RoBERTa: A Robustly Optimized BERT Pretraining Approach

BERT를 발전시킨 RoBERTa를 제안한 논문인 RoBERTa: A Robustly Optimized BERT Pretraining Approach에 대해 리뷰해보려 한다.

원문 링크는 다음과 같다.

RoBERTa: A Robustly Optimized BERT Pretraining Approach

Introduction

논문의 저자들은 hyperparameter tuning과 training set size의 효과에 관련된 평가를 포함한 BERT의 pre-training에 대한 replication study를 제시한다.

이러한 replication study를 통해, 저자들은 BERT가 상당히 undertrain 됨을 발견하였고, 이를 개선하여 RoBERTa라는 BERT training의 향상된 방법을 제시한다. 이러한 RoBERTa는 기존 BERT 기반 model들의 성능을 뛰어넘는다고 주장한다.

저자들은 RoBERTa가 기존 BERT와 다른 점에 대해 다음과 같이 기술하였다.

- **더 많은 data로 더 큰 batch size를 설정하여 더 오래 training함**
- **NSP(Next Sentence Prediction) 제거**
- **더 긴 sequence로 training**
- **training data에 dynamic masking 적용**

또한, CC-NEWS라는 large dataset을 새롭게 제안한다.

RoBERTa는 GLUE와 SQuAD에서 기존 BERT보다 향상된 성능을 보여주었으며, 4개의 GLUE task에서(MNLI, QNLI, RTE, STS-B) SOTA를 달성하였다고 한다.

정리하자면, 저자들이 논문에서 설정한 contribution은 아래와 같다.

- BERT의 중요한 **design choice와 training strategies를 소개하며, downstream task에서 더 좋은 성능을 내는 대안 제시**
- Downstream task에서의 성능 향상을 위해 **더 많은 data를 포함하고 있는 CC-NEWS dataset 제안**
- 올바른 **design choice를 통한 masked language model pretraining은 타 방법론과 비교했을 때 충분히 경쟁력이 있음**을 보임

그러면, 이러한 RoBERTa에 대해 하나씩 살펴보도록 하자.

Background

논문에서의 Background section은 BERT에 관한 overview를 다루고 있다.

이 부분은 하단에 사전에 작성했었던 BERT관련 게시물 링크를 남겨놓고 넘어가도록 하겠다.

(논문 리뷰) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - BERT

Experimental Setup

Experimental setup section에서는 저자들이 수행했던 BERT의 replication study에 대해 다룬다.

Implementation

논문에서는 기본적으로 original BERT의 optimization hyperparameter를 따르지만, 예외적으로 peak learning rate와 warmup step은 각각의 setting마다 달라지게끔 하였다.

저자들은 training이 adam optimizer의 epsilon term에 매우 민감하다는 것을 발견하였고, 몇몇 경우에는 adam optimizer의 epsilon term을 조정한 후에 더 높은 성능 혹은 안정성을 확보하였다고 한다.

이와 비슷하게, large batch size로 training을 진행할 경우에는 adam optimizer의 β_2 hyperparameter를 0.98로 조정할 때 안정성이 향상됨을 발견하였다. (Original BERT에서는 β_2가 0.999였다.)

이어서, 저자들은 sequence의 maximum length T를 512로 설정하였지만, original BERT에서 pre-training step의 90%에 대해 T를 128로 줄여서 학습한 것과는 달리 **pre-training의 처음부터 끝까지 T=512를 유지했다고 한다.**

(위에 언급된 논문 포스팅에서는 추가하지 않았지만, BERT 논문의 appendix A.2 에서는(단, 2019년에 업로드된 버전 기준이며, 이는 위 BERT 리뷰 게시글의 원문 링크와 일치한다.) pre-training step의 90%에 대해 T를 128로 줄여서 학습하고, 나머지 10%의 step에서는 원래대로인 T=512로 학습했다고 밝힌다.)

또한 저자들은 Mixed precision training을 적용하였다. 이에 관해서는 아래의 링크를 참고 바란다.

(논문 리뷰) Mixed Precision Training

Data

BERT-style의 pretraining은 많은 양의 text크기에 의존한다. 선행 연구들에서도 data size가 커지면 end-task에서의 성능이 향상됨이 입증되어왔다.

Original BERT보다 더 크고 더 다양한 dataset으로 train 하는 연구들이 진행되어왔지만, 이러한 연구들에서 사용된 추가적인 dataset들이 전부 공개되지는 않는다. 그래서 저자들은 실험을 위해 많은 양의 data들을 확보해왔고, 해당 목록은 아래와 같다.

- BookCorpus + English Wikipedia : original BERT에서 사용된 dataset (16GB)
- CC-NEWS : 2016년 9월부터 2019년 2월까지 수집한 뉴스 기사 dataset (76GB)
- OpenWebText : Reddit에서 최소 3개의 추천을 받은 content를 수집한 dataset (38GB)
- Stories : Winograd schema의 story-like style과 일치하도록 필터링된 CommonCrawl data의 하위 집합을 포함함 (31GB)

Evaluation

본 연구에서는 model의 성능을 평가하기 위해 GLUE, SQuAD, RACE의 3가지 benchmark를 사용했다.

GLUE는 QA(Question Answering), Sentiment analysis, Textual entailment를 포함하는 NLU task의 모음이며, 이 또한 기존에 리뷰한 적이 있기에, 하단에 리뷰 게시물 링크를 덧붙이도록 하겠다.

(논문 리뷰) GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding - GLUE

SQuAD(Stanford Question Answering Dataset)는 question과 answer를 담고 있는 passage를 제공하며, 이를 통해 passage 속에 있는 answer를 예측하는 Question Answering task를 진행한다.

논문에서는 SQuAD v1.1과 v2.0 모두 사용하였는데, SQuAD v2.0은 제공된 passage에 answer이 존재하지 않을 가능성을 허용함으로써 SQuAD v1.1에서의 문제 정의를 확장한 버전이다. RoBERTa에서는 이를 반영하기 위해 answer가 존재하는지에 대한 여부를 CLS token을 이용해 분류하는 과정을 추가한다. (이 점은 BERT와 동일하다.)

이후 evaluation이 진행될 때는 answer가 존재하는 pair들에 대해서만 answer를 예측한다.

RACE는 28,000개 이상의 passage와 100,000개 이상의 question으로 이루어진 reading comprehension dataset이다. 이는 중국의 중고등학교 영어 시험으로부터 수집되었다. 각각의 passage가 multiple question을 가지고 있으며, 모든 question에 대한 4개의 option 중 하나의 정답을 선택하는 task이다.

이러한 **RACE는 타 reading comprehension dataset보다 더 긴 context를 가지고 있으며, reasoning을 필요로 하는 question의 비율이 매우 높은 특징**을 가지고 있다.

Training Procedure Analysis

본 연구에서는 성공적으로 BERT를 pretrain 하기 위해서는 어떠한 선택이 중요한지에 대해 탐색하고 정량화한다. 저자들은 BERT-BASE model의 architecture($L = 12$, $H = 768$, $A = 12$, 110M params)를 유지한 상태로 다양한 실험을 진행한다.

Static vs Dynamic Masking

Original BERT의 경우, masking을 data preprocessing 과정에서 한 번만 진행한다 (single-static mask)

매 epoch마다 각각의 training instance에 같은 mask가 사용되는 것을 피하기 위해, training data가 10번 복제되면서 각각의 sequence가 40 epoch의 training동안 10개의 다른 방식으로 masking이 되는 방식을 사용하고, 이는 결론적으로, 각각의 training sequence는 전체 training 과정에서 동일한 mask로 4번 관측되는 효과를 불러일으킨다. (아래의 결과 table에서의 static)

저자들은 위와 같은 방법론과 **model에 sequence를 feed 할 때마다 masking pattern을 생성하는 (즉, 매 epoch마다 다른 masking을 적용하는) dynamic masking**을 비교한다. 또한 이러한 dynamic masking은 더 많은 step 혹은 더 큰 dataset으로 pretraining 할 때 매우 중요해진다고 말한다. (dataset을 여러 번 복제하는 방식은 step이 많아지거나 dataset이 커지면 computational cost도 기하급수적으로 차지하기 때문이다.)

static masking과 dynamic masking의 결과 비교는 다음과 같다.

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).

static masking으로 재 구현한 경우에는 original BERT와 비슷한 성능을 내고, dynamic masking의 경우에는 이와 비슷하거나 조금 더 좋은 성능을 내는 것을 확인할 수 있다.

저자들은 이러한 결과와 더불어 dynamic masking이 주는 efficiency로 인해, 이후의 실험들에서도 dynamic masking을 사용한다고 밝힌다.

(처음 논문을 읽었을 때, training data를 10번 복제하는 방법이 나온 뒤, "We compare this strategy with dynamic masking where we generate the masking pattern every time we feed a sequence to the model"이라는 문구가 이어지기에 당연히 dynamic masking은 매 epoch마다 다른 masking을 생성하는 방법론이라고 생각했었다. 그러나, 구글 BERT의 정석이라는 교재에서는 training data를 10번 복제하는 방법론을 dynamic masking이라고 소개한다. 이에 잘못 이해했나 싶어 자세한 정보를 찾기 위해 여러 문헌들을 찾아봤지만 전자로 설명하는 곳과 후자로 설명하는 곳이 혼재하였다. 그러던 와중, 한국전자통신연구원 ETRI에서 발표한 자료를 찾게 된다. 해당 자료로 인해 dynamic masking에 대한 혼동을 멈출 수 있었으며, 이 부분에 대해 위와 같이 정의하게 되었다. ETRI에서 발표한 자료는 아래에 첨부하도록 하겠다. 그러나 나의 이해가 틀렸을 수도 있기에, 구글 BERT의 정석 출판사 측에 문의를 넣어보려고 한다.)

첫 번째, dynamic masking은 매 학습 단계마다 15% 임의 마스킹 변환을 새로 적용한 방법으로, BERT 모델의 경우 미리 정의한 횟수(10회) 만큼의 랜덤 학습 데이터를 생성한 이후, 해당 데이터를 고정하여 학습에 반복적으로 사용하였음을 이야기하였으며, 더 많은 학습 데이터로 더 많은 step을 학습하기 위해서는 dynamic masking이 중요함을 이야기하였다.

딥러닝 사전학습 언어모델 기술 동향

Model input format and Next sentence Prediction

Original BERT에서는 NSP를 사용하였고, 이러한 NSP loss를 보조 목적 함수로 두어 pre-training을 진행하였다.

(NSP에 관한 자세한 설명은 위에 언급된 BERT 논문 리뷰 포스팅 링크를 참고 바란다.)

BERT가 제안된 논문에서는 이러한 NSP loss가 매우 중요한 요소라고 가정하였고, 실제 실험 결과를 통해 NSP를 적용하지 않았을 때의 성능 하락을 보여주기도 했다.

그러나, 최근 진행된 여러 연구들은 이러한 NSP loss의 필요성에 의문을 제기한다. 이를 확인하기 위해 본 연구에서는 다음과 같은 training format들로 실험을 진행하였다.

- Segment-pair + NSP
  - Original BERT에서 사용했던 방법으로, 각각의 input은 여러 문장을 담을 수 있는 segment의 pair를 가지고 있다. Input의 max length는 512이다. NSP가 적용된다.
- Sentence-pair + NSP
  - Segment pair와는 다르게 각각의 input이 한 문장씩의 pair를 가지고 있다. Input이 한 문장씩을 담고 있기에 max length인 512보다 작을 것이기 때문에, batch size를 늘려 segment-pair와 비슷한 수의 token개수로 train 될 수 있게끔 한다. 마찬가지로 NSP가 적용된다.
- Full-sentence
  - 각각의 input은 1개 이상의 document에서 연속적으로 sentence를 추출한다. 즉, input에 계속해서 corpus의 document를 채우고, 만약 max length인 512보다 짧은데 document가 끝나면 다음 document를 이어서 input에 채운다. 이때 document 사이에는 [SEP] token을 추가해주고, NSP를 적용하지 않는다.
- Doc-sentence
  - Full-sentence처럼 document에서 연속적으로 sentence를 추출하지만, document를 넘나들지 않는다. document 끝부분에서 추출된 input의 경우 max length인 512보다 작을 수 있기에, batch size를 늘려 Full-sentence와 비슷한 수의 token 개수로 train 될 수 있게 한다. NSP를 적용하지 않는다.

위와 같은 training format들로 실험한 결과는 다음과 같다.

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).

먼저, segment-pair와 sentence-pair를 비교한다. 둘 다 NSP가 적용되었지만, sentence-pair는 single sentence로 구성되어있다. Sentence-pair의 성능이 더 낮은데, 이를 통해 개별 문장의 사용이 downstream task에서의 성능에 악영향을 준다는 것을 확인할 수 있다. 저자들은 이러한 현상의 원인으로 **개별 문장의 사용으로 인한 long-range dependency 학습의 부족**을 지목한다.

다음으로는 Doc-sentence의 결과를 보자. Original BERT에서 사용되었던 segment-pair를 적용했을 때보다 성능이 향상됨을 확인할 수 있는데, 이를 통해 **NSP의 제거가 downstream task에서의 성능이 향상함을 확인할 수 있다.**

저자들은 **BERT 논문에서 NSP를 제거했을 때의 성능 하락은 segment pair input format은 유지한 상태로 NSP만 제거**했기 때문에 발생했을 것이라 한다.

또한, Full-sentence보다 Doc-sentence가 미미하게 좋은 성능을 보이지만, batch size가 변경되는 불편함으로 인해 이후의 실험들에서 Full-sentence를 사용한다고 한다.

Training with large batches

Neural Machine Translation(NMT)에서의 기존 연구들은 learning rate를 적절히 증가시킬 때, 큰 batch size로 training 하는 것이 optimization speed와 end task에서의 성능 모두 향상시킴을 보여왔다. 또한 최근에 진행된 연구들 중에서는 BERT가 large batch training에도 적합함을 보이는 연구도 존재했다.

Original BERT에서는 1M step, 256 batch size로 학습하였는데, 이는 gradient accumulation을 통한 125K step, batch size 2K 혹은 31K step, batch size 31K의 training과 computational cost가 같다.

그래서 저자들은 BERT의 batch size를 늘리며 perplexity와 end task에서의 성능을 비교해봤다. 결과는 다음과 같다.

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.

결과를 통해 **large batch는 perplexity와 end task에서의 성능 모두 좋은 영향**을 미친다는 것을 알 수 있었다. 이러한 대규모의 batch는 병렬화할 때에도 이점이 있기에, 저자들은 이후의 실험들에서 8K의 batch size를 사용한다고 말한다.

(또한, 저자들은 l**arge batch training이 gradient accumulation을 통해 large scale parallel hardware가 없어도 훈련 효율성을 극대화**할 수 있다고 덧붙인다.)

Text encoding

Byte-Pair Encoding(BPE)는 subword segmentation을 수행하는 방법론이며, training corpus의 통계적인 분석을 통해 추출된 subword로 vocabulary를 구성한다.

BPE로 구성된 vocabulary는 일반적으로 10K~100K의 크기를 가진다. 그런데, 크기가 크고 다양한 corpus로 modeling 할 때 unicode character가 vocabulary에서 상당한 양을 차지할 수 있다. 논문에서는 GPT-2 논문에서 제안된 Byte-level BPE를 언급하면서, 이를 통해 OOV를 제거할 수 있다고 말한다. 그러나, vocabulary size 가 50K 정도로 증가하고 성능도 미미하게 감소하였지만, 추가적인 **전처리 과정 없이 universal 하게 적용할 수 있는 encoding 방법**임을 강조한다.

RoBERTa

이번 section에서는 **이제까지 살펴본 dynamic masking, NSP 제거, large mini batch, Byte-level BPE를 통합하여 training을 진행하고, 그 영향을 평가**한다. 저자들은 이러한 configuration을 **Robustly optimized BERT approach(RoBERTa)**라고 명명한다.

추가적으로, 저자들은 기존에 진행되었던 연구들에서 덜 강조된 두 가지의 중요한 요소를 조사한다고 한다. 두 가지의 요소는 다음과 같다.

- **Pre-training을 위해 사용되는 data**
- **number of training passes through the data**

(예를 들어, XLNet의 경우 BERT에 비해 거의 10배의 data로 pre-training을 진행하였으며, 8배 큰 batch size와 절반의 optimization step으로 인해 pretraining에서 sequence를 4배가량 더 많이 본다. 저자들은 이러한 측면에서 다른 모델 혹은 RoBERTa 안에서 비교하며 실험을 진행한다.)

RoBERTa는 BERT-LARGE architecture를 기반으로 training되며($L = 24$, $H = 1024$, $A = 16$, 355M params) 데이터셋 추가 여부, pre-training 횟수를 바꿔가며 실험한다. 결과는 다음과 같다.

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.

우선, original BERT를 pre-training 한 dataset을 roBERTa에도 동일하게 적용한 결과를 살펴보자. 결과를 통해, RoBERTa는 original BERT-LARGE에 비해 비약적인 성능 향상을 이루어냈음을 확인할 수 있다. 이를 통해 논문에서 제시하는 **design choice의 중요성을 재확인할 수 있다.**

이후, 위에서 언급한 3개의 additional dataset을 pre-training에 적용했을 때 전반적으로 성능이 향상했음을 확인할 수 있다. 이를 통해 **pre-training에서 dataset의 size와 diversity가 중요함을 확인할 수 있다.**

마지막으로, RoBERTa의 pre-training step을 100K에서 300K, 최종적으로 500K로 늘린 결과를 확인해보자. Step이 늘어날수록 성능이 향상되는 것을 확인할 수 있고, 특히 step이 300K일 때와 500K일 때는 XLNet의 성능을 능가함을 확인할 수 있다. 즉, 가장 오랫동안 훈련한 model인 500K의 경우에도 overfitting이 일어나지 않은 것으로 확인되며, **additional training에서 성능 향상을 얻을 수 있음을 확인하게 되었다.**

GLUE result

저자들은 GLUE에서의 2가지 finetuning setting을 고안했다.

첫 번째는 (single task, dev)이며, 이는 RoBERTa를 GLUE의 task마다 주어지는 training data만을 이용하여 개별적으로 fine-tuning 하는 것이다. 이때에는 model ensembling 없이 진행되며, dev set으로 5번 evaluation 한 결과의 중윗값을 report 한다.

두 번째는 (ensemble, test)이며, GLUE leaderboard의 결과를 비교한다.(즉, test set의 결과를 비교한다는 것이다.)

저자들은 다른 model들과 달리 RoBERTa는 multi-task fine-tuning을 수행하지 않았다는 점을 강조한다.

또한 RTE, SST, MRPC subtask에 대해서 pretrained RoBERTa에서 fine-tuning을 시작하지 않고, MNLI single-task model에서 fine-tuning을 시작한 것이 더 좋은 결과를 냈다고 한다.

결과는 다음과 같다.

	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.

우선, (single task, dev) setting의 결과를 확인해보면, 9개 task에서 좋은 성능을 내는 것을 확인할 수 있다. 또한, 저자들은 RoBERTa가 BERT-LARGE와 같은 MLM pre-training object를 사용하고, architecture도 같지만 BERT-LARGE의 성능과 XLNet-LARGE의 성능을 넘는 것에 대해 주목한다.

이어서, (ensemble, test) setting에서 4개의 GLUE task(MNLI, QNLI, RTE, STS-B)에서는 SOTA를 달성한 것을 확인할 수 있다.

SQuAD result

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.

RoBERTa는 original BERT나 XLNet과 달리 pre-training에서 추가적인 QA dataset을 사용하지 않고, 바로 SQuAD에 대해 fine-tuning을 진행했음에도 더 좋은 성능을 확인할 수 있다.

RACE result

RACE task에서도 Middle school data와 High school data 모두에서 BERT-LARGE나 XLNet-LARGE 대비 더 좋은 성능을 확인할 수 있다.