를 반복할 때 같은 마스크를 갖는 것을 피하기 위해
  10번 복재를 하고 다른 Masking을 적용한 뒤 40 epoch 를 동작하도록 함

2) RoBERTa는 매번 문장을 feeding 할 때 마스크 패턴을 바꿔서 적용함

Model Input Format & Next Sentence Prediction

SEGMENTS-PAIR(with NSP)

- original input format in BERT
- a pair of arbitrary spans of text

SENTENCE-PAIR(with NSP)

- a pair of natural sentences
- increase the batch size

FULL-SENTENCES

- 512 token의 연속된 문장
- EOD token

DOC-SENTENCES

- 하나의 문서만 연속 추출
- 문서의 마지막 부분 잘릴 수 있음

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

- 문장의 길이를 채워서 input을 넣는 것만으로도 NSP를 넣은 것보다 좋아짐
- FULL-SENTENCE와 토탈 토큰 수를 맞추기 위해 배치사이즈를 크게 증가시킴
- DOC-SENTENCES 형식은 배치 크기가 다양하기 때문에 쉽게 비교하기 위해 Full sentences를 사용

Training with Large Batches

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

1) BERTBASE for 1M steps with a batch size of 256 sequences. This is equivalent in computational cost, via gradient accumulation, to training for 125K steps with a batch size of 2K sequences, or for 31K steps with a batch size of 8K

2) 두 가지 간과한 것은 전 학습에 사용된 데이터와 훈련 횟수
   - XLNet은 10배 많은 데이터를 사용
   - 최적화 스탭은 반이었는데 배치 사이즈는 8배 였음

Byte-Pair Encoding

BERT : A character level BPE vocabulary size of **30K**, learned after preprocessing the input with heuristic tokenization rules.

RoBERTa : BPE subword vocabulary size of **50K**

This adds approximately 15M and 20M additional parameters for BERT BASE and BERT LARGE, respectively

Evaluation

GLUE 9 datasets

- MNLI : Multi-Genre Natural Langage Inference

  현재 문장 다음에 이어지는 문장이 문맥상 이어지는 문장인지, 반대인지, 상관 없는 문장인지 분류
- QQP: Quora Question Pairs

  두 질문이 의미상 같은지 다른 지 분류
- QNLI : Question Natural Language Inference

  질의응답 데이터셋
- SST-2: The Stanford Sentiment Treebank

  영화 리뷰 문장에 관한 감정 분석을 위한 데이터셋
- CoLA: The Corpus of Linguistic Acceptability

  문법적으로 맞는 문장인지 틀린 문장인지 분류
- STS-B : The Semantic Textual Similarity Benchmark

  뉴스 헤드라인과 사람이 만든 paraphrasing 문장이 의미상 같은 문장인지 비교
- MRPC : Microsoft Research Paraphrase Corpus

  뉴스의 내용과 사람이 만든 문장이 의미상 같은 문장인지 비교
- RTE : Recognizing Textual Entailment

  MNLI와 유사하나 상대적으로 훨씬 적은 학습 데이터셋
- WNLI : Winograd NLI

  문장 분류

Result

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

Result

Model	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Single models on dev, w/o data augmentation				
$\text{BERT}_{\text{LARGE}}$	84.1	90.9	79.0	81.8
$\text{XLNet}_{\text{LARGE}}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$\text{XLNet}_{\text{LARGE}}$			$86.3^{\dagger}$	$89.1^{\dagger}$
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			$\mathbf{87.0}^{\dagger}$	$\mathbf{89.9}^{\dagger}$

Table 6: Results on SQuAD. † indicates results that depend on additional external training data. RoBERTa uses only the provided SQuAD data in both dev and test settings. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively.

Model	Accuracy	Middle	High
Single models on test (as of July 25, 2019)			
$\text{BERT}_{\text{LARGE}}$	72.0	76.6	70.1
$\text{XLNet}_{\text{LARGE}}$	81.7	85.4	80.2
RoBERTa	**83.2**	**86.5**	**81.3**

Table 7: Results on the RACE test set. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Yang et al. (2019).

Hyperparam	RACE	SQuAD	GLUE
Learning Rate	1e-5	1.5e-5	{1e-5, 2e-5, 3e-5}
Batch Size	16	48	{16, 32}
Weight Decay	0.1	0.01	0.1
Max Epochs	4	2	10
Learning Rate Decay	Linear	Linear	Linear
Warmup ratio	0.06	0.06	0.06

Table 10: Hyperparameters for finetuning $\text{RoBERTa}_{\text{LARGE}}$ on RACE, SQuAD and GLUE.

감사합니다