**BERT: Pre-training of Deep Bidirectional Transformers for**

**Language Understanding 논문리뷰**

국민대학교 BIT대학원 DELab. 석사과정 이승건

1. **Introduction**

PLM(Pre-trained Language Model)은 많은 NLP task에서 효과적임이 증명되었다.

pre-trained language model을 downstream task에 적용하는 방법에는 크게 **두 가지 방법**이 있는데, feature-based와 fine-tuning이다. 이 방법론은 2번에서 설명한다.

BERT는 Google에서 발표한 논문에서 제안된 모델이며, Bidirectional Encoder Representations from Transformer의 약자이다. 다양한 NLP task에서 혁신적인 성능 향상을 가져왔다고 한다.

*단방향 문맥 이해는 양방향 문맥(contexts)를 모두 반영하지 못한다는 문제가 있다. 예를 들어 Question Answering과 같이 양방향 맥락을 통합하는 것이 중요한 task에서는 좋은 결과를 도출하기 힘들다고 생각할 수 있다.*

이러한 한계를 넘어설 수 있는 **BERT**는 MLM(Masked Language Model)을 pre-training object로 사용하면서 해결하였다. MLM은 3. BERT step 2. 에서 설명한다.

1. **Related Work**

**Unsupervised Feature-based Approaches, 비지도 특징 기반 접근**

ELMo(Embeddings from Language Models)는 bidirectional(양방향) LSTM 기반의 언어 모델을 활용하여 문맥을 반영한 단어 임베딩을 생성한다. ELMo는 left-to-right와 right-to-left 두 개의 언어 모델을 각각 학습한 후, 이 두 방향의 표현을 결합하여 최종적인 context-sensitive representation을 생성한다. 사전 학습된 model의 parameter는 update하지 않고, 단순히 feature extractor로 사용한다.

**Unsupervised Fine-tuning Approaches, 비지도 미세 조정 접근**

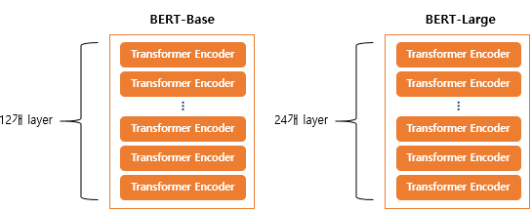
GPT를 예로 들 수 있는데, 최소한의 Task-Specific parameter만을 추가하고, Downstream task들에서 이전에 사전학습된 모든 parameter를 간단히 Fine-tuning함으로써 할 수 있다. Parameter 수가 적다는 장점이 있다.

**Transfer Learning from Supervised Data, 지도학습 데이터로부터 전이 학습**

NLI(Natural Language Inference)나 MT(Machine Translation)과 같이 대규모 데이터셋을 이용한 지도학습에서 전이학습이 효과적임을 보여주는 이전 연구가 존재한다.

1. **BERT**

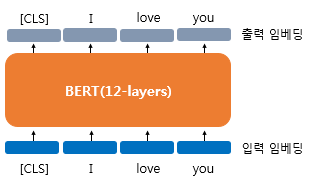
**BERT의 기본 구조**는 다음과 같다.



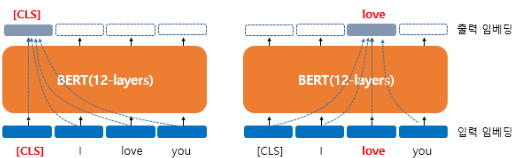
BERT는 기본적으로 Transformer Encoder를 쌓아올린 구조이다. 버전에 따라 크기가 다른데, BERT-Base는 그 당시 GPT-1과 성능을 비교하기 위해 동등한 크기로 설계했다고 한다. Layer **L**, d\_model **D**, Self-Attention Head **A** 수는 각각 다음과 같다.

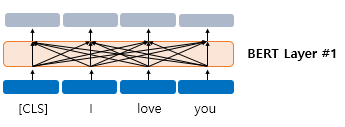
* BERT-Base : L=12, D=768, A=12 : 1억 1천만개의 parameter
* BERT-Large : L=24, D=1024, A=16 : 3억 4천만개의 parameter

BERT는 ELMo나 GPT-1과 마찬가지로 contexts를 반영한 embedding(contextual embedding)을 사용한다.

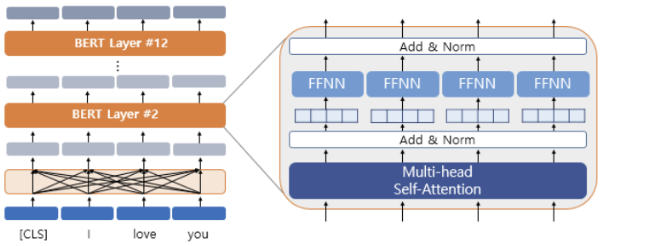


BERT-Base의 경우 d\_model = 768이므로 모든 단어들은 768차원의 Embedding vector가 되어 BERT의 입력으로 사용된다. BERT는 내부연산을 거쳐 동일하게 각 단어에 대해 768차원의 벡터를 출력한다. 위의 그림에서는 BERT가 각 768차원의 [CLS], I, love, you라는 4개의 벡터를 입력받아서(input embedding) 동일하게 768차원의 4개의 벡터를 출력하는 모습(output embedding)을 보여준다. *[CLS]토큰은 sentence의 시작을 알려주는 토큰이며, 분류 task에서는 sentence의 정보를 담아 최종적으로 label을 알려주는 역할을 한다.*



BERT의 연산을 거친 output embedding은 sentence contexts를 모두 반영하게 된다. 

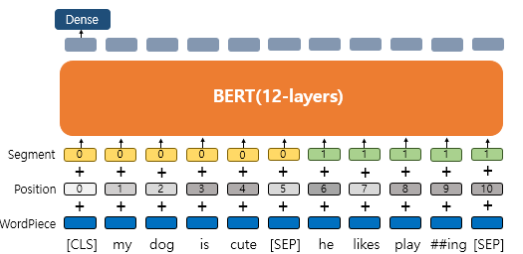
하나의 단어가 모든 단어를 참고하는 연산은 BERT의 12개 층에서 모두 이루어지는 연산이다. BERT의 첫번째 층의 output embedding은 BERT의 두번째 층 input embedding이 되는 것이다.



모든 Encoder Layer에서 Multi-Head Self Attention – 잔차연결 & 정규화 – Feed Forward 신경망 – 잔차연결 & 정규화 – output embedding을 거쳐 contextual embedding을 출력한다.

언급했던 MLM(Masked Language Model)에 대해 좀 더 자세히 설명해보려고 한다.

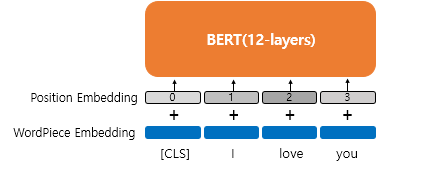
**Step 1. input representation = Token embedding + Segment embedding + Position embedding**



**Token embedding** - BERT에서는 WordPiece tokenization을 사용한다. 바이트 페어 인코딩(Byte Pair Encoding, BPE)와 유사 알고리즘이다. Text를 단어가 아닌 sub-word 단위로 분할하는데, 빈도가 높은 단어일수록 분할되지 않고 그 자체로 vocabulary에 추가된다. 반대로 빈도가 낮은 단어는 더 작은 sub-word로 분할되어 vocabulary에 추가된다. 따라서 WordPiece embedding vector 종류는 vocabulary의 크기와 같다.

**Segment embedding –** 여러 sentence를 연결하여 입력값으로 사용하는 경우, sentence 사이에 구분 토큰 [SEP]를 추가한다. 각 sentence의 토큰을 [SEP]을 기준으로 어떤 sentence에 속하는지 알려주는 embedding이다.

**Position embedding –** Transformer에서는 cos/sin함수의 값을 embedding vector에 더하는 방식이었지만, BERT에서는 학습을 통해 얻는 방법을 사용한다.



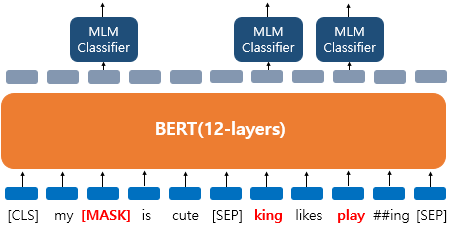
위 그림을 참고하여 예시를 들어보면, 문장의 길이가 4라면 4개의 Position embedding vector를 학습시킨다. 그리고 BERT의 입력마다 다음과 같이 Position embedding vector를 더해주는 것이다. BERT에서는 문장 최대의 길이를 512로 하고 있으므로, 512개의 vector가 학습된다.

**Step 2. pre-training: MLM & NSP**

**Masked Language Model(MLM) –** 입력에서 무작위 15%의 token을 masking 후 주변 단어를 사용해 masking된 token을 예측한다. BERT가 단어의 양방향 문맥, 단어간 관계, 문법이나 문장 구조 등을 학습한다.

Masking된 15%의 token들은 아래와 같이 구성된다.

* 80%: 토큰을 [MASK] 토큰으로 변경한다.  
  Ex) My dog is cute. he likes playing → My [MASK] is cute. he likes playing
* 10%: 토큰을 랜덤하게 다른 토큰으로 변경한다.   
  Ex) My dog is cute. he likes playing → My dog is cute. King likes playing
* 10%: 동일한 토큰으로 그대로 남겨둔다.  
  Ex) My dog is cute. he likes playing → My dog is cute. he likes playing



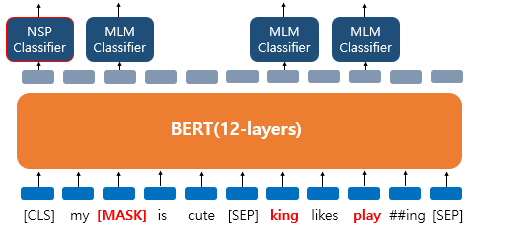
모델이 [MASK] token에 대해서 잘 예측하도록 학습시키는 것뿐만 아니라 들어온 모든 token에 대해 실제 단어의 예측, 즉 원본이 맞는지 확인하게 만드는 것이다.

이를 통해 pre-training과 fine-tuning간 입력 token의 불일치 문제를 해결하면서 모델의 언어 이해력을 더 향상시켰다.

**Next Sentence Prediction(NSP) –** 두 문장 A, B가 이어지는 문장인지 이진분류

BERT가 문장 관계를 학습한다. 문장 쌍 학습 데이터셋은 다음과 같이 구성한다.

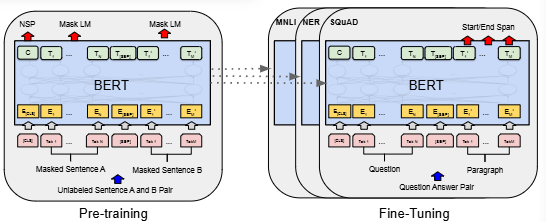
* 50%: A와 B가 관련된 문장, A뒤에 B가 따라오는 문장, 'IsNext' 라고 labeling
* 50%: A와 B가 관련 없는 문장, 무작위 선택, 'NotNext' 라고 labeling



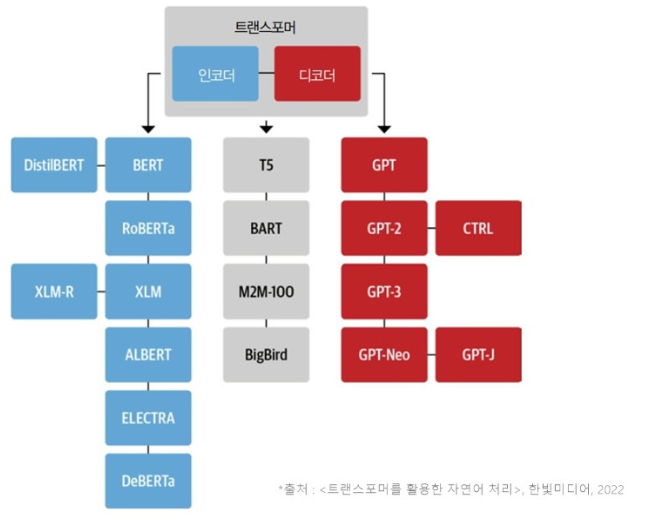
[CLS] token의 final hidden vector C를 이용해 이진 분류한다. C를 Feed-Forward network – Softmax 계층을 통과시켜 isNext, NotNext으로 labeling한다. [CLS] token은 BERT가 분류 문제를 풀기 위해 추가된 특별 token이다. QA(Question Answering)나 NLI(Natural Language Inference)같이 두 문장 관계를 이용하는 것이 중요한 task가 있기 때문이다.

**Step 3. fine-tuning –** 사전학습된 BET를 NLP task에 맞춰 조정하는 과정

BERT의 가중치를 기반으로, task-specific layer나 파라미터를 추가하여 labeled data에서 전이학습을 한다. 전이학습이란, 사전 학습된 모델을 새로운 관력 task에 맞게 조정하는 기계학습 기법이다. Pre-tuning된 BERT를 NLP task에 맞게 조정하는 과정이다.



\*BERT는 GPT에 비해 scale이 작다. BERT는 다양한 task에서 활용될 수 있도록 만든 모델이기 때문에, 변형 모델이 매우 많다. 또한, 학습시 전체 데이터의 15%만 사용하기 때문에, scale를 높여도 계산량 대비 학습 효율이 높지 않다고 한다. 반면 GPT는 생성모델이기 때문에 데이터를 많이 사용할수록 더 정확한 예측이 가능하다.



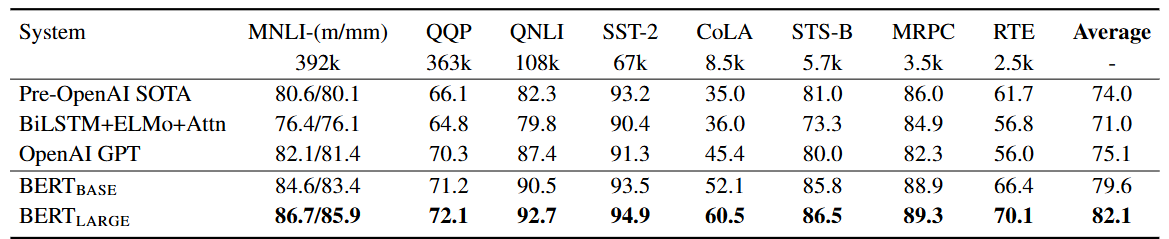
1. **Experiments**

논문에서 11개의 NLP task에 대해 BERT fine-tuning 테스트를 진행했다고 한다.

GLUE의 8개 task(MNLI, QQP, QNLI, STS-B, MRPC, RTE, SST-2, CoLA), SQuAD v1.1, SQuAD v2.0, SWAG을 이용했다.

**GLUE(General Language Understanding Evaluation) benchmark**

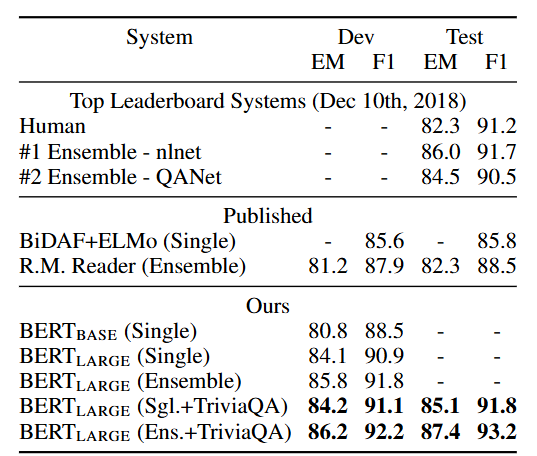
인간의 언어 능력을 얼마나 따라왔는지 정량적 성능지표를 만들어 NLP task의 평가체계를 표준화한 것이다. 총 9개의 NLP task로 언어모델의 성능을 평가한다.



BERT Large가 더 좋은 성능을 보였으며, 특히 학습 데이터셋의 크기가 작을수록 성능이 우위에 있다.

**SQuAD v1.1(Stanford Question Answering Dataset) - 10만개의 QA쌍 데이터셋**

QA가 포함되어 있는 passage가 주어지면 passage에서 어디부터 어디까지가 Answer에 해당되는지 예측하는 모델을 학습한다.

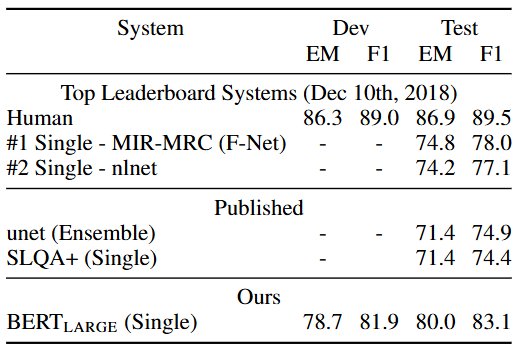


SQuAD v1.1 task 결과에서도 BERT가 우위를 차지했다.

**SQuAD v2.0**

주어진 passage에 Answer가 존재하지 않을 가능성을 추가하여 SQuAD v1.1을 확장한 task이다. SQuAD v1.1 BERT model에 Answer가 존재하는지에 대한 여부를 고려하는 과정을 추가했다.

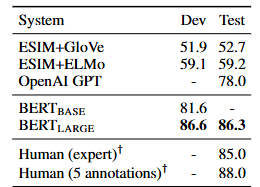
Answer가 존재하지 않는 것은 [CLS] token으로 시작하고 끝나는 Answer span이다. no-answer span의 score와 best non-null span의 score를 비교한다.



확장된 version의 task에서도 BERT의 성능이 가장 높게 측정되었다.

**SWAG(The Situations With Adversarial Generations) dataset**

근거 있는 추론을 평가할 수 있는 약 11만개의 문장 쌍 dataset이다. 문장이 주어졌을 때, 보기로 주어진 4개의 문장 중 가장 잘 어울리는 문장을 찾는 task이다.



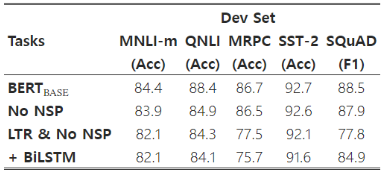
학습 결과, SWAG 역시 BERT의 성능이 매우, 그리고 가장 좋았다.

1. **Ablation Studies**

**Effect of Pre-training Tasks**

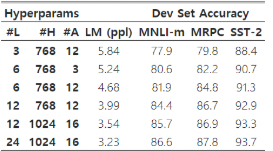
BERT의 2가지 pre-training objectives MLM과 NSP의 중요성을 평가함으로써 BERT의 deep bidirectionality의 중요성을 보였다.

NSP를 사용하지 않았을 때와, MLM대신 단방향(left to right) LM을 사용하는 모든 task에서 성능이 하락했다.



**Effect of Model Size**

Model의 크기가 fine-tuning의 정확도에 미치는 영향을 확인해본다. Layer **L**, d\_model **D**, Self-Attention Head **A** 값에 변화를 준다.



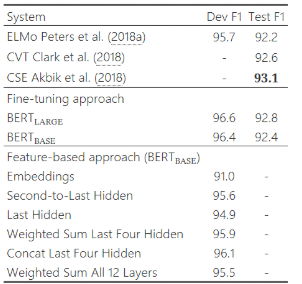
실험 결과 모든 task에 대해 model size가 커질수록 성능도 향상되었다. Model이 충분한 사전학습이 된다면, 매우 크기가 큰 model로 확장하는 것이 작은 규모의 task에서 큰 개선으로 이어질 수 있음을 보여준다.

**Feature-based Approach with BERT**

논문에서 BERT는 fine-tuning approach를 사용하고 있으나 다음의 경우에서 feature-based approach를 사용한 경우에도 장점이 있다고 한다.

* transformer encoder로 표현하기 힘들어 task-specific architecture가 추가되어야 하는 task일 때
* computational benefit: pre-trained된 features를 고정적으로 사용하면서 연산을 줄일 수 있다.

NER(Named Entity Recognition) task에 대해서 feature-based approach를 적용하여 두 approach 방식의 성능을 비교했다. feature-based approach는 fine-tuning을 거치지 않고, BERT의 output을 bidirectional LSTM의 input embedding으로 바로 사용했다. 실험 결과, fine-tuning approach의 성능이 더 좋긴 하지만, feature-based approach와의 차이가 적었다. 따라서 BERT는 fine-tuning approach와 feature-based approach 모두에서 효과적이다.



1. **Conclusion**

**언어모델에서 unsupervised pre-training와 transfer learning은 매우 중요하다.**

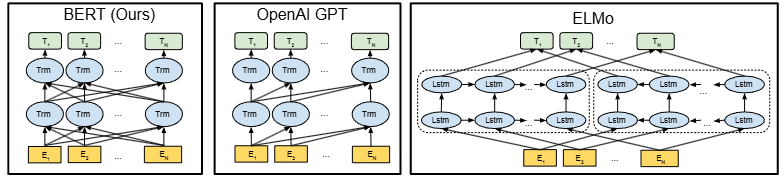
본 논문은 deep unidirectional architecture를 사용함으로써 동일한 pre-trained model이 광범위한 NLP task를 더 성공적으로 처리할 수 있도록 하는데 기여하였다.

1. **Appendix**

**Downstream task** 사전 학습된 모델을 활용하여 수행하는 특정한 목표 작업을 의미한다. BERT를 예로 들면, 다양한 NLP task에 활용할 수 있는데 이때 적용되는 task가 Downstream task이다.

**BPE** Byte Pair Encoding는 자주 등장하는 문자 쌍을 병합하여 sub-word를 학습하는 알고리즘이다. GPT-2 및 SentencePiece의 Unigram 모델에서도 사용된다.

**Token** embedding vector가 tokenization되는 과정이다.



ELMo는 bidirectional connection이라고 해석될 수 있지만 left to right, right to left를 concat했을뿐 한 target에 대해 모든 contexts가 고려되지 않아 BERT에 비해 표현력이 떨어질 것이라 예상한다. RNN의 특성상 이전 시각의 정보들을 담고 있더라도, LSTM은 gate를 지나며 정보가 삭제/추가되기 때문이다.

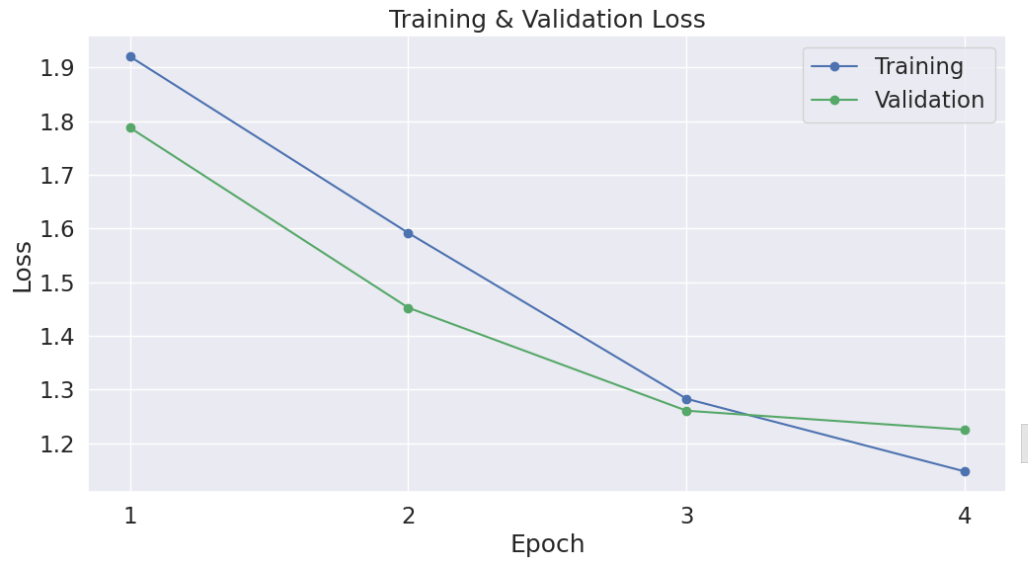
그래서 본 논문을 작성한 저자들도 기존의 bidirectional connection보다 더 깊은 BERT를 deep bidirectional이라 강조했다고 생각한다.

**MLM** Masked Language Model

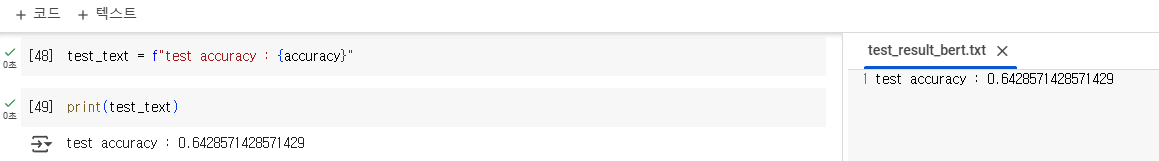
Label data가 적거나 없을 때는 지도학습을 하기 어렵다. NLP task의 특성상 레이블 데이터가 적고, 인간이 dataset을 만들기엔 비효율적이기 때문에 비지도학습을 한다.

하지만 비지도학습의 가장 큰 단점은 레이블이 없기 때문에 성능을 평가하기 어렵고, model이 찾은 패턴이 우리가 원하는 의미와 다를 수 있다. 하지만 **MLM**에서 masking된 단어가 **Pseudo Label**의 역할을 한다. 비지도학습이면서 지도학습처럼 작동할 수 있다는 것이다. Label data가 없어도 거대한 corpus만 있다면 학습이 가능하다는 장점이 있다.

**BERT: Fine-tuning 과제**

**2. bert-base-multilingual-cased Model, Tokenizer**를 이용한 학습 후 Training & Validation Loss

**Test Accuracy: 0.6428571428571429**



**3. 실습 코드에서 학습에 주요한 파라미터**

1. Optimizer: AdamW을 optimizer로 설정했는데, 여러 가지의 optimizer중 모델에 최적화된 optimizer를 고르는 것도 중요하다고 생각한다.
2. Learning Rate: 학습률이 너무크면 발산할 수 있고, 너무 작으면 수렴속도가 느려진다. 학습률은 10단위 곱으로 사용하는 것이 일반적이다. 이 모델에서는 lr = 2\*10^-5(2e-5)를 사용하였다.
3. Batch size: 연산 한번에 들어가는 데이터의 크기이며, 이 모델에서는 batch size를 32로 설정했다. 주로 배치사이즈는 2의 배수(16, 32, 64, 128, 256, …)을 사용한다고 한다. Batch size가 너무 커지면 자원소모 및 학습 속도가 느려지고, Batch size가 너무 작아지면 적은 데이터를 대상으로 가중치 갱신이 이루어지고, 이 갱신이 자주 일어나므로 train이 불안정해질 수 있다.
4. Epoch: 전체 학습 데이터셋을 몇회(몇바퀴) 학습시킬지 정하는 hyperparameter이다. 해당 모델에서는 4epoch으로 설정했으며, epoch을 지나치게 높이면 overfitting의 위험이 있고, epoch이 지나지게 낮으면 underfitting의 위험이 있다.

좋은 모델을 얻기 위해서는, 여러 번의 train과 validation을 반복하며 최적 Loss의 Epoch값을 찾는 것이 맞는 방향이다고 생각한다.

데이터셋의 크기, 모델의 종류에 따라 epoch수는 달라지기 때문에 일반적인 epoch 횟수는 없다고 생각한다.