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

**Language Understanding 논문리뷰**

국민대학교 BIT대학원 DELab. 석사과정 이승건

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

**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.

**Feature-based approach**의 경우 ELMo에서 사용한 방식으로, pre-trained representation을 architecture의 추가적인 feature로 사용하는 방법론이다.

**fine-tuning approach**의 경우 GPT에서 사용한 방식이며, pre-trained parameter 전체를 가지고 downstream task에서 학습하며 feature-based approach와는 다르게 task-specific parameter를 최소화하는 방법이다. **BERT의 등장 배경**이 여기서 등장한다.

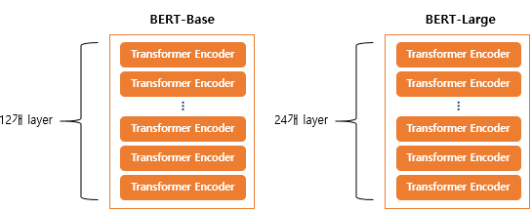
위에서 소개한 두가지 방법의 pre-training 모델은 단방향 모델(unidirectional language model)이였다.

*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.

**BERT**

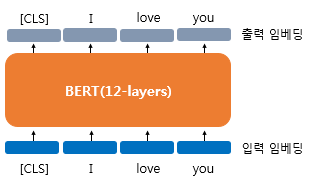
**BERT의 기본 구조**는 다음과 같다.



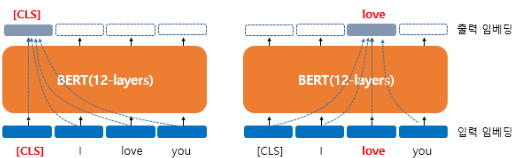
BERT는 기본적으로 Transformer Encoder를 쌓아올린 구조이다. 버전에 따라 크기가 다른데, BERT-Base는 그 당시 GPT-1과 성능을 비교하기 위해 동등한 크기로 설계했다고 한다. Layer **L**, d\_model **D**, Self-Attention Head **A** 수는 각각 다음과 같다.

* BERT-Base : L=12, D=768, A=12 : 110M개의 파라미터
* BERT-Large : L=24, D=1024, A=16 : 340M개의 파라미터

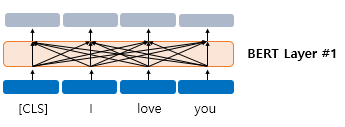
BERT는 ELMo나 GPT-1과 마찬가지로 contexts를 반영한 embedding(contextual embedding)을 사용한다.



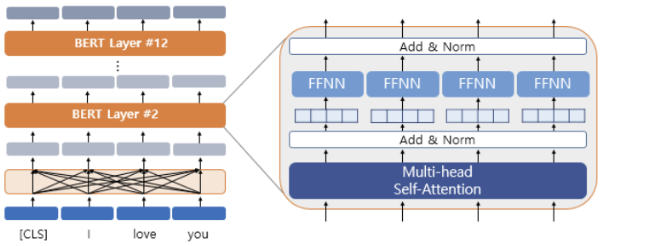
BERT-Base의 경우 d\_model = 768이므로 모든 단어들은 768차원의 Embedding vector가 되어 BERT의 입력으로 사용된다. BERT는 내부연산을 거쳐 동일하게 각 단어에 대해 768차원의 벡터를 출력한다. 위의 그림에서는 BERT가 각 768차원의 [CLS], I, love, you라는 4개의 벡터를 입력받아서(input embedding) 동일하게 768차원의 4개의 벡터를 출력하는 모습(output embedding)을 보여준다. *[CLS]토큰은 sentence의 시작을 알려주는 토큰이며, 분류 task에서는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



BERT의 연산을 거친 output embedding은 sentence contexts를 모두 반영하게 된다.



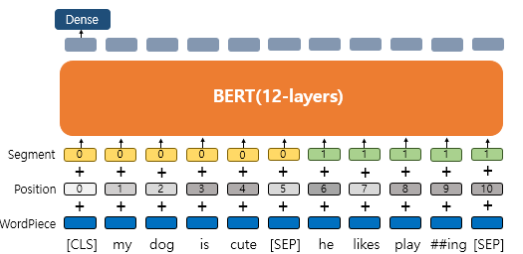
하나의 단어가 모든 단어를 참고하는 연산은 BERT의 12개 층에서 모두 이루어지는 연산이다. BERT의 첫번째 층의 output embedding은 BERT의 두번째 층 input embedding이 되는 것이다.



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.

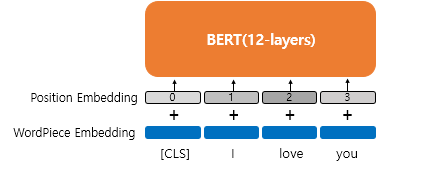
**Step 1. input representation = Token embedding + Segment embedding + Position embedding**



**Token embedding** - BERT에서는 WordPiece tokenization을 사용한다. 바이트 페어 인코딩(Byte Pair Encoding, BPE)와 유사 알고리즘이다. Text를 단어가 아닌 subword 단위로 분할하는데, 빈도가 높은 단어일수록 분할되지 않고 그 자체로 vocabulary에 추가된다. 반대로 빈도가 낮은 단어는 더 작은 subword로 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 방식이였지만, BERT에서는 학습을 통해 얻는 방법을 사용한다.



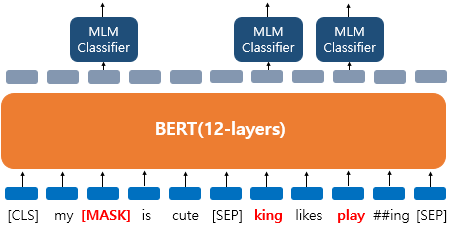
위 그림을 참고하여 예시를 들어보면, 문장의 길이가 4라면 4개의 Position embedding vector를 학습시킨다. 그리고 BERT의 입력마다 다음과 같이 Position embedding vector를 더해주는 것이다. BERT에서는 문장 최대의 길이를 512로 하고 있으므로, 512개의 vector가 학습된다.

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



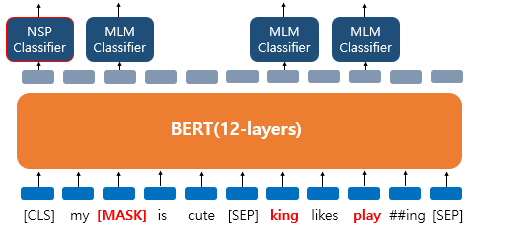
모델이 [MASK] token에 대해서 잘 예측하도록 학습시키는 것 뿐만 아니라 들어온 모든 token에 대해 실제 단어의 예측, 즉 원본이 맞는지 확인하게 만드는 것이다.

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. 문장쌍 학습 데이터셋은 다음과 같이 구성한다.

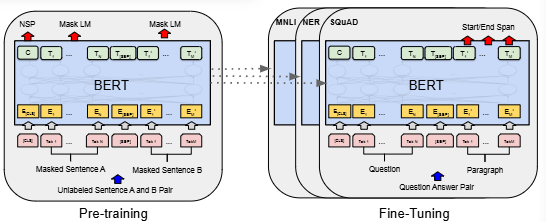
* 50% : A와 B가 관련된 문장, A뒤에 B가 따라오는 문장, 'IsNext'라고 labeling
* 50% : A와 B가 관련없는 문장, 무작위 선택, 'NotNext'라고 labeling



[CLS] token의 final hidden vecor 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.



**Fine-tuning이란? 다른 작업에 대해서 파라미터 재조정을 위한 추가 훈련 과정**

**Pre-training이란?**

**ELMo, Feauture-based와 fine-tuning approach의 정의**

**BPE, WordPiece 토큰화**