RoBERTa : A Robustly Optimized BERT Pretraining Approach

Yinhan Liu et al

Paul G. Allen School of CS & Engineering | Univ of Washington

Seungone Kim

Department of Computer Science

Yonsei University

louisdebroglie@yonsei.ac.kr

2021.07.14

연세대학교 YONSEI UNIVERSITY | SOFT COMPUTING LABORATORY

Outline

- 관련 연구
  - 논문 발표 시점 언어 모델 동향 분석
  - 기존 방법들의 한계점

- 제안하는 방법
  - Training the model longer, with bigger batches, over more data
  - Removing the NSP Objective
  - Training on longer sequences
  - Dynamically changing the masking pattern
  - Text Encoding

- 실험 결과
  - Hyperparameters compared to BERT
  - GLUE, SQuAD, RACE Results

- 분석 및 요약

- 결론

관련 연구

- 논문 발표 시점 : 2019.07
  - BERT 발표 시점 : 2018.10
  - XLNet 발표 시점 : 2019.06

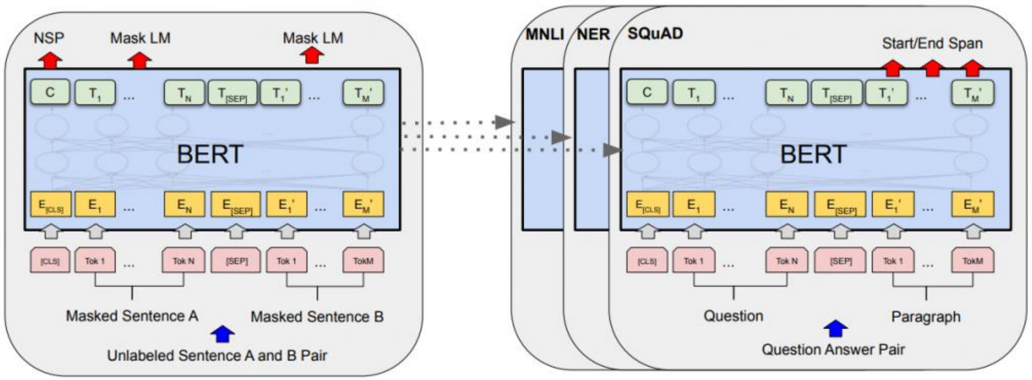
- Self-training methods have brought significant performance gains
  - ELMo, GPT, BERT, XLM, XLNet
  - It can be challenging to determine which aspects of the methods contribute the most

- Training is computationally expensive
  - Limits the amount of tuning that can be done

- Training is often done with private training data of varying size
  - Limits the ability to measure the effects of modeling advances

관련 연구 (기존 방법들의 한계점)

- BERT was significantly undertrained(underfit)
  - Need to evaluate the effects of hyperparameter tuning and training set size
  - BERT's MLM objective is competitive with other recently proposed training objectives



Pre-training   Fine-Tuning

- Background of BERT
  - Takes as input a concatenation of two segment
    - $[\text{CLS}]\ x_1, x_2, \ldots, x_N\ [SEP]\ y_1, y_2, \ldots, y_M\ [EOS]\ such\ that\ M + N < T, where\ T\ is \max sequence\ length$
  - Model is first pretrained on a large unlabeled text corpus and finetuned using labeled data
  - Pretrain : MLM objective is a cross-entropy loss on predicting 15% masked tokens
    - In original implementation, random masking and replacement is performed once in beginning
  - Pretrain : NSP objective is a binary classification loss for predicting two segments are consecutive
    - Positive, negative examples are chose by 50%, and was designed to improve performance in NLI
  - Pretrain data : BOOKCORPUS *(Zhu et al., 2015)*

제안하는 방법

- Modification made on BERT & Changed Training Procedures

    - Training the model longer with more data
    - Training the model with bigger batches
    - Removing the next sentence prediction objective(NSP)
    - Training on longer sequences
    - Dynamically changing the masking pattern applied to the training data
    - Text Encoding

제안하는 방법

- More data! (16GB -> 160GB)
  - Increasing data size can result in improved end-task performance (Baevski et al., 2019)
  - BOOKCORPUS (Zhu et al., 2015) + ENGLISH WIKIPEDIA
    - 16GB
    - Original data used to train BERT
  - CC-NEWS
    - 76GB after filtering
    - Newly collected dataset for pretraining
    - English portion of the CommonCrawl News dataset (Nagel.,. 2016)
  - OPEN WEB TEXT
    - 38GB
    - Open-source recreation of WebText corpus (Radford et al., 2019)
  - STORIES
    - 31GB
    - Subset of CommonCrawl data filtered to match the story-like style (Trinh and Le., 2018)

제안하는 방법

- More data! (16GB -> 160GB)

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

제안하는 방법

- Training with large batches

  - Training with large mini-batches improve optimization speed and end-task performance
    - This is when learning rate is increased appropriately
    - Ott et al., 2018

  - Training with large batches improve perplexity for MLM Objective
    - BERT-base : 1M steps with batch size of 256 sequences
    - RoBERTa : 2K steps with batch size of 8000 sequences (8x larger than BERT)

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

제안하는 방법

- Does Next Sentence Prediction(NSP) & Sentence Length matter?

  - NSP Loss was hypothesized to be an important factor in training BERT
    - Removing NSP hurts performance on QNLI, MLNI and SQuAD

  - Some recent work has questioned necessity of NSP Loss
    - Lample and Conneau., 2019 / Yang et al., 2019 / Joshi et al., 2019

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

  - Compare several alternative training formats
    - Segment-pair + NSP : original input format used in BERT with NSP Loss
    - Sentence-pair + NSP : input is consisted of natural sentences with NSP Loss
    - Full Sentence : Input is packed with full sentences sampled contiguously (May cross document boundary)
    - Doc Sentence : Similar to Full Sentence, but do not allow cross document boundaries

    - For Sentence-pair and Doc Sentence, if shorter than 512 tokens, increase batch size
    - Then, the total number of tokens per batch is similar

제안하는 방법

- Static vs Dynamic Masking

  - BERT implementation performed masking once during data preprocessing (static mask)
    - Data was duplicated 10 times so that each sequence is masked 10 different ways over 40 epoch
    - Each training sequence was seen with the same mask four times during training

  - Dynamic Masking : generate the masking pattern every time feeding a sequence
    - Crucial when pretraining for more steps or larger datasets
    - Comparable or slightly better than static 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

제안하는 방법

- Text Encoding
  - BPE is a hybrid between character and word level representations
    - Relies on subword units, which are extracted by performing statistical analysis on training corpus
    - Radford et al., 2019 introduces a clever implementation of BPE that uses bytes instead of Unicode
    - Subword vocabulary of a modest size(50K)
    - Original BERT : WordPiece, character-level BPE vocabulary of size 30K

실험 결과

- Hyperparameters compared to BERT

BERT	RoBERTa
Base : 110M (12/768) Large : 340M (24/1024)	Base : 125M (12/768) Large : 355M (24/1024)
Base : 8 * V100 * 12days Large : 280 * V100 * 4days	Large : 1024 * V100 * 1day (4~5 times more than BERT)
$\beta_1 = 0.9, \beta_2 = 0.999$	$\beta_1 = 0.9, \beta_2 = 0.99$
lr = warmup over 10000 until 1e-4, then drop linearly	lr = warmup over 30000 until 4e-4, then drop linearly
Batchsize=256 seq for 1M update	Batchsize=8K seq for 500K update

실험 결과

- GLUE Result
  - First setting : finetune RoBERTa separately for each GLUE task
  - Second Setting : Other depend on multitask finetuning, our submission depends only on single-task finetuning
  - For RTE, STS and MRPC it is helpful to finetune starting from MNLI single-task model

	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.

실험 결과

- SQuAD Results
  - Finetune RoBERTa using the provided SQuAD training data
  - While BERT, XLNet rely on external training data, our submission does not use additional 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.

실험 결과

- RACE Results
  - Provided with a passage of text, an associated question, and four candidate answers

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).

분석 및 요약

- Our goal was to replicate, simplify, and better tune the training of BERT
  - Importance of these previously overlooked design decisions and suggest that BERT's pretraining objective remains competitive with recently proposed alternatives.

- Question about relative importance between model architecture and pretraining objective, compared to mundane details like dataset size and training time
  - Performance can be substantially improved with subtle change to existing model

결론

- Possible future works
  - Exploration of the limits of large batch training
  - It is possible that other methods could also improve with more training
  - Detailed comparison between encoding schemes
  - Increase in data size and diversity should be more carefully analyzed
  - Using multi-task fine tuning, we could expect better results