**@ BERT**

실제로 transformer 그대로 쓰기 보다는 BERT를 사용함

BERT는 사전 학습된 모델

BERT 이후 다른 사전 학습 모델이 많이 나옴

작년 2018년은 NLP의 획기적인 모델이 많이 나옴

* ELMo, BERT, GPT, ULM-FiT…
* BERT 그동안의 sequential 모델의 패러다임을 바꿈
* 어디에나 쓸 수 있는 pre-training을 하고, fine-tuning으로 다른 모든 task를 수행함  
  (기존에 어려울 것이라고 예측되던 부분)
* 1. 반 지도학습으로 지도학습을 시킨 training한 모델  
  2. 지도학습으로 특정 task를 수행
* Meaning extract는 쉽지 않음  
  텍스트에서 의미 추출은 classification task (긍정, 부정, 중립)
* NLP의 대부분의 task는 classification
* 대부분의 task에서 BERT의 성능이 다른 모델을 능가함

#Sentence Classification

* Input을 입력 받아 사전학습 -> fine-tuning  
  fine-tuning은

BERT base와 BERT large가 있음

* Base는 GPT와 같은 사이즈
* BERT는 학습된 transformer encoder stack
* BERT는 Transformer의 Encoder만 가지고 만듦 (번역 task를 할 것이 아니기에 디코더 필요 없음)
* BERT base는 Encoder가 12개, large는 24개
* Hidden layer도 768과 1024로 증가하였고, multi-head attention도 12개와 16개로 증가
* [CLS] token이 문장 맨 앞에 붙임  
  -> 단어별 특징을 추출하거나 분류하기 위함
* BERT의 사용법만 알 경우, 모든 task를 쉽게 수행 가능함

BERT의 특징

* Transformer의 Encoder만 사용함
* BERT의 사전학습 모델로 특징 추출, 분류 task를 수행
* RNN, CNN의 경우 직접 학습을 시켜야 했지만, BERT는 이미 사전학습 되어 있기에 fine-tuning만 함으로 task를 수행 가능함
* Transformer는 직접 번역기를 만들지 않는 한 사용하지 않음

#Embedding

* 기존에는 one-hot Encoding
* 원-핫 인코딩은 단어 간의 의미관계를 파악할 수 없음
* 주변의 단어를 통해 단어의 의미를 파악하는 것이 word embedding  
  단어를 벡터로 -> word2vec
* 워드임베딩이 텍스트처리에 큰 기여를 함
* 학습 방법은 주변 단어들과의 관계로 학습
* 100차원의 벡터, 10000차원의 벡터 등으로 표현  
  벡터의 차원이 커질수록 의미의 미세한 표현이 가능함

ELMo

* 동의어의 문제를 해결함
* 기존의 워드임베딩은 동의어의 문제를 해결할 수 없었음
* 단어의 context를 학습해서 단어마다 다른 벡터값을 학습함
* 2018년에 나옴
* 3개의 레이어로 문맥을 추측하여 좋은 성능을 냄
* Language modeling으로 이후에 올 단어를 예측
* LSTM 레이어를 쓰고, forward, backward 양방향으로 학습 하고 두 레이어를 concat
* 각 레이어에 각각의 weight를 곱함
* 문장을 넣어야 embedding이 나옴
* 같은 stick일지라도 다른 vector값을 가짐 (다른 문장으로 학습하였기 때문)
* 임베딩으로 나온 값에서 특정 단어의 벡터를 추출함 **(feature extraction)**
* ELMo의 경우 레이어가 올라갈 경우 단어의 context 벡터를 잘 학습함

ULM-FiT도 사전학습을 함

Transformer

OpenAI Transformer

* BERT가 나오기 전에 디코더만을 가지고 학습을 함
* Encoder를 넣지 않았기에 참조값 x

BERT

* 트랜스포머의 **인코더**의 값을 참조
* ELMo의 bidirectional함은 두 방향을 concat한 것
* BERT는 동시에 한 포인트에서 **bidirectional 학습**을 함 (masking으로 가능했음)
* 학습 값이 [CLS]에 담겨 있음
* 15% 확률로 [mask], 다른 단어, 기존 단어로 대체
* 이 단어를 앞, 뒤 단어로 학습을 하기 때문에 성능을 향상시킴
* 이 학습을 함으로써 양 방향 학습이 가능
* 첫 번째, 두 번째 문장 끝에 [SEP] token을 넣음
* 1. [mask]된 값을 train함
* 2. 첫 문장과 두번째 문장이 연속된 문장인지 train함
* BERT 임베딩은 ELMo와 같이 문맥에 따라 다른 의미의 단어의 의미를 알 수 있음
* 마지막 레이어의 학습 값이 가장 좋을지?  
  레이어마다 학습되는 값이 다르기에 모든 레이어의 합이 가장 좋은 학습 값을 가짐  
  다른 task에서는 두번째부터 마지막 레이어의 합이 가장 좋은 학습 값
* Feature-extraction도 가능하게 되어 있음
* 개체명 인식

#BERT 특징

* Wordpiece  
  unknown 토큰을 없애기 위해 토크나이징  
  wordpiece를 어떻게 자르냐에 따라 성능이 달라짐
* 3가지의 토큰 값을 합쳐서 값을 얻음

#Fine-tuning

* 실제 pre-training 모델을 가지고 진행
* cls는 바이너리 클래시피케이션
* 멀티 클래스로 classification을 할 경우 멀티클래스가 될 수 있게 마지막 히든 레이어에 소프트맥스를 씌움
* 모든 task에서 가장 좋은 성능을 보여줌

#XLNet

* Autoregressive LM?  
  이전 두 단어, 혹은 이후 두 단어를 가지고 예측함  
  context word에서 단어를 예측함 (GPT…)  
  일반적 NLP 테스크에 큰 성능  
  하지만 앞, 뒤 동시 학습을 하지 못한다는 특징
* 버트는 오토인코더  
  앞에서도 뒤에서도 [mask]된 단어를 예측함  
  버트의 단점은 [mask]를 임의로 넣고 사전 학습을 함  
  예를 들어 baking crisis의 경우 두 단어가 dependent한데 independent하게 마스킹을 함
* 그래서 permutation을 사용함  
  어떤 포지션이 있으면 서로 permute시키면 전체 토큰수의 팩토리얼 개수 만큼 가능한 경우의 수가 나옴  
  t의 t-1의 토큰으로 학습
* 아이디어는 굉장히 놓음  
  한국어 조사의 경우 이 경우에 해당  
  하지만 모든 경우를 팩토리얼 만큼 계산을 해야 하기 때문에 계산량이 매우 많음

에트리의 KorBERT

* 형태소 분석기
* Classifier와 feature-extraction을 하면 목적에 맞게 쓸 수 있음

SKT의 KoBERT

서울대도 만들고 있음 - 다른 방식으로 (하지만 힘듦 ;;;)

#Fine-tuning 실습

허깅페이스의 프리트레인드 된 웨이트를 가져와서 클래시피케이션을 함