[전체 논문 리뷰] BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding

*본 템플릿은 DSBA 연구실 이유경 박사과정의 템플릿을 토대로 하고 있습니다.

1. 논문이 다루는 Task

- **Task:** 양방향 트랜스포머 인코더를 활용해 Pretrained 모델 fine-tuning 하여 각종 Task에 적용할 수 있는 Transformer 모델

기존의 모델들은 사전학습된 language representation을 활용하여 다운스트림 Task에 적용한다. 현재 BERT는 GPT와 마찬가지로 큰 대규모 데이터셋에 학습된 Pretrained BERT 모델을 각기 다른 Task Dataset에 마지막 레이어를 추가해 fine-tuning 하여 각종 GLUE에 있는 task에서 압도적인 SOTA 성능을 달성하였다.

💡 상세 설명 :

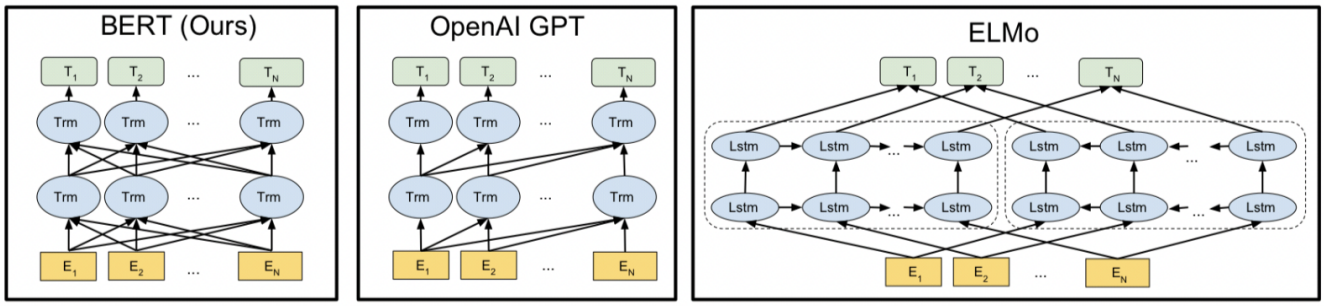
- BERT는 Transformer의 인코더만 사용하여 GLUE 데이터셋에 대하여 SOTA 성능을 달성하였다.
- 모델을 각기 다른 Task에 fine-tuning 할때 Task에 맞는 레이어를 맨 마지막에 추가시켜 fine-tuning을 진행한다.
- Pretrained 모델을 학습 할때 Unsupervised learning을 통하여 대규모 텍스트 데이터를 통하여 학습을 진행하였다.
- Pretrained 모델의 학습은 **MLM** 방식과 **Next sentence prediction**을 채택하여 두가지 방법으로 학습을 진행하였다.

2. 기존 연구 한계

1 기존 학습 방식

기존의 pre-trained 모델을 down-stream 할때 $$feature-based$$ and $$fine-tuning$$ 방식으로 진행하였다.

Unsupervised Feature-based Approaches

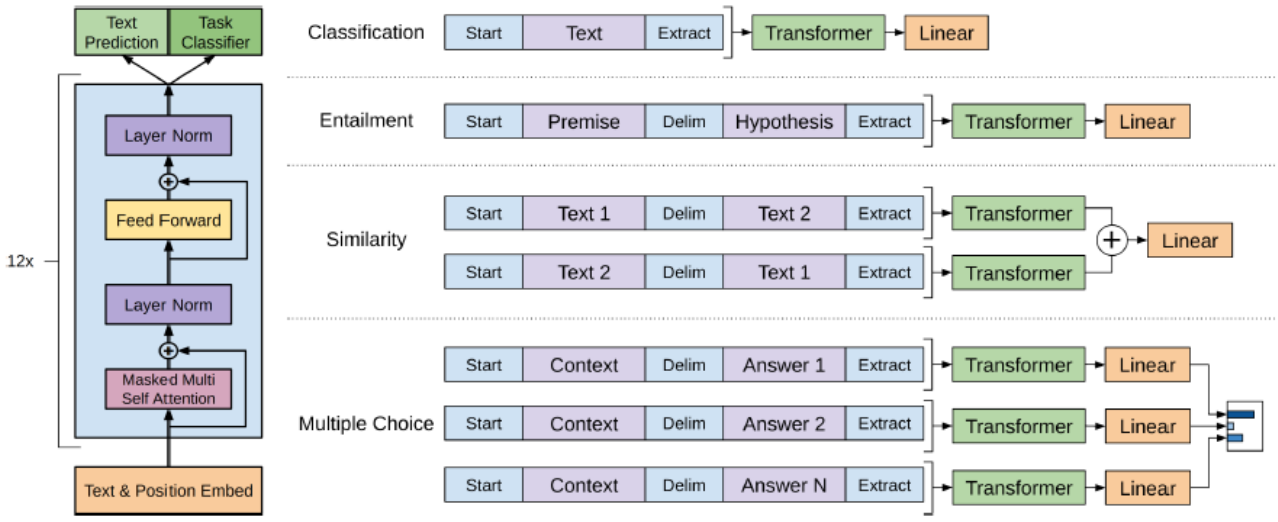


$$feature-based$$ 방식의 대표적인 모델로는 ELMo가 있다. ELMo는 양방향인 Bi-LSTM을 사용하여 양방향 정보를 모아서 문맥의 정보를 추출하여 임베딩 벡터를 생성하게 된다. ELMo의 장점으로 임베딩 벡터를 학습하는데 중점을 두었다고 볼 수 있는데

> Str1 : 나 **배**가 고파 Str2 : 저기 **배**가 지나간다.

위와 같이 다른 **배**의 표현일지라도 Word2Vec과 같은 모델은 같은 의미라고 해석하지만 문맥을 파악하기 때문에 ELMo모델은 서로 다른 벡터로 구분하여 의미가 다른것을 표현하게 된다.

Unsupervised Fine-tuning Approaches



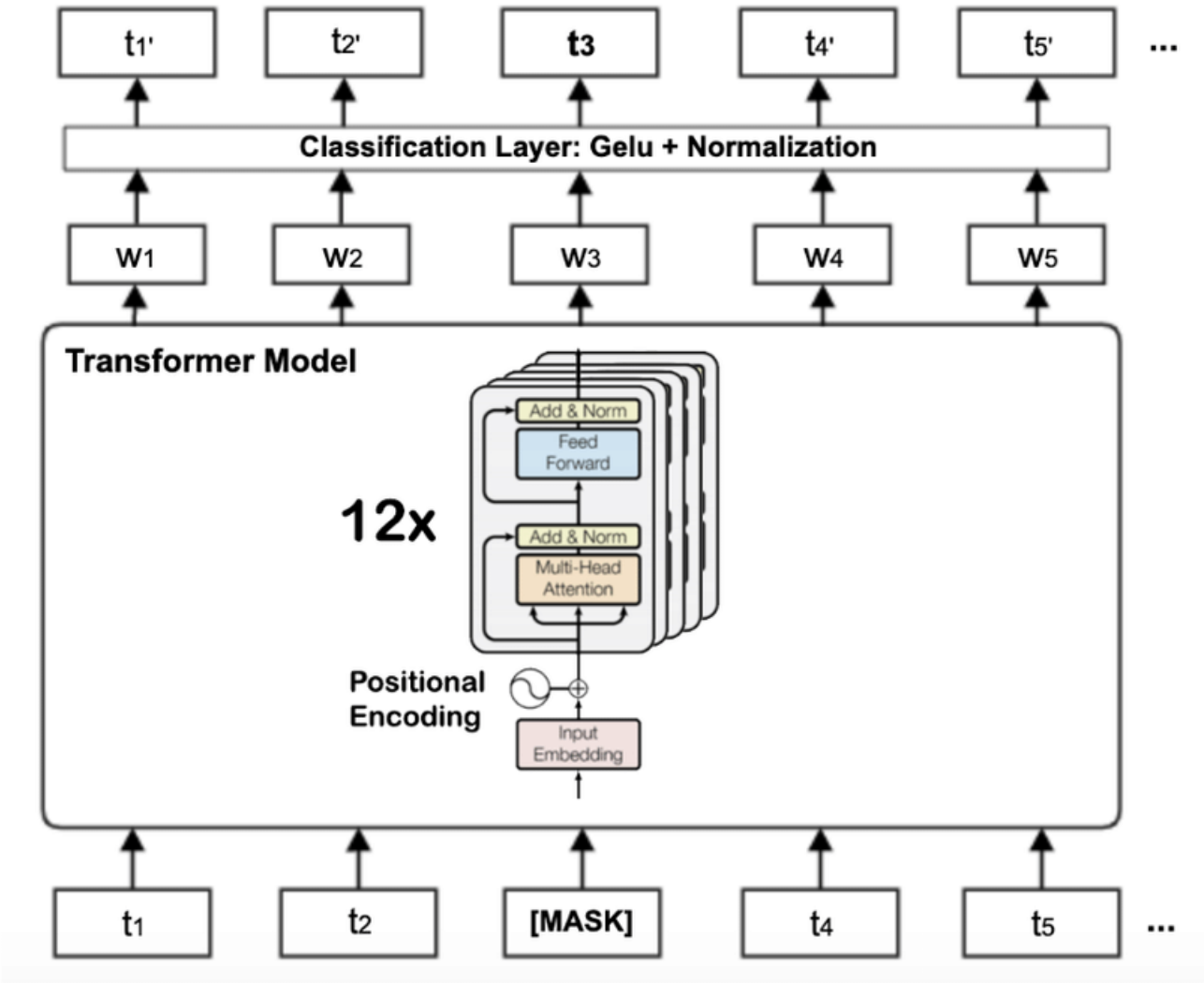
Fine-tuning의 대표적인 방식으로는 GPT-1 모델이 존재한다. 최소한의 파라미터의 도입으로 모든 매개 변수를 미세조정하여 다운스트림 작업에 훈련된다. GPT는 Transformer의 Decoder Block만 활용하여 모델을 구성하였으며 초기 GPT-1과 같은 모델은 현재 논문에서 소개하는 BERT와 같이 다운스트림때 마지막 레이어만을 수정하여 Task를 진행하게된다.

2 단방향 구조의 한계

BERT가 나온 논문 당시에는 GPT-1은 Transformer의 디코더만 사용하게 되었는데 현재 논문에서 주장하는 바에 따르면 GPT는 디코더만을 사용했음으로 단방향으로만 학습을 진행하게 되는데 이때 단방향은 양방향에 비해 fine-tuning 방식에서 pre-trained 모델의 성능을 제한한다고 주장한다. 특히 디코더의 Attention은 이전 토큰에 대해서만 주의를 줌으로써 정보를 통합하는데 중요한 Sentence-level tasks, Question Answering task에서 매우 안좋은 영향을 준다고 주장한다. 또한 일종의 양방향 모델인 ELMo 또한 Bi-LSTM을 사용하지만 진정한 양방향의 학습이 아니라고 주장한다.

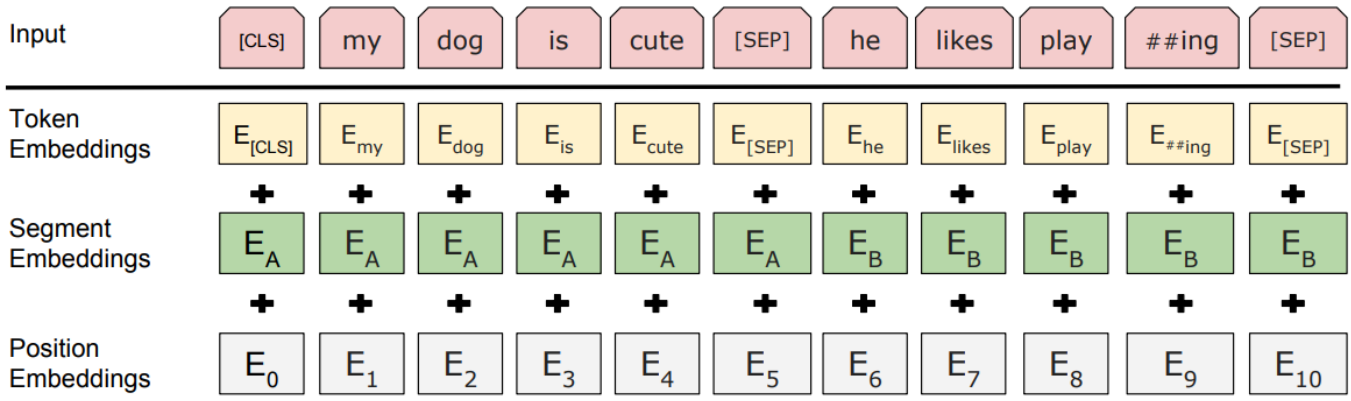
3. 제안 방법론:

Model Architecture



BERT는 Transformer의 Encoder Block만 사용하며 이는 Transformer의 Encoder Block과 동일하게 일치한다. 위의 그림 예시로는 각 $$w_s$$ 의 벡터가 나오게 되는데 $$w_1$$의 벡터는 일종의 [CLS] token으로써 모든 문장의 문맥을 가지고 있다고 해석된다. 이때 사용자는 각 Task에 맞게 자신이 사용할 BERT의 output Token과 Classifcation Layer을 정의하여 사용한다.

Input/Output Representations



BERT의 임베딩은 3가지의 임베딩으로 구성되며 각기 하는 역할은 아래와 같으며 이때 사용되는 3가지의 Special Token이 존재한다.

1. Special Token

- [CLS] : 문장의 입력의 맨 앞에 부착되는 Token으로써 BERT의 Output으로써 모든 문맥의 정보를 담고 있다.
- [SEP] : BERT의 Pretraining 방식의 Next sentence prediction 방식에 사용되는 토큰으로써 1번째 문장과 2번째 문장 사이에 부착되는 Token이다. 이는 1번째 문장과 2번째 문장을 구분하는 Token이다.
- [MASK] : MLM Pretraining 방식의 사용되는 Token이다.

2. Embedding Layer

- Token Embeddings : 흔히 아는 토큰 임베딩이며 WordPiece tokenizer를 사용하며 vocab size는 30000이다.
- Segment Embeddings : 1번째 문장과 2번째 문장을 비록 [SEP] Token이 구분하긴 하지만 Token의 임베딩 값에 직접적인 영향을 줄 수가 없다 이를 각기 다른 문장을 표현하기위해 segment embeddings를 추가하여 더해주게 된다.
- Position Embeddings : 트랜스포머와 동일하게 토큰의 위치정보를 표시하게 해주는 embeddings 레이어다.

Pretraining BERT

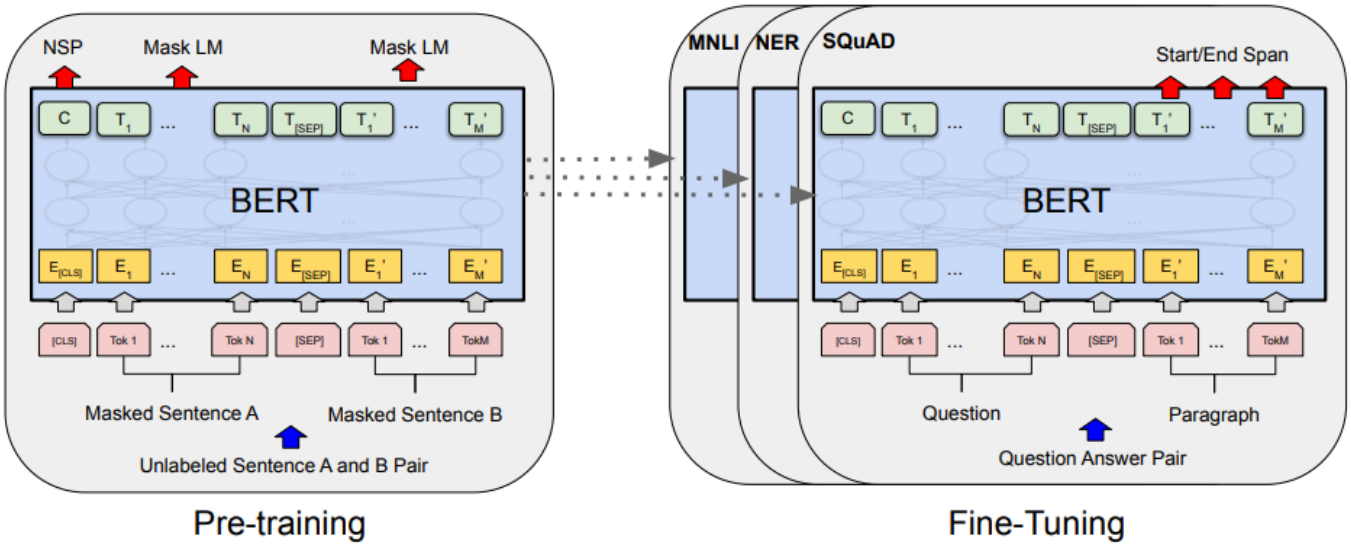
1.Masked LM Masked LM은 입력 문장에서 랜덤하게 일부 단어를 마스크하고, 해당 단어를 예측하는 것으로 학습하며 이때 3가지 유형의 마스크를 사용하게 된다. Masked LM 방식으로 문장의 문맥을 이해하며 단어의 의미를 파악하는 능력을 학습하게 된다. 마스킹 방식은 아래와 같다. 전체 토큰 중 15%를 랜덤하게 고른다. 이때 15% 중에서도 3가지 유형으로 마스킹을 진행한다.

- 첫 번째 유형: 입력 문장에서 랜덤하게 일부 단어를 마스크하는 경우, 이 유형의 확률은 80%이다.
- 두 번째 유형: 일부 단어를 마스크하는 대신 랜덤하게 다른 단어로 대체하는 경우, 이 유형의 확률은 10%이다.
- 세 번째 유형: 마스크하지 않고 그대로 둔 단어를 임의의 확률로 마스크하는 경우, 이 유형의 확률은 10%이다.

2.Next sentence prediction Next sentence prediction는 두 문장이 주어졌을 때, 두 번째 문장이 첫 번째 문장의 다음 문장인지 아닌지를 예측하는 것으로 학습한다. 이를 통해 모델은 문장과 문맥의 의미를 파악하는 능력을 배우게 된다. 이때 문장의 유형은 2가지이며 50%의 확률로 결정된다. 유형은 아래와 같다

- 첫 번째 유형: 첫 번째 문장과 두 번째 문장은 원본 문장의 이어지는 문장이다.
- 두 번째 유형: 첫 번째 문장과 두 번째 문장은 원본 문장에서 이어지지 않는 관계 없는 문장이다.

Fine-tuning BERT

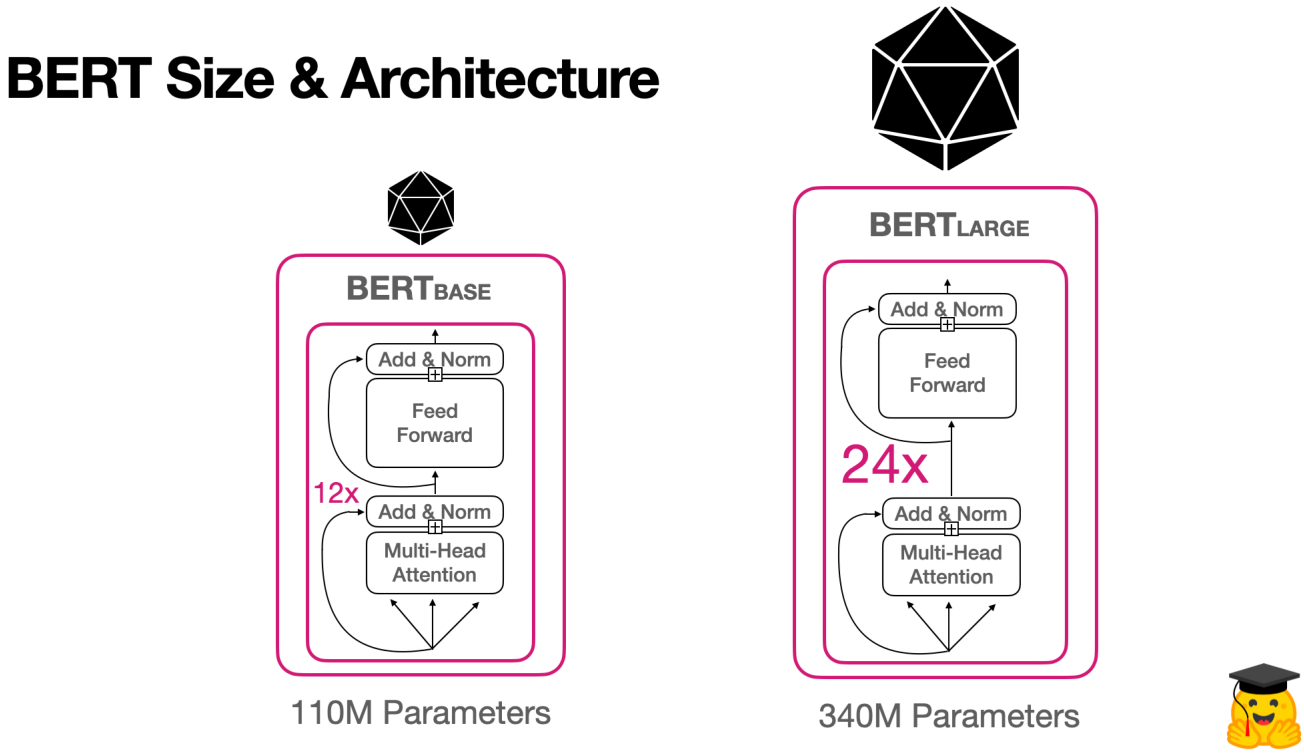


BERT의 Fine-Tuning은 사용되는 text pair을 이루어지더라도 매우 간편하게 학습을 진행하게 된다. 기존의 텍스트가 pair을 이루고 있는 경우 텍스트를 독립적으로 인코딩 한 후 이를 cross attention으로 처리하지만 BERT는 이미 pretraining 과정에서 두 문장을 입력받아 self-attention을 bidirectional하게 진행하여 적용이 매우 간단하게 된다. text pair의 목록은 아래와 같다.

- 같은 의미를 갖지만 다른 단어들로 구성된 문장
- 가설과 전제의 pair 문장
- 질문-답변의 pair 문장
- 텍스트 분류나 태깅에 사용되는 no pair 문장

pair가 존재하는 task에서 output으로 사용되는 token은 output layer의 최종 토큰들에서 각 토큰들의 위치에 맞는 토큰 표현이 토큰 단위 테스크 Layer에 그대로 입력 된다. 이러한 태스크는 태깅, Question Answering 작업이 표현된다. [CLS] 토큰들을 사용하는 Task는 주제 분류, 감성분석등 문장 단위의 테스크에서 사용된다.

BERT Model Size



BERT의 Model Size로는 2개의 모델이 존재한다 각 모델의 파라미터는 아래와 같다.

1. $$BERT_{base}$$

- num of layers : 12
- hiddien size : 768
- num of attention head : 12
- total parms : 110M

2. $$BERT_{Large}$$

- num of layers : 24
- hiddien size : 1024
- num of attention head : 16
- total parms : 340M

✨Contribution

- Transformer의 인코더만을 사용하여 양방향 학습을 진행하게 되었으며 MLM과 NSP의 테스크를 통시에 학습하여 문맥과 단어를 모두 효과적으로 학습할수 있게 되었다.
- 3가지의 다른 임베딩 레이어와 스페셜 토큰을 사용하여 문장의 위치정보와 단어의 위치정보 그리고 문맥의 정보를 담아낼수있게되었다.
- fine-tuning시 text pair에 대한 cross attention의 사용을 제거하였으며 기존의 pretrained 모델을 사용하여 마지막 레이어만을 교체하여 매우 빠르고 효과적이게 task에 맞게 학습을 진행할수있다.

4. 실험 및 결과

GLUE

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	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

GLUE 데이터셋을 사용하여 평가를 진행하였으며 각 테스크 마다의 다른 learning rate를 사용하였다 $$BERT_{Large}$$모델의 같은 경우 데이터셋이 작을경우 학습이 불안정하여 학습을 무작위로 종료시키고 재시작하여 동일 체크포인터에서 데이터 셔플링을 다르게 진행하여 학습을 진행하였다고 한다. score를 보게될경우 $$BERT_{Base}$$만으로도 모든 Task에서 SOTA를 달성하였으며 $$BERT_{LARGE}$$는 더욱 높은 성능으로 SOTA를 기록하여 기존 SOTA에 비해 말도 안되는 성능 향상을 이루어 내었다.

SquAD

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$BERT_{BASE}$ (Single)	80.8	88.5	-	-
$BERT_{LARGE}$ (Single)	84.1	90.9	-	-
$BERT_{LARGE}$ (Ensemble)	85.8	91.8	-	-
$BERT_{LARGE}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$BERT_{LARGE}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$BERT_{LARGE}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

질의 응답 데이터셋으로 위키피디아의에서 답변 텍스트 범위를 예측하는 task이다. 출력으로 답변의 범위의 시작점과 종료점을 찾게 설계하였다. 이때 start vector($$S$$)과 end vector($$E$$)는 $$E∈R^{H}$$ 차원으로 구성되며 $$H=hidden$$$$size$$이다. 단어 $$i$$ 가 시작일 확률은 $$T_{i}$$와 $$S$$을 내적하여 softmax를 통과하여 구하게 된다. $$P_i=\frac{e^{S \cdot T_i}}{\sum_j e^{S \cdot T_j}}$$으로 구성되며 목적 함수는 $$S·T_{i} + E·T_{j}$$가 최대화 되도록 수행된다. SQuAD v1.1에서도 SOTA를 달성하였으며 SQuAD v2.0에서 답변이 없는 본문에 대해서는 시작 종료 모두 [CLS] 토큰을 선택하였다.

SWAG

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$BERT_{BASE}$	81.6	-
$BERT_{LARGE}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

SWAG는 주어진 문장과 이어지는 문장을 객관식으로 선택하는 상식 추론 데이터셋인데 이때 [CLS]토큰과 $$C$$라는 벡터를 추가하여 이 둘을 내적하여 softmax에서 최댓값을 뽑아내는 방식으로 훈련되었다. 이 또한 SOTA를 달성하였으며 1명의 전문가 사람보다 높은 점수를 기록하였다.

Training Details

(1) GLUE

- batch_size : 32
- learning rate : (among 5e-5, 4e-5, 3e-5, and 2e-5)
- epochs : 3

(2) SquAD

- batch_size : 48
- learning rage : 5e-5
- epochs : 2

(3) SWAG

- batch_size : 16
- learing rage : 2e-5
- epochs : 3

회고

> 역시 BERT는 미친 성능을 보여주며 모든 각종 Task에서 SOTA를 찍은 모델이다. 대단하다고 생각되며 심지어 사용하기도 매우 간편해서 더욱 잘만들었다고 생각한다. 무조건 읽어야할 NLP 필수 논문답다고 생각이 든다. 드디어 미루고 미루던 BERT논문 리뷰를 진행하여 뿌듯하고 하루 빨리 GPT를 읽고 작성해야겠다. 모델 리서치가 꿈으로써 하루 빨리 다양한 논문리뷰를 진행해야겠다. 그리고 멀티모달도 얼른 하고싶다 ^,^