RoBERTa:
A Robustly Optimized BERT Pretraining Approach

2021.08.05

백서인

Abstract

- We present a replication study of BERT pertaining that carefully measures the impact of many key hyper parameters and training data size

- We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it

- Our Best model achieves state-of-the-art results on GLUE, RACE and SQuAD.

Introduction

- It can be challenging to determine which aspects of the methods contribute the most

- We find that BERT was significantly undertrained and propose an improved recipe for training BERT models, which we call RoBERTa, that can match or exceed the performance of all there post-BERT methods

Introduction

- Our modifications are simple
  - (1) training the model longer, with bigger batches, over more data
  - (2) removing the next sentence prediction objective
  - (3) training on longer sequences
  - (4) dynamically changing the masking pattern applied to the training data
- We also collect a large new dataset (CC-NEWS) of comparable size to other privately used dataset, to better control for training set size effects

Background

- Brief overview of the BERT
  - Input: concatenation of two segments (sequences of tokens) with special tokens like CLS, SEP, EOS
  - The model is first pertained on a large unlabeled text corpus and subsequently fine-tuned using end-task labeled data
  - Use transformer architecture with L layers, each block uses A self-attention heads and hidden dimension H
  - Training objectives: masked language modeling and next sentence prediction
  - Data: combination of BOOKCORPUS + English WIKIPEDIA

Experimental Setup

BERT

Adam using the following parameters:
$\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 1e-6$ and L_2 weight decay of 0.01

Learning rate: warmed up over the first 10,000 steps to a peak value of $1e-4$, and then linearly decayed

Dropout: 0.1 on all layers and attention weights

GELU activation function

Models are pertained for S = 1,000,000 updates, and with minimum batches containing B = 256 sequences of maximum length T = 512 tokens

RoBERTa

Follow the original BERT optimization hyper parameters, except for the peak leaning rate and number of warmup step

Additionally found training to be very sensitive to the Adam epsilon term, and in some cases we obtained better performance or improved stability after tuning it

Also, found that setting $\beta_2 = 0.98$ to improve stability when training with large batch sizes

We do not randomly inject short sequences, and we do not train with a reduced sequence length for the first 90% of updates.

We train only with full-length sequences

Data

- For our study, we focus on gathering as much data as possible for experimentation, allowing us to match the overall quality and quantity of data as appropriate for each comparison

- We consider five English-language copra of varying sized and domains, totaling over 160GB of uncompressed text

  - BOOKCORPUS + English WIKIPEDIA (16GB)

  - CC-NEWS (76GB)

  - OPENWEBTEXT (38GB)

  - STORIES (31GB)

Evaluation

- GLUE: a collection of 9 datasets for evaluating natural language understanding systems
- SQuAD: the Stanford Question Answering Dataset
- RACE: a large-scale reading comprehension dataset

Static vs. Dynamic Masking

- The original BERT implication performed masking once during data preprocessing, resulting in a single static mask

- To avoid using the same mask for each training instance in every epoch, training data was duplicated 10 times so that each sequence is masked in 10 different ways over the 40 epochs of training (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

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

Model Input Format and Next Sentence Prediction

- Some recent work has questioned the necessity of the NSP loss
- We find that using individual sentences hurts performance on downstream tasks
- We find that removing the NSP loss matches or slightly improves downstream task performance

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

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

- NSP를 제거했을 때, 전체적으로 좋은 성능을 보임.
- Full-sentence와 Doc-sentence를 비교했을 때, Doc-sentence의 경우가 미세하게 좋지만 Batch size를 다양하게 조절해야하기 때문에 이후 실험들에서는 Full-sentence의 입력 구성을 이용

Training with large batches

- Past work has shown that training with very large mini-batches can both improve optimization speed and end-task performance
- Recent work has shown that BERT is also amenable to large bath training
- We observe that training with large batches improves perplexity for the tasked language modeling objective, as well as end-task accuracy
- Large batches are also easier to parallelize via distributed data parallel training, and in later experiments we train with marches of 8K sequences

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.

- 2K일 때의 성능이 가장 좋지만, 분산 처리 시의 이점으로 이후 실험에서는 8K 사용

Text Encoding

- Byte-Pair Encoding (BPE) is a hybrid between character- and word-level representations that allows handling the large vocabularies common in natural language corpora

- Instead of full words, BPE replies on subwords units, which are extracted by performing statistical analysis of the training corpus

- Radford et al. (2019) introduce a clever implementation of BPE that used bytes instead of unicode characters as the base subword units

RoBERTa

- (1) dynamic masking
- (2) Full-sentence without NSP
- (3) large mini-batches
- (4) larger byte-level BPE

- Additionally, we investigate two other important factor
- (1) data used for pretraining
- (2) number of training passes through the data

Results

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

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

GLUE Results

- Our submission depends only on single-task fine-tuning

	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.

SQuAD Results

- We only finetune RoBERTa using the provide SQuAD training data

Model	SQuAD 1.1 EM	SQuAD 1.1 F1	SQuAD 2.0 EM	SQuAD 2.0 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†	89.1†
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

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.

RACE Results

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).