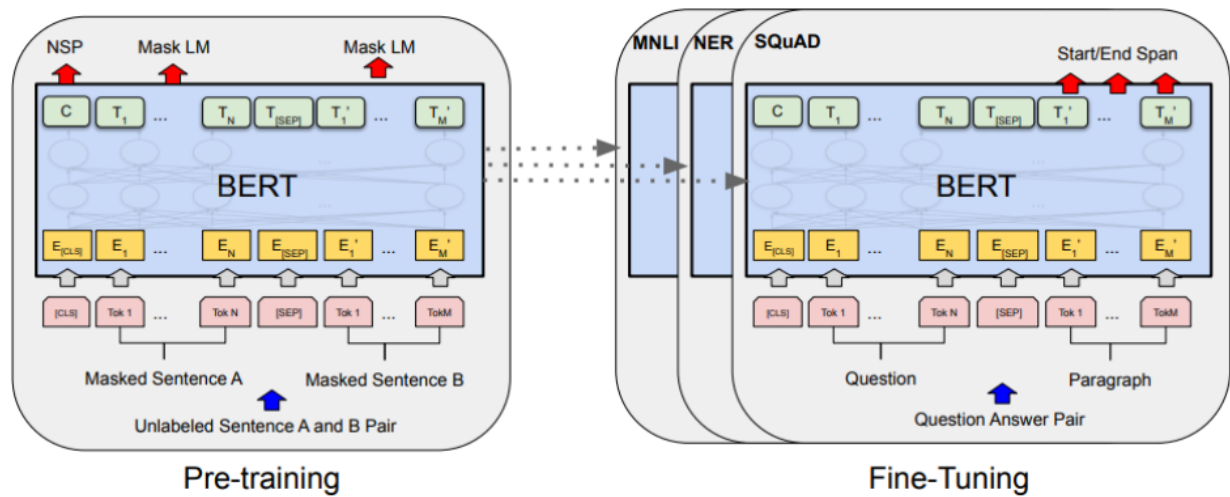


BERT

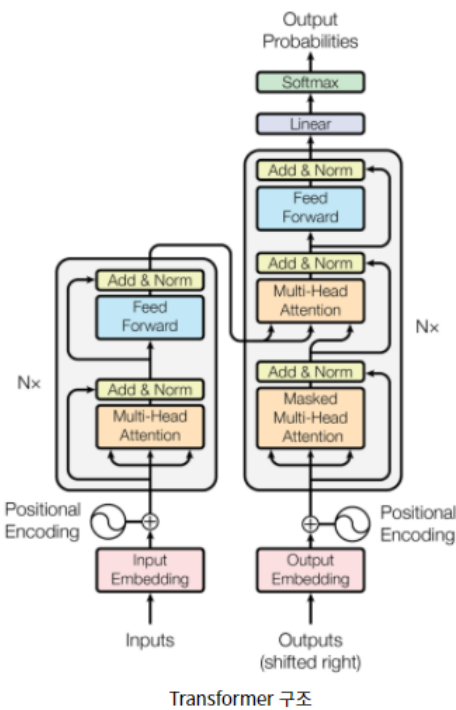
BERT (Bidirectional Encoder Representations from Transformers)

대용량 unlabeled data로 모델을 학습시킨후, 특정 task를 가지고 있는 labeled data로 transfer learning하는 모델
BERT이전에는 language 모델을 학습하고, 뒤쪽에 특정 task를 처리하는 network를 붙이는 방식 (ELMo, GPT ..)
BERT는 새로운 network를 붙일 필요 없이 BERT 모델 자체의 fine-tuning을 통해 task 처리



BERT의 프로세스

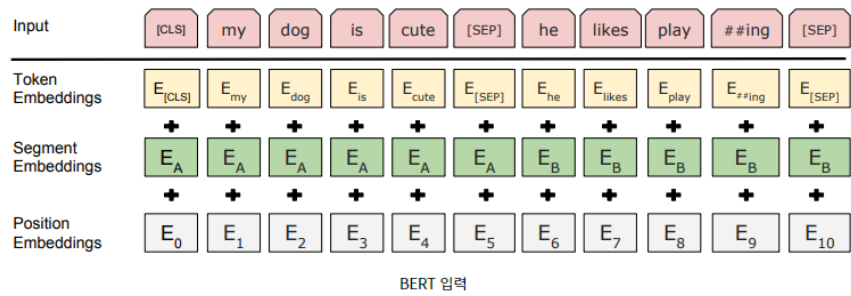
- model architecture
transformer의 encoder 부분만 사용



Transformer 구조

모델의 크기에 따라 분류

- BERT_base : layer = 12, hidden size = 768, self-attention head 수 = 12, 전체 파라미터 = 1.1억 (GPT와 동일한 사이즈)
 - BERT_large : layer = 24, hidden size = 1024, self-attention head 수 = 16, 전체 파라미터 = 3.4억
- Input embedding



3가지 입력 임베딩을 취합해 하나의 임베딩 값을 만들 + layer normalization + dropout 적용 ⇒ 입력

1. token embedding

WordPiece임베딩 방법을 사용 : 자주 등장하면서 가장 긴 길이의 sub-word를 하나의 단위로 만들고, 자주 등장하지 않는 rare-word를 sub-word로 쪼개서 만든다.
 두 가지 특수 토큰 (CLS, SEP)을 사용해 문장을 구별
 기존 임베딩 방법과 다르게 Out-of-Vocabulary(OOV) 문제가 없어짐

2. segment embedding

두 개의 문장을 문장 구분자([SEP])와 합쳐서 입력
 입력 길이는 두 문장을 합쳐서 512 subword 이하로 제한

3. position embedding

self-attention모델을 사용하기 위해 입력 토큰의 위치 정보를 줘야함
 transformer에서는 sinusoid함수를 이용해 positional encoding을 하고, BERT에서는 이를 변형해서 position encoding 사용 (단순히 token 순서대로 0, 1, 2 ..)

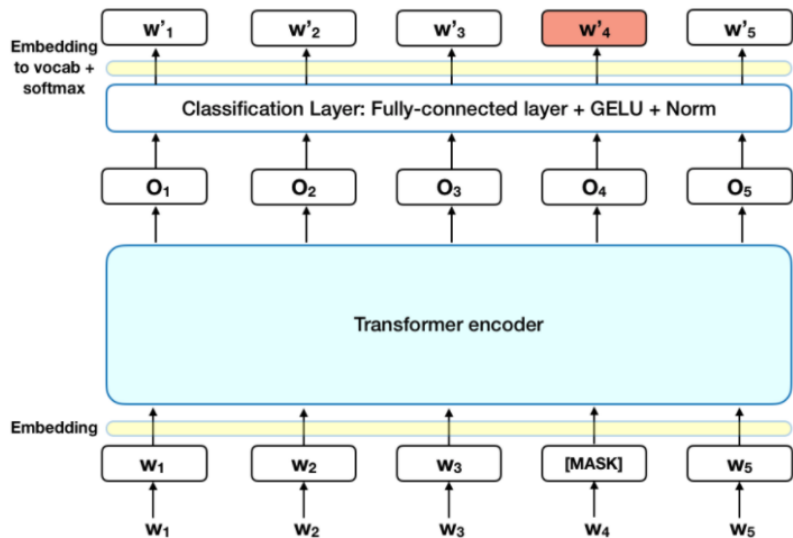
• pre-training BERT : unsupervise 방법 2가지

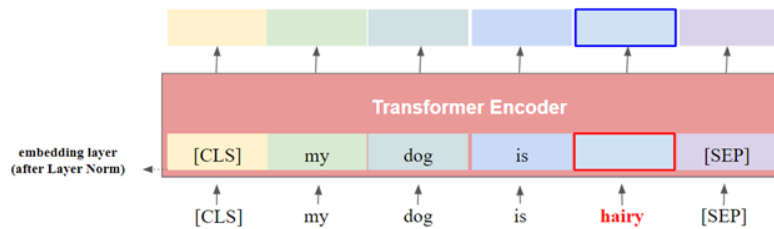
1. 기존 방법

ELMo나 GPT는 language model을 사용 : 앞의 n개 단어를 가지고 뒤의 단어 예측 → unidirectional (단방향) : left to right 이나 right to left

2. Masked Language Model (MLM) 방법

input에서 무작위로 몇개의 token을 mask시키고 transformer의 encoder 구조에 입력하면 주변 단어의 문맥만을 보고 mask된 단어 예측 → 문맥을 파악하는 능력을 기름





Mask **15%** of all WordPiece tokens in each sequence at **random**. (e.g., **hairy**)

- [MASK] → [MASK] 80% of the time : Replace **[MASK]** token.
- apple → apple 10% of the time : Replace the word with a **random** word
- hairy → hairy 10% of the time : Keep the word **unchanged**.

단어의 15%를 mask token으로 변경해서 mask token만을 예측하는 pre-training task 수행
 → 이때, 80% : mask token, 10% : token을 random 단어로 변경, 10% : token을 원래 단어 그대로
 mask token은 fine-tuning시에는 사용되지 않음

3. **Next Sentence Prediction (NSP)** 방법

두 문장을 pre-training 시 같이 넣고, 두 문장이 이어지는 문장인지 아닌지 맞추는 것 → QA나 Natural Language Inference(NLI)와 같이 두 문장 사이 관계가 중요한 모델에 필요

Input = [CLS] the man went to [MASK] store [SEP] → **Sentence A**

he bought a gallon [MASK] milk [SEP] → **Sentence B**

Label = IsNext

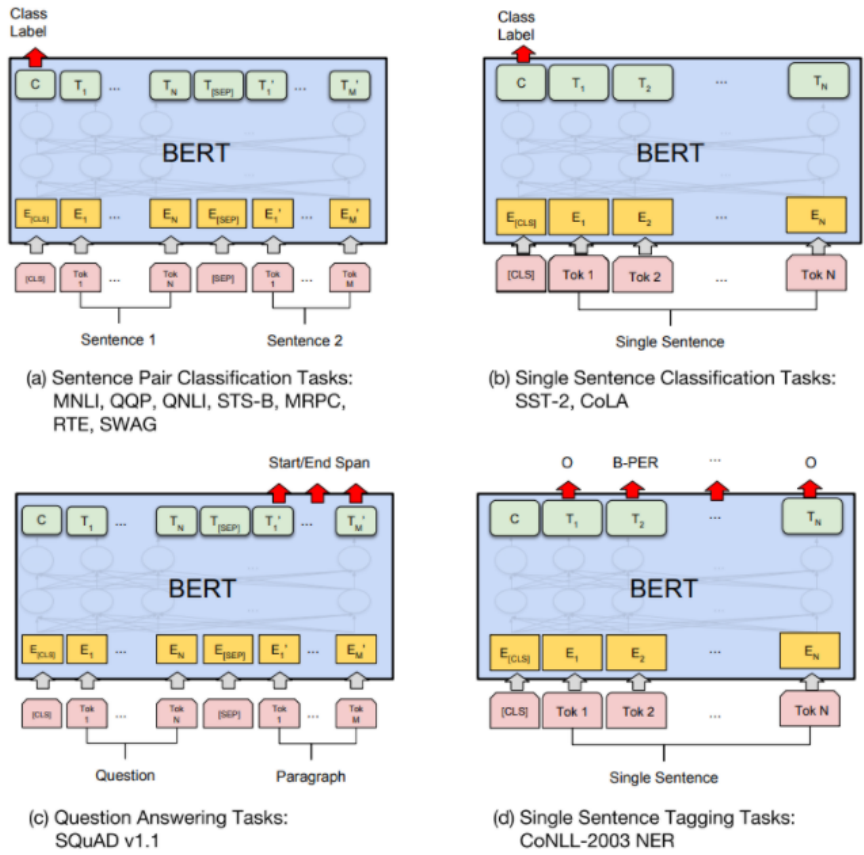
Input = [CLS] the man [MASK] to the store [SEP]

penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

50%는 두 문장이 실제로 연관이 있는 예제, 50%는 두 문장이 서로 연관이 없는 예제

- pre-training 절차
 1. NSP를 위해 문장을 뽑아서 임베딩
 2. Masking 작업
 3. 하이퍼파라미터 : 배치 사이즈 256, Adam 옵티마이저, dropout 0.1, gelu 활성화 함수 사용
- fine-tuning (transfer learning : 전이학습)



11개 task에 대한 fine-tuning

- a) 문제 쌍 분류 문제로 두 문장을 하나의 입력으로 넣고 두 문장의 관계를 구함
- b) 한 문장을 입력으로 넣고 문장의 종류 분류
- c) 문장이나 문단 내에서 원하는 정답 위치의 시작과 끝을 구함
- d) 입력 문장의 token명이나 품사를 구하는 문제

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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 데이터셋의 BERT 실험 결과

- Ablation 연구
 - effect of pre-training task

Tasks	Dev Set				
	MNLI-m (Acc)	QNLI (Acc)	MRPC (Acc)	SST-2 (Acc)	SQuAD (F1)
BERT _{BASE}	84.4	88.4	86.7	92.7	88.5
No NSP	83.9	84.9	86.5	92.6	87.9
LTR & No NSP	82.1	84.3	77.5	92.1	77.8
+ BiLSTM	82.1	84.1	75.7	91.6	84.9

BERT의 두가지 학습방법 (MLM, NSP)를 제거하며 결과 비교

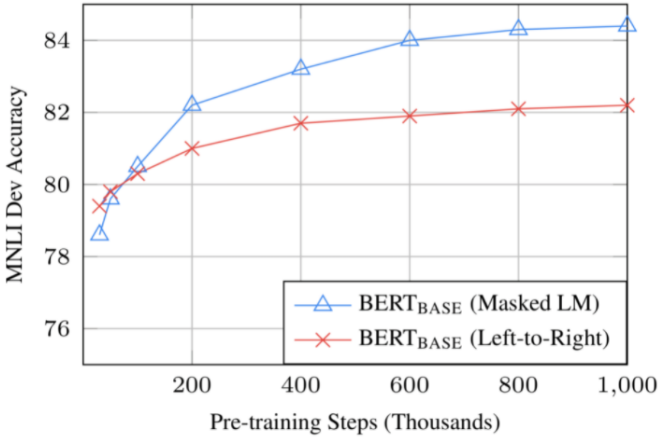
- effect of model size

Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

모델 크기에 따른 결과

모델이 커질수록 정확도 상승

- effect of number of training steps




- feature-based approach with BERT
 - ELMo와 같이 특정 NLP task를 수행할 수 있는 network를 부착하여 쓸 수 있음

Layers	Dev F1
Finetune All	96.4
First Layer (Embeddings)	91.0
Second-to-Last Hidden	95.6
Last Hidden	94.9
Sum Last Four Hidden	95.9
Concat Last Four Hidden	96.1
Sum All 12 Layers	95.5

finetune all과 성능 차이가 별로 없다

BERT 논문정리

최근에 NLP 연구분야에서 핫한 모델인 BERT 논문을 읽고 정리하는 포스트입니다. 구성은 논문을 쪽 읽어다가 며 정리한 포스트기 때문에 논문과 같은 순서로 정리하였습니다. Tmax Data AI 연구소에서 제가 진행한 세미나 동영상도 첨부합니다. BERT : Bidirectional Encoder Representations form Transformer 논문의 제목

 <https://mino-park7.github.io/nlp/2018/12/12/bert-%EB%85%BC%EB%AC%B8%EC%A0%95%EB%A6%AC/?fbclid=IwAR3S-8iLWEVG6FGUVxoYdwQyA-zG0GpOUzVEsFBd0ARFg4eFXqCyGLzn u7w#35-fine-tuning-procedure>

