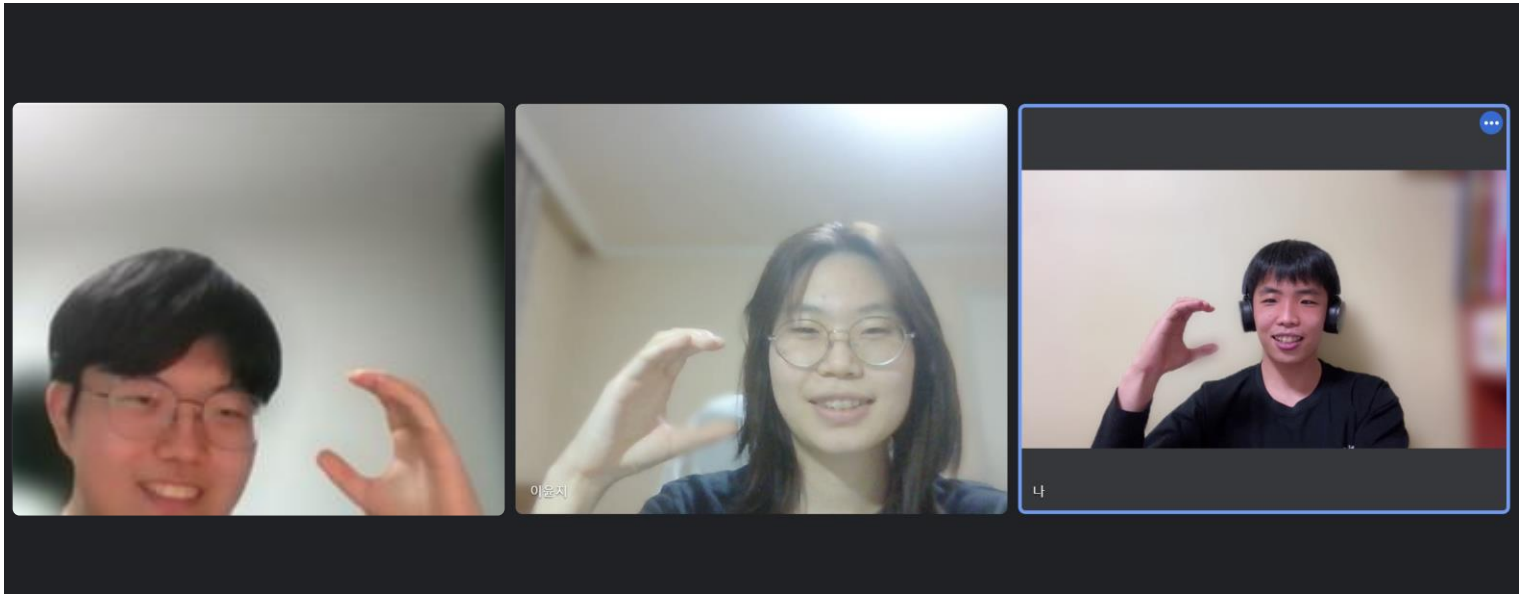
CUAI 5기 Kaggle / Dacon Study 3팀

2022.05.17

발표자 : 박도영

스터디원 만남 인증

스터디 사진_구글 밋



INDEX

INDEX

리뷰 대회 소개

소설 작가 분류 AI 경진대회

자연어 처리

Transformer

Bert

1등 코드 리뷰

Preprocessing 부분 리뷰

개선점 및 느낀점

리뷰 대회 소개

목적 : 문체 분석 알고리즘 개발

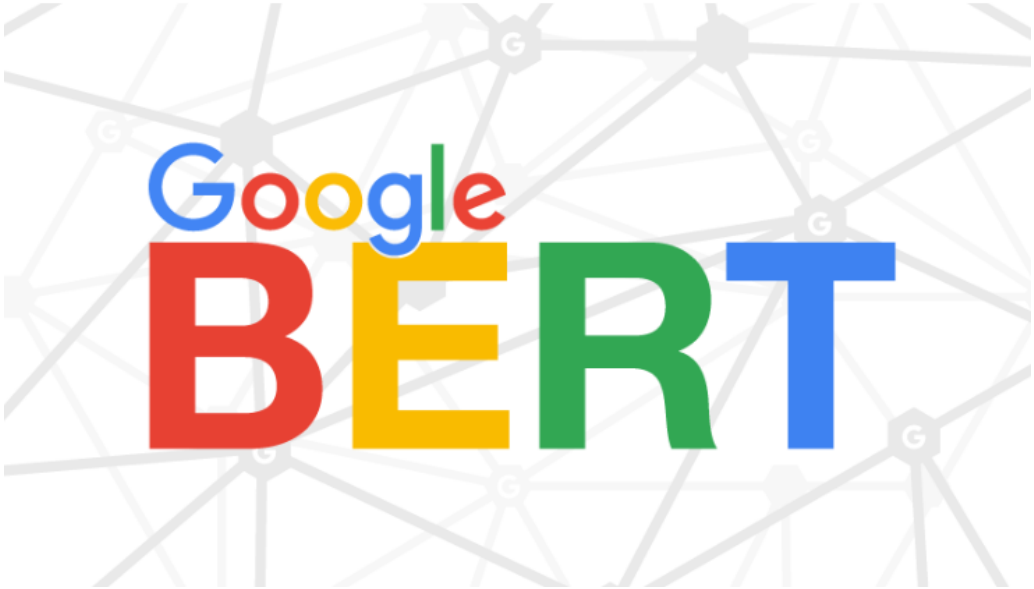
사전 학습 모델(pre trained model) 사용 불가
외부 데이터 사용 불가
Data Leakage 확인 시 실격

	index	text	author
0	0	He was almost choking. There was so much, so m...	3
1	1	"Your sister asked for it, I suppose?"	2
2	2	She was engaged one day as she walked, in per...	1

작가를 인코딩화
One Sample Multi Texts
약 50000개의 데이터

자연어 처리

Bidirectional Encoder Representations from Transformers

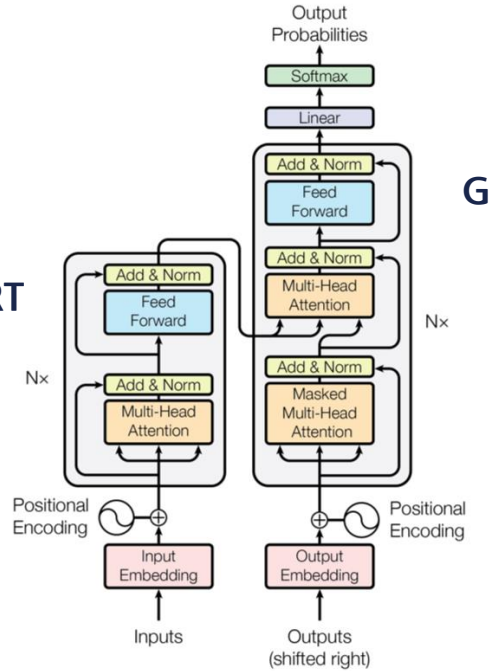


BERT

Bidirectional Encoder Representations from Transformers

한꺼번에 입력받아 양방향으로 문맥을 이해할 수 있다.

BERT



GPT

단어를 하나씩 읽어가면서 다음 단어를 예측하는 방법으로 학습

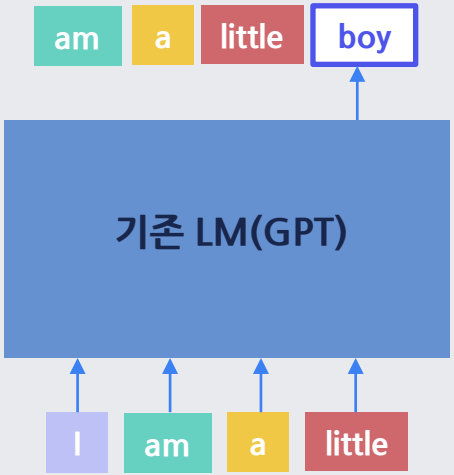
구글은 GPT-1이 문맥이해능력에 문제가 있을 수 있으며, 질의응답 task에 약하다고 판단.

BERT

기존 모델(ex. GPT) vs BERT

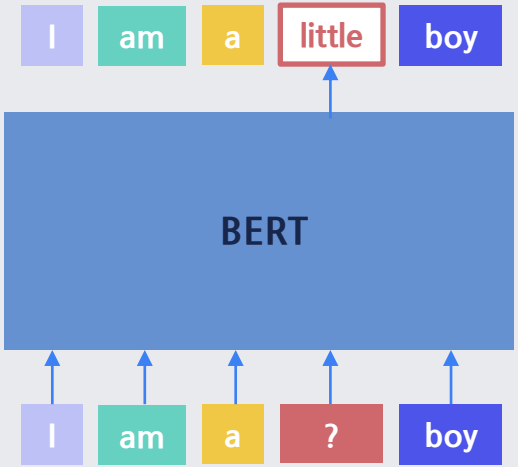
기존 LM(GPT)

현재까지 입력받은 문장을 통해 다음 단어 예측하도록 학습

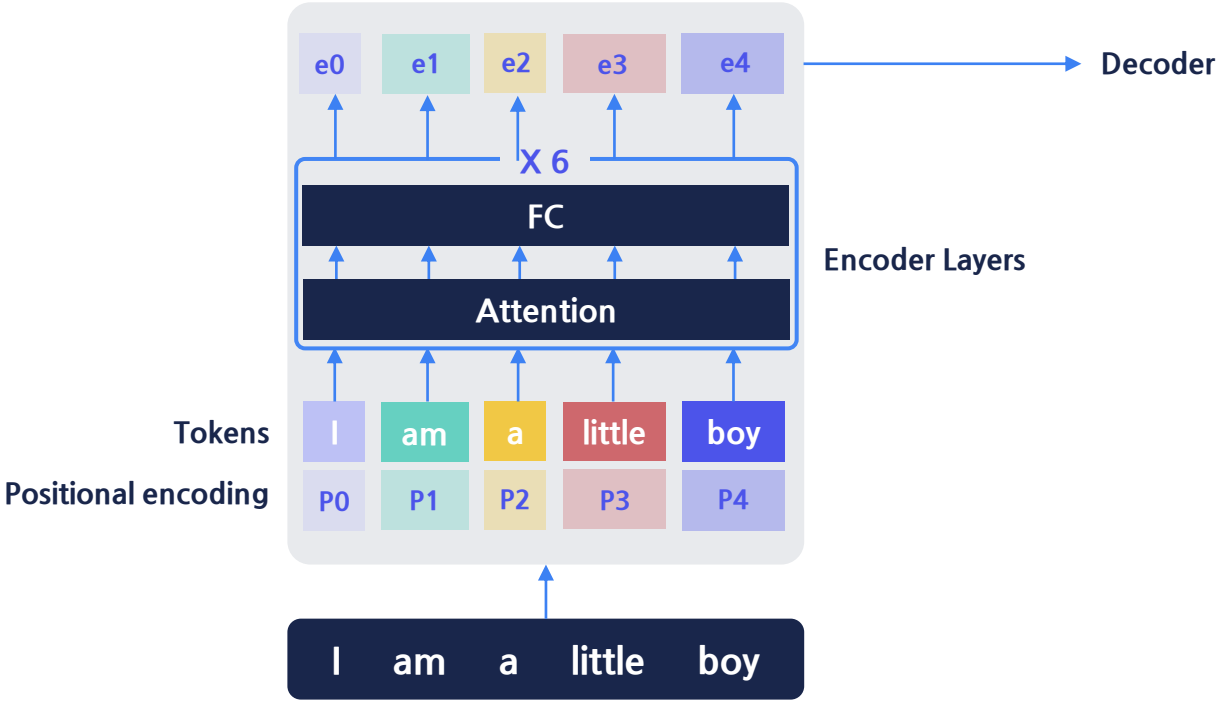


BERT

문장을 한꺼번에 입력받고 가려진 단어를 예측하도록 학습

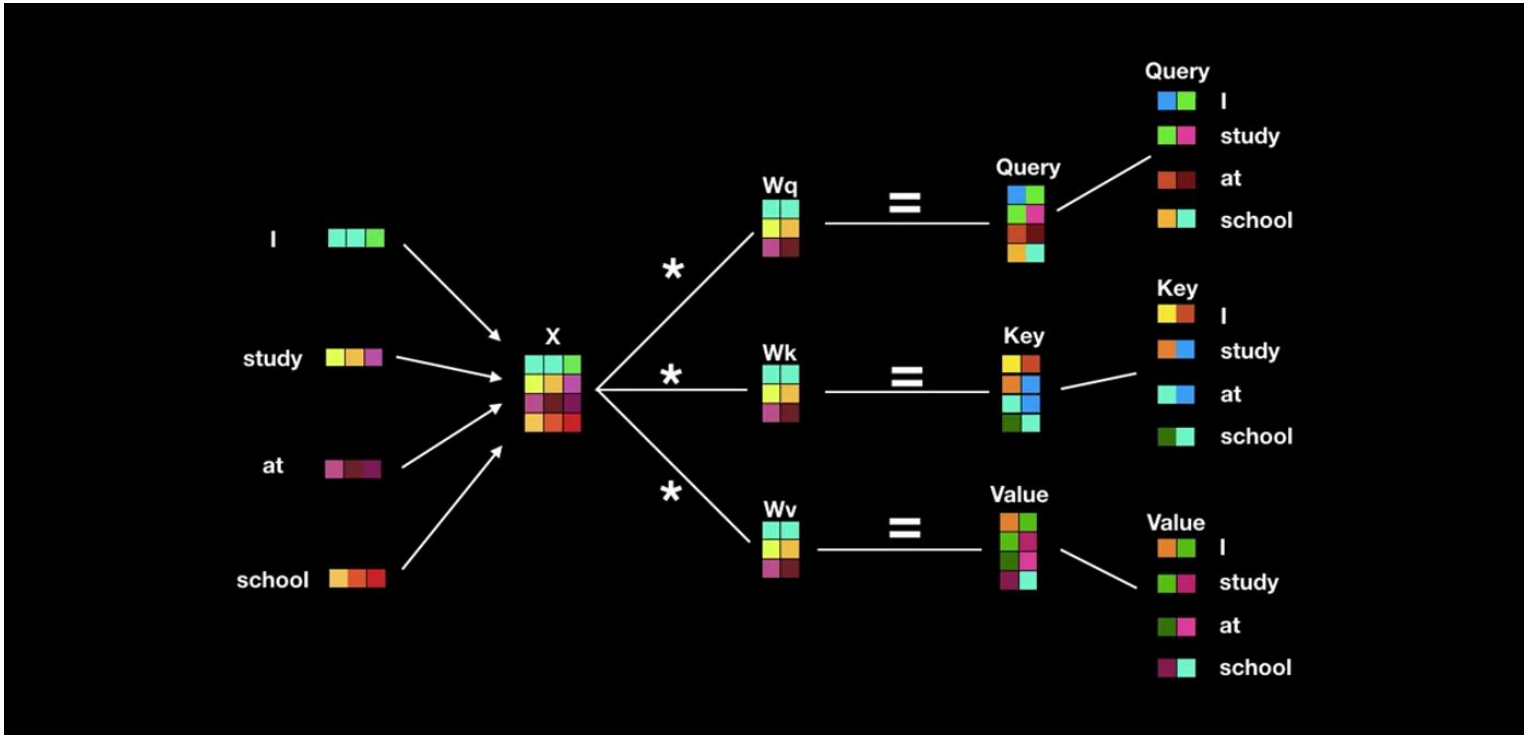


Transformer 작동방식(Encoder 부분)



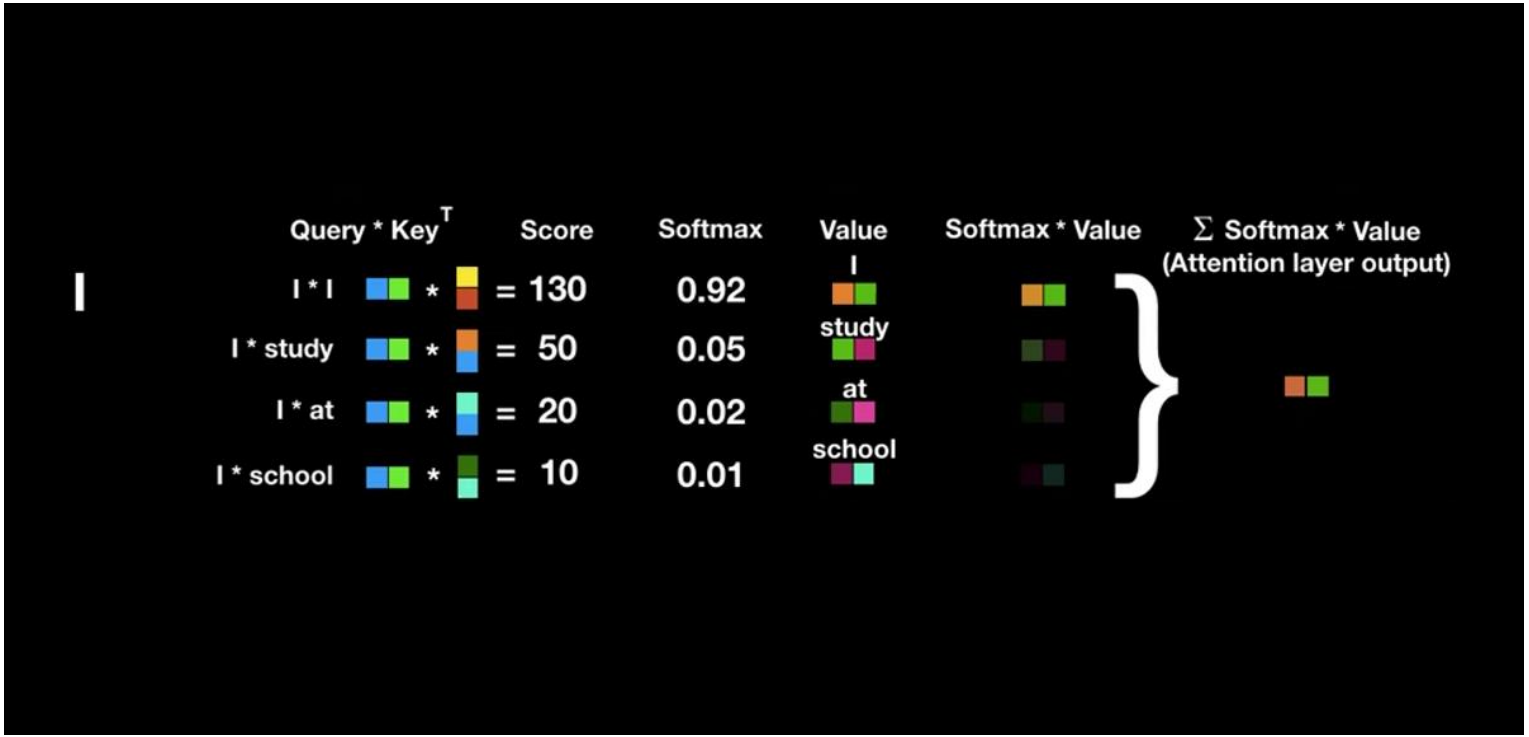
Transformer 작동방식(Encoder 부분)

Attention



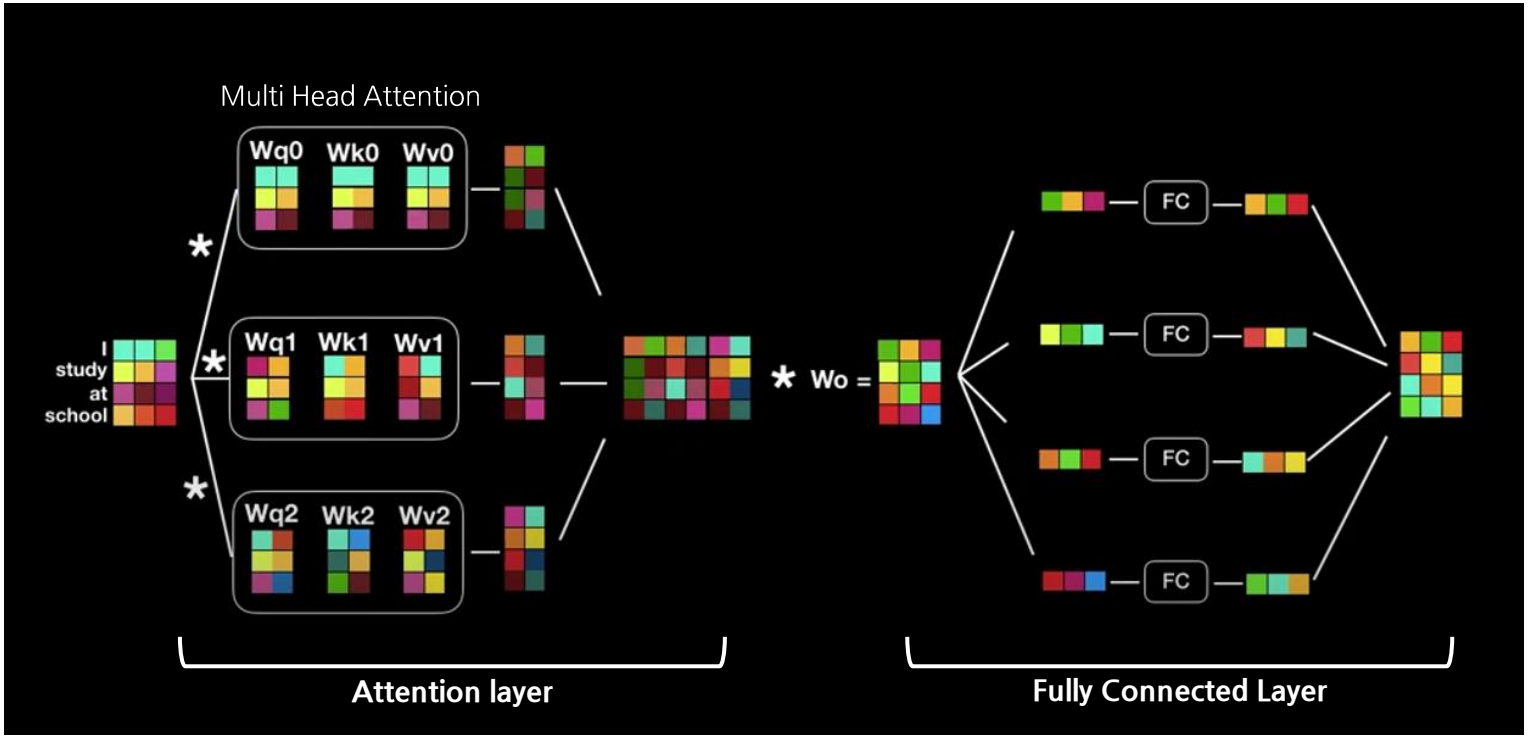
Transformer 작동방식(Encoder 부분)

Attention



Transformer 작동방식(Encoder 부분)

Encoder 전반



BERT

1. [CLS], [SEP]

문장 전체가 하나의 벡터로 표현된 스페셜 토큰

여러 문장을 구분짓기 위한 스페셜 토큰

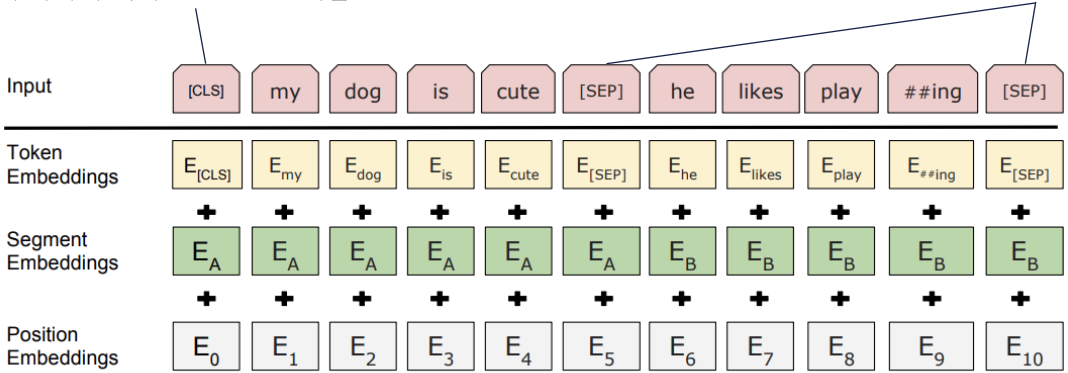


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT

2. Segment Embeddings

딥러닝 모델에게 여러 문장이 있다는 것을 알려주는 임베딩으로 각각의 문장에 다른 숫자를 부여하여 구별지음

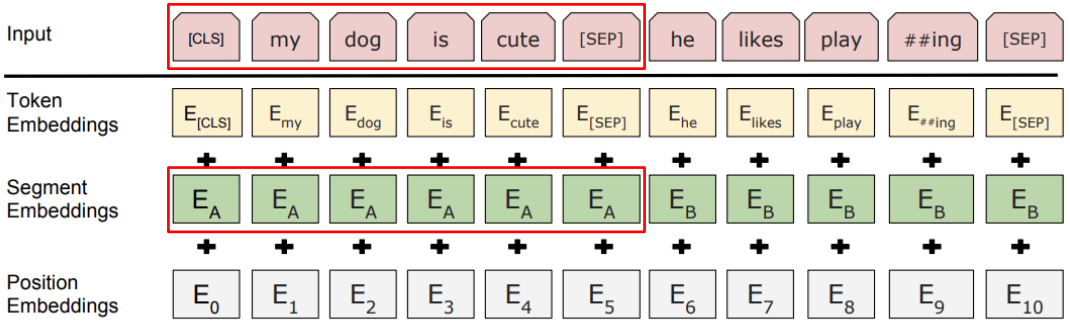
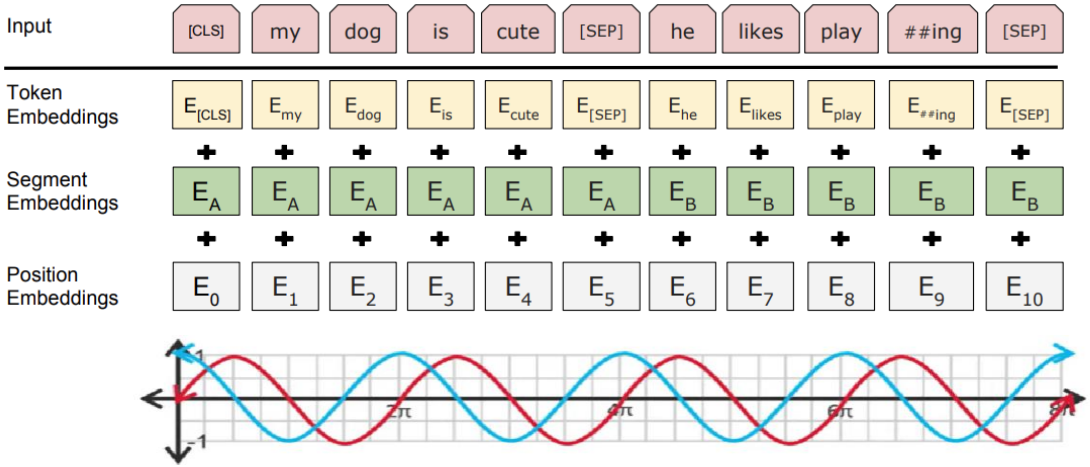


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

BERT

3. Position Embeddings

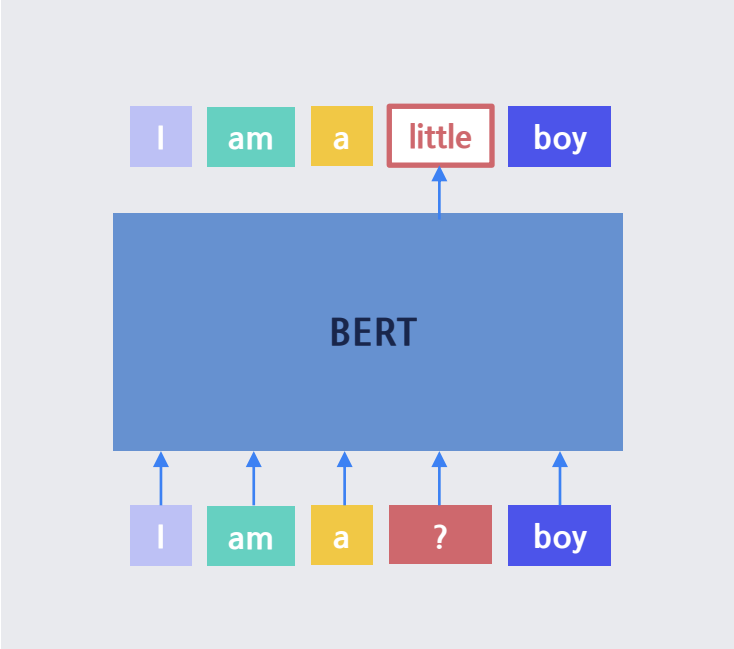
사인, 코사인 그래프는 규칙적으로 증가 감소하기때문에 딥러닝이 이 규칙을 활용해 토큰의 상대적 위치를 파악하기 쉬움
입력값으로 무한히 많은 토큰을 받아도 -1 ~ 1 사이의 숫자들로 상대적인 위치를 표현 가능함



BERT

Pre-training

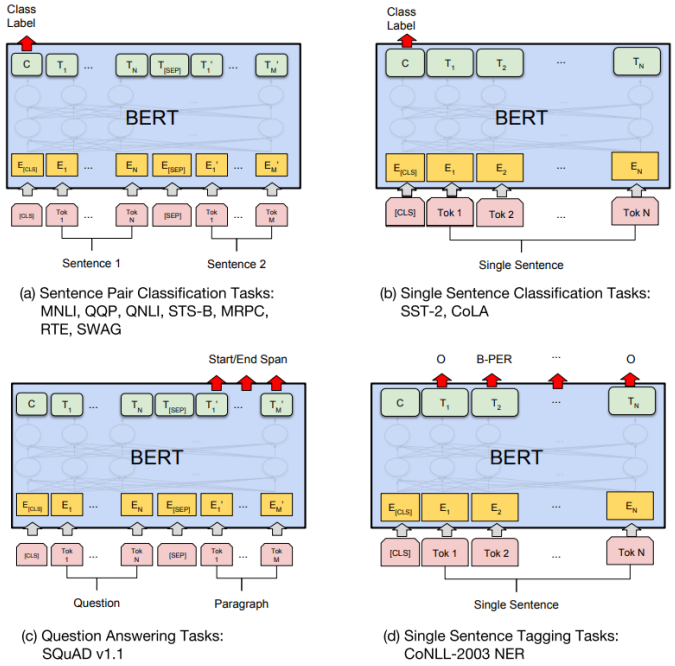
입력 문장에서 임의로 토큰을 버리고(Mask), 그 토큰을 맞추는 방식으로 학습을 진행



BERT

Fine Tuning

Pre Training한 모델을 전이학습시켜 NLP task를 수행하는 과정



1등 코드 리뷰(preprocessing)

```
def bert_encode(texts, tokenizer, max_len=max_len):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks), np.array(all_segments)
```

형태소 단위로 문장 쪼개기

토큰 맨 앞, 뒤에 [CLS], [SEP] 토큰 넣기

Padding : 문장을 벡터화 했을때 서로 다른 길이로 벡터화 되어 벡터 계산이 불가능한 경우를 방지하기 위해 짧은 문장의 남은 부분을 padding token으로 채우는 과정

형태소로 분리되어 있는 토큰을 숫자(id)로 변경

Padding token들이 self-attention 단계에 접근하는 것을 막기 위해 실제 문장의 인덱스에 1을 주고 padding token에는 0을 주는 attention mask.

한 샘플의 문장들을 하나로 받아 예측하여 segment_id를 모두 0이라는 한 숫자로 부여함

1등 코드 리뷰(pre processing)

개선점

1. 본 대회의 한 샘플은 여러 문장으로 이루어져 있음.

   코드 작성자는 Kaggle에서 구한 pre-training BERT 모델을 그대로 사용하여 [SEP]으로 문장들을 구분지어 임베딩하지 않고 Segment embedding 과정에서 한 샘플에 0을 부여하여 한 문장으로 취급하여 mask 예측을 학습시킴.

```
input_sequence = ["[CLS]"] + text + ["[SEP]"]
segment_ids = [0] * max_len
```

   --> [SEP]과 segment embedding을 활용하여 문장을 구분지어 모델을 학습시키는 분류 모델을 생성.

2. 본 대회는 사전학습 모델이 불가능한 대회로 BERT의 구조를 차용하여 학습시키는 것까지만 허용됨.

   코드 작성자는 BERT를 가져오는 과정에서 trainable=True로 두어 사전학습을 허용하였음.

```
bert_layer = hub.KerasLayer(module_url, trainable=True)
```

   이 실수로 1000만원과 애플워치를 잃어버림.

1등 코드 리뷰(pre processing)

느낀점

참조

- https://www.youtube.com/watch?v=mxGCEWOxfe8
- https://www.youtube.com/watch?v=30SvdoA6ApE
- https://ebbnflow.tistory.com/151
- https://dacon.io/competitions/official/235670/codeshare/1801?page=1&dtype=recent

감사합니다