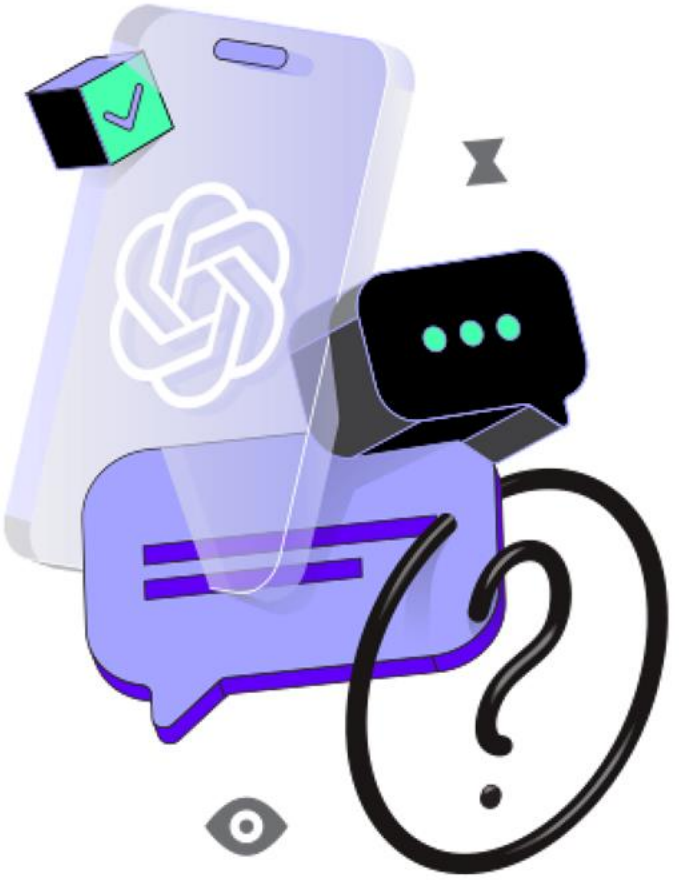
Paper review

GPT1, BERT, GPT2



카피바라팀 | 배누리, 김호정, 전사영, 박현아

2024.12.03

목차

01 GPT1

02 BERT

03 GPT2

04 코드 구현

Part.01

GPT1

등장 배경

기존 NLP 모델

- 기존의 NLP 모델은 대부분 지도 학습 방식을 사용했지만, NLP task에서 labeled data를 충분히 확보하기 쉽지 않음.
- unlabeled data는 대규모로 존재하지만 효과적으로 활용할 수 있는 방법이 부재.
- 서로 다른 task에 대해서 새로운 모델을 학습시켜함.

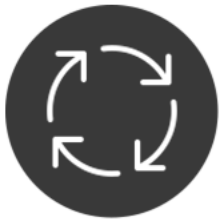
GPT 1

- 대규모 unlabeled data로 사전 학습(생성 모델) 후, 지도 학습으로 task에 맞게 미세 조정.
- 전이 학습을 통해 적은 양의 라벨 데이터로도 다양한 언어 task에 적용 가능.
- 최소한의 수정으로 여러 NLP 작업에서 높은 성능 달성.

주요 구성 및 목표

목표

모든 task에 보편적인 표현을 학습하여, 기존 모델은 최소한의 변화로 특정 task에 맞게 학습하는 것



훈련 절차

1. 대규모 unlabeled 데이터를 사용해 언어의 일반적인 구조와 문맥을 학습
2. Pre-trained 모델을 기반으로 labeled 데이터로 특정 task에 맞춰 미세 조정 (Fine-tuning)

네트워크 구조

Transformer의 Decoder 구조 사용

Framework

Unsupervised pre-training

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$Tokens\ U = \{u_1, \ldots, u_n\}$
k : context window의 사이즈
Θ : 파라미터 (w, bias 등)

- $u_{i-k}, \ldots, u_{i-1}$ 가 주어졌을 때 그 다음 단어가 u_i 일 확률이 높도록 Θ 값 학습
- 언어 모델이 문장을 얼마나 잘 예측하는지 평가

- 예) I love you라는 문장이 있으면 I, love 가 주어졌을 때 you를 예측하는 확률값

Framework

Unsupervised pre-training

$$
\begin{aligned}
h_0 &= UW_e + W_p \\
h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\
P(u) &= \texttt{softmax}(h_n W_e^T)
\end{aligned}
$$

Language model – Transformer Decoder

- 기존 decoder 부분에서 encoder layer가 없기 때문에 encoder-decoder attention을 제외
- 입력 토큰에 임베딩(W_e)을 진행 후 위치 임베딩(W_p)을 더해 초기 상태를 만들고, n개의 Transformer Block을 통과한 후 Softmax를 통해 각 단어의 확률 분포를 계산

Framework

Supervised fine-tuning

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

- 모델이 주어진 입력 $x^1, \ldots, x^m$ 에 대해 출력 y 의 확률 분포를 계산

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

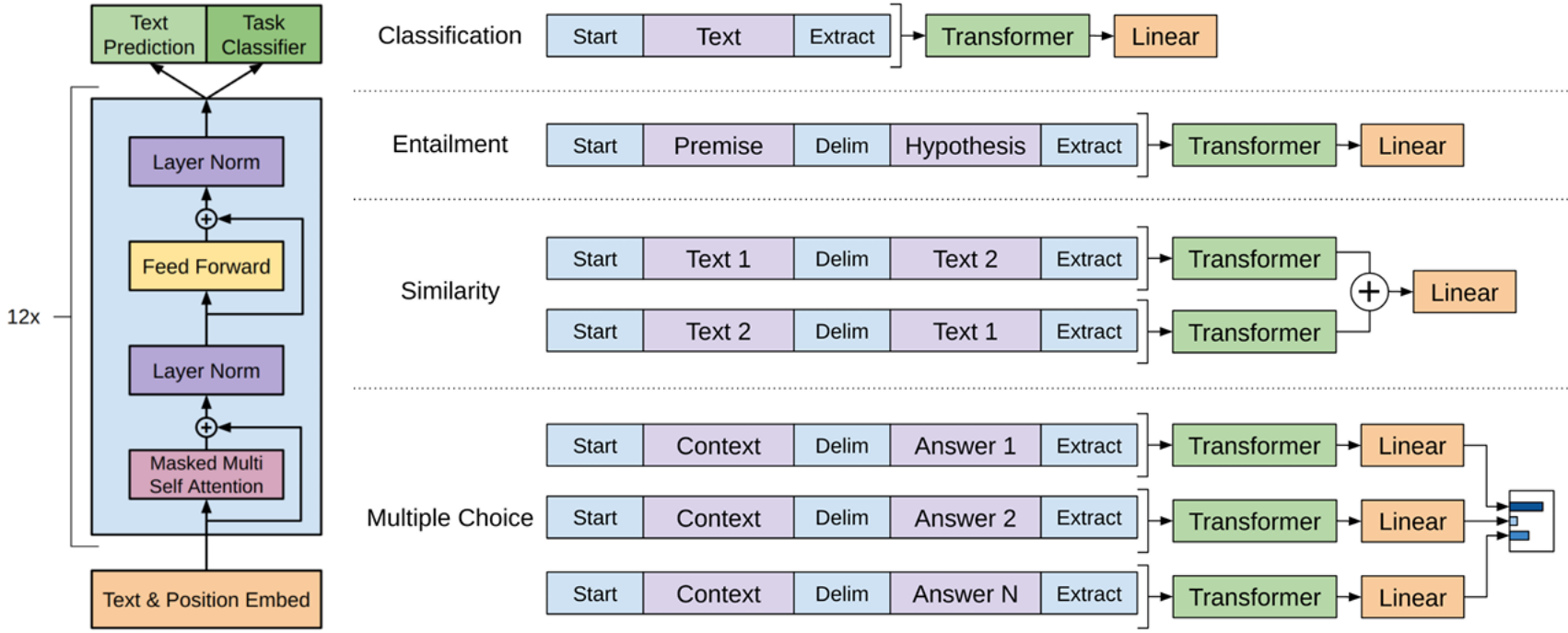
- 라벨이 있는 데이터셋 C에서 정답 y에 대한 확률 로그값을 합산한 손실 함수
- 모델은 손실 함수를 최소화하여 정답을 더 잘 예측하도록 학습

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

- Fine-Tuning 과정에서 Supervised 손실 $L_2(C)$ 에 Pre-training 손실 $L_1(C)$ 을 결합하여 모델의 일반화 성능을 향상

Framework

Task-specific input transformations



Part.02

BERT

등장 배경

unidirectional language model

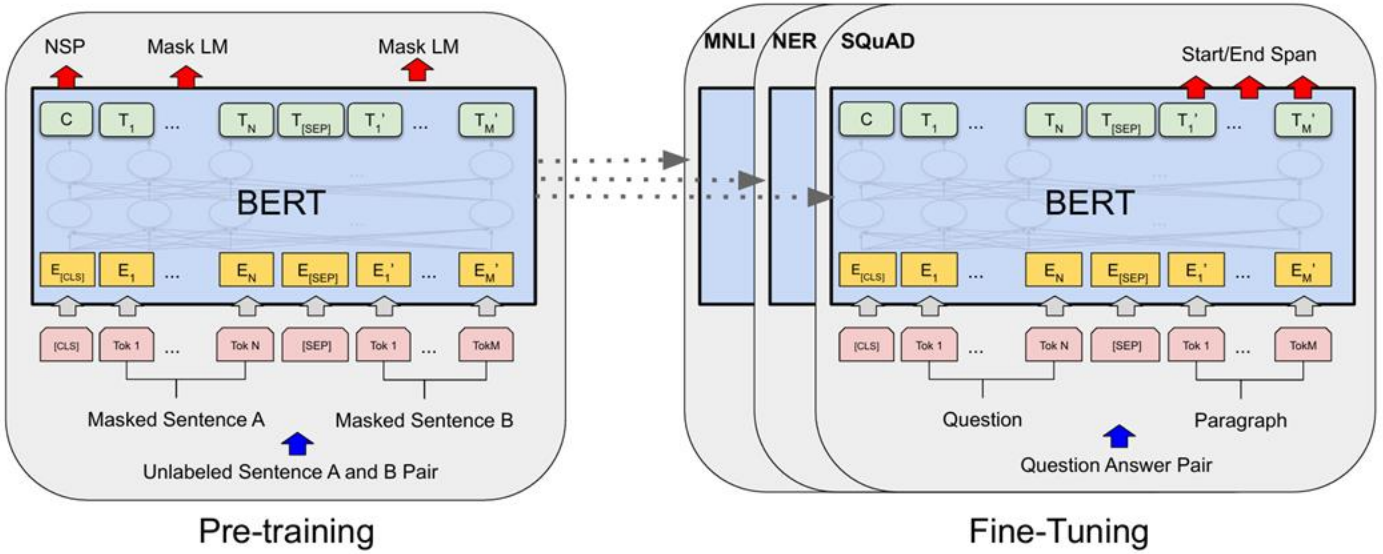
- 2018년 당시 단방향 언어 모델은 사전 학습된 표현 (pre-trained representation)의 성능에 한계가 있음
- GPT-1은 Transformer의 디코더 구조로 이전 단어만 참고하는 방식이어서, 문맥을 모두 이해해야 하는 Q&A 같은 작업에서 한계가 있음.

BERT

- 단방향 처리의 한계를 극복하기 위해 MLM (Masked Language Model) 방식을 도입
- 텍스트 쌍(Text Pair)을 학습하기 위해 NSP (Next Sentence Prediction) task를 추가로 사용

BERT

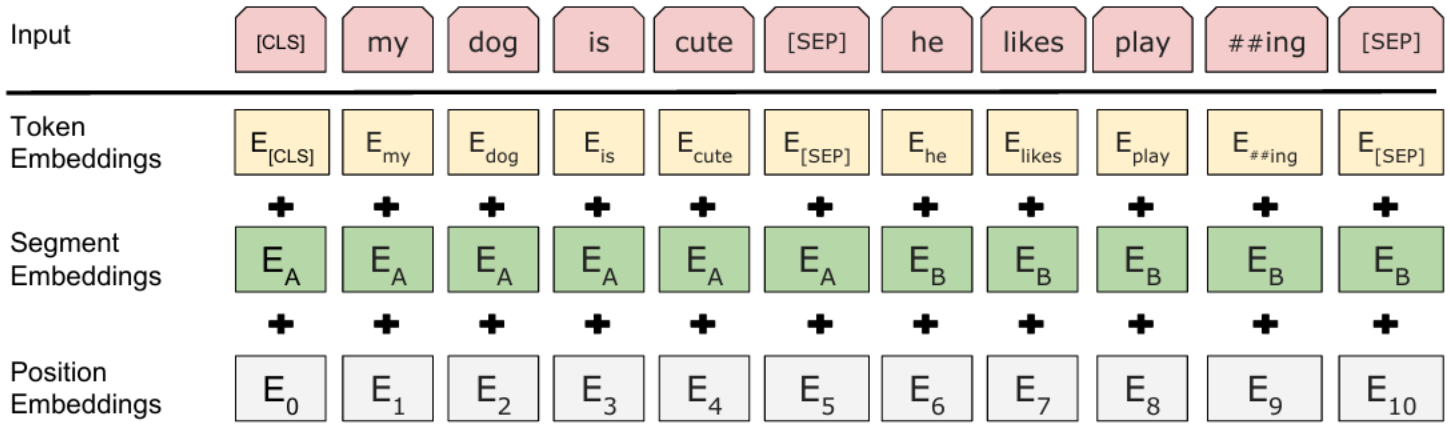
Model Architecture



- Transformer의 encoder 부분만 사용 (양방향 Transformer encoder를 여러 층 쌓은 것)
- BASE, LARGE 두 가지 모델
  - $BERT_{BASE}$: L=12, H=768, A=12, Total Parameters=110M
  - $BERT_{LARGE}$: L=24, H=1024, A=16, Total Parameters=340M
  - L= Transformer block layer의 수, H= hidden size, A= self-attention 헤드의 수

BERT

Input / Output Representations

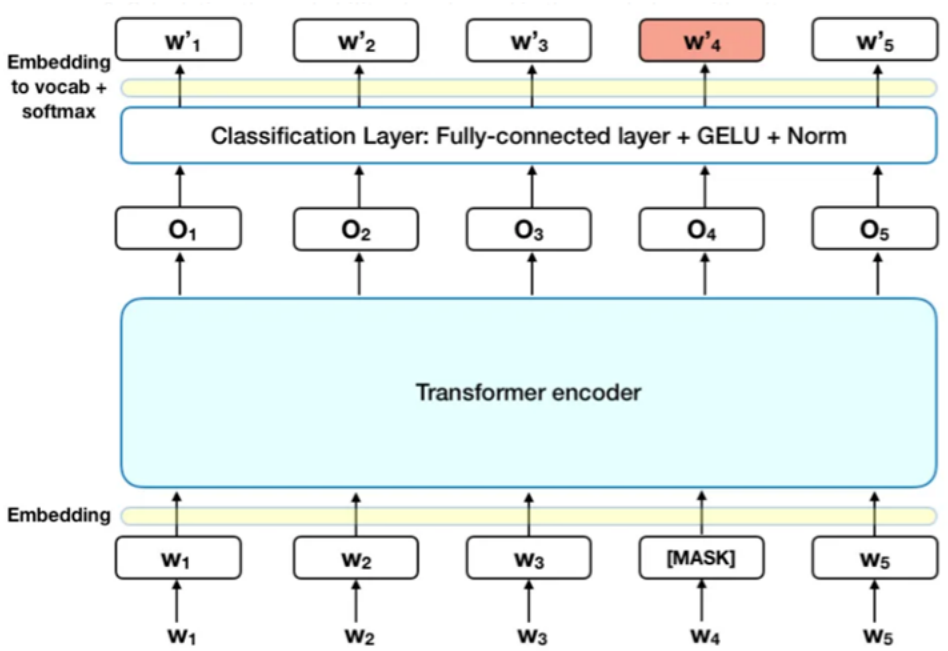


Input = Token Embedding + Segment Embedding + Position Embedding

- Token Embedding : 텍스트의 각 단어(토큰)를 벡터 형태로 변환
- Segment Embedding : 입력된 문장이 문장 A와 문장 B로 나뉘었을 때, 각 토큰이 어느 문장에 속해 있는지를 나타내는 임베딩
- Position Embedding : 입력 텍스트에서 각 토큰의 위치(순서)를 나타내는 임베딩

BERT

Pre-training BERT : Masked LM (MLM)



- Token에 랜덤한 확률값에 따른 Mask를 씌워 문맥을 양방향으로 학습
- 문장 전체를 predict 하는 것이 아닌, [Mask] token만을 predict
- pre-training에만 사용되고, fine-tuning 과정에서는 사용되지 않음

  → mismatch가 발생할 수 있음
- Mismatch 해결방안 : 15%의 [MASK] token에서 추가적인 처리를 더 해줌.
  - 80%의 경우 : token을 [MASK] token으로 바꿈.

    ex) my dog is hairy -> my dog is [MASK]
  - 10%의 경우 : token을 random word로 바꿈

    ex) my dog is hairy -> my dog is apple
  - 10%의 경우 : token을 원래 단어 그대로 놔둔다.

    ex) my dog is hairy -> my dog is hairy

BERT

Pre-training BERT : Next Sentence Prediction (NSP)

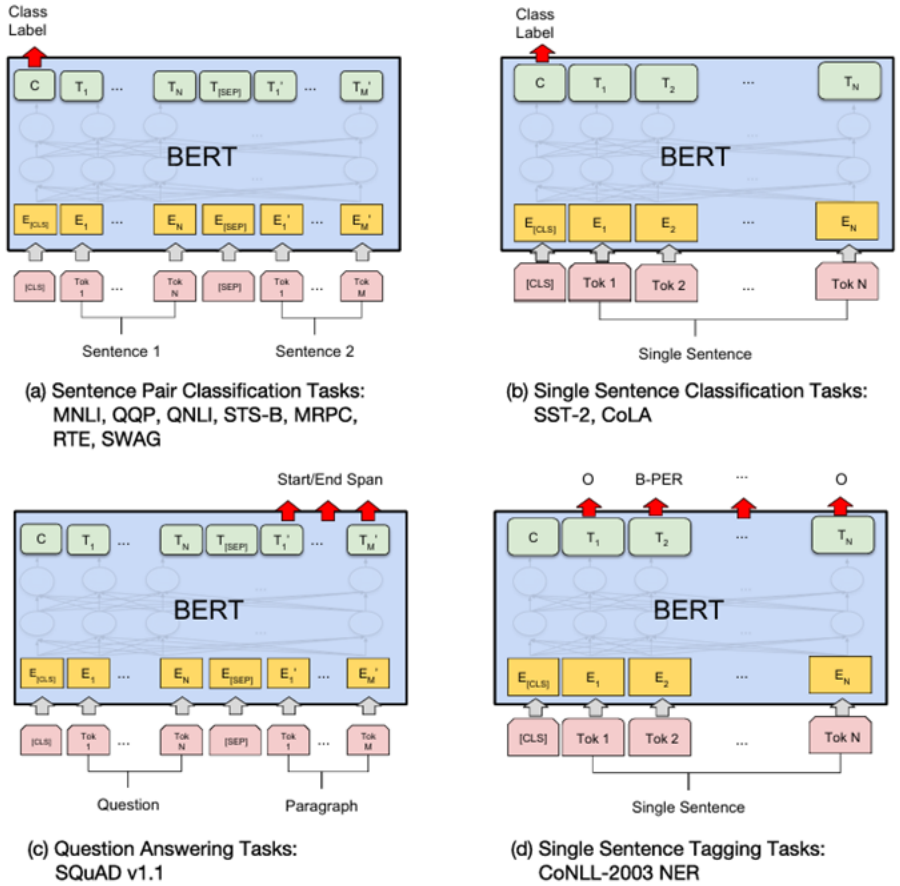
- 문장 A와 문장 B가 실제로 연속된 문장인지(관계가 있는지)를 학습하는 과정
- pre-training 데이터로 문장 A와 B를 선택할 때, 50퍼센트는 실제 A의 다음 문장인 B를(IsNext), 나머지 50퍼센트는 랜덤 문장 B를(NotNext) 고름

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For

- IsNext
  - Sentence A: We argue that current techniques restrict the power of the pre-trained representations
  - Sentence B: especially for the fine-tuning approaches.
- NotNext
  - Sentence A: We argue that current techniques restrict the power of the pre-trained representations
  - Sentence B: The major limitation is that standard language models are unidirectional

BERT

Fine-tuning BERT



- 각 task 마다 입력(Input)과 출력(Output)의 형식은 약간 다를 수 있지만, 모델 구조 자체를 변경할 필요가 없음
- Pre-training을 통해 대부분의 파라미터가 이미 학습된 상태에서 Fine-tuning은 소량의 labeled 데이터와 적은 학습 시간만으로 진행

Sentence Pair Classification Tasks (a)

- 두 문장 간의 관계를 예측하며 [SEP] 토큰으로 문장을 구분.

Single Sentence Classification Tasks (b)

- 하나의 문장을 입력으로 받아 분류 결과를 [CLS] 토큰으로 출력.

Question Answering Tasks (c)

- 질문과 문단을 입력으로 받아 답의 시작과 끝 위치를 예측.

Single Sentence Tagging Tasks (d)

- 문장의 각 단어에 태그(예: 개체명)를 부여하는 작업.

Part.03

GPT 2

등장 배경

pretrained language model

- BERT와 GPT-1 같은 모델은 Transformer의 self-attention 구조를 사용해서, task 에 맞는 새로운 구조를 따로 만들 필요가 없음.
- 분류 같은 작업에서는 단순히 하나의 출력 레이어만 추가하면 다양한 task에 쉽게 적용할 수 있음.
- 하지만, 이 과정에서 Fine-tuning(세부 조정)을 위해 지도학습이 필요.

GPT2

- 비지도학습 기반 언어 모델
- 모델 구조나 파라미터를 바꾸지 않고도 새로운 task를 바로 수행할 수 있음 (zero-shot setting)

Approach

language modeling

$$p(x) = \prod_{i=1}^{n} p(s_n | s_1, \ldots, s_{n-1})$$

- 단일 과정 학습은 $p(output|input)$ 을 추정
- 범용 시스템은 여러 다른 과제들을 수행할 수 있어야 하기 때문에 같은 입력이라도 입력뿐 아니라 과제의 종류라는 조건이 들어가야 함

  → $p(output|input, task)$로 표현

- translation train example : (translate to french, english text, french text).
- reading comprehension training example : (answer the question, document, question, answer).

Approach

Input Representation

- Byte Pair Encoding(BPE) 방식 사용
- BPE : subword 기반의 인코딩 방법으로 문자 단위로 단어를 분해하여 Vocabulary를 생성하고, 반복을 통해 빈도수가 높은 문자 쌍을 지속적으로 Vocabulary에 추가하는 방법

$$Vocabulary_{word} = \{apple, available, capable\}$$
$$Vocabulary_{character} = \{a, p, l, e, v, i, b, c, p\}$$

{apple, available, capable} 로 구성된 Word Vocabulary로부터 Character vocabulary {a,p,l,e,v,i,b,c,p}를 얻음

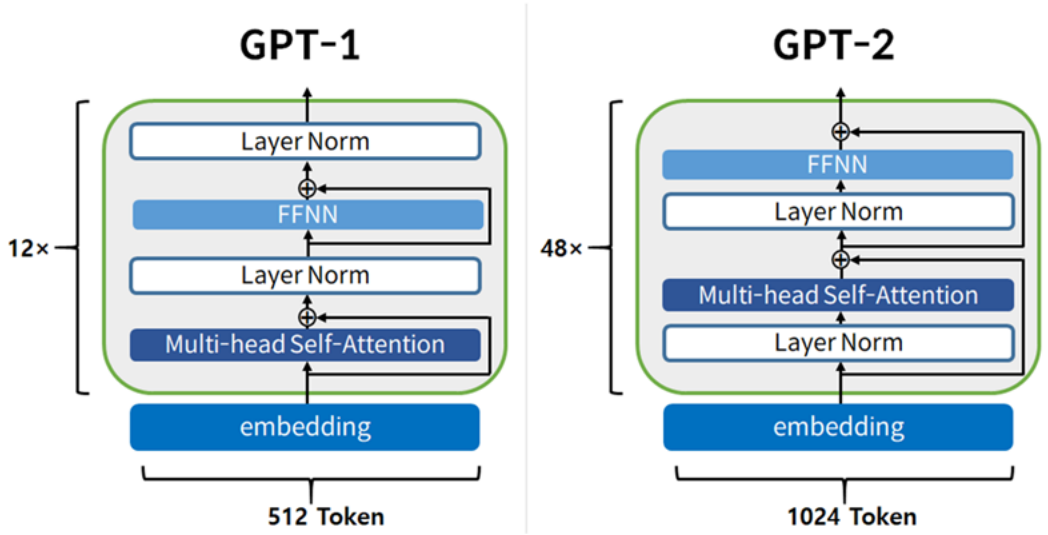
$$Vocabulary_{BPE} = \{a, p, l, e, v, i, b, c, p, le, ble, able\}$$

그 후, 매 회 반복을 통해 le, ble, able 과 같이 함께 자주 등장하는 문자 쌍을 Character vocabulary 에 추가

예상되는 결과는 반복에 따라 {a,p,l,e,v,i,b,c,p,le,ble,able}과 같은 vocabulary가 완성

Approach

Model



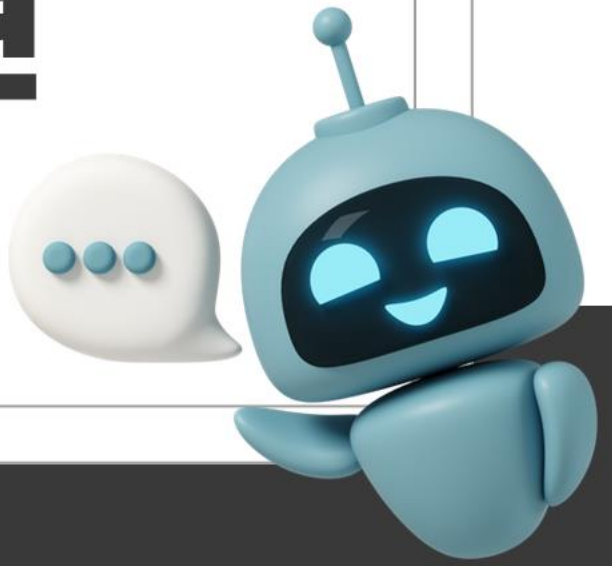
Transformer가 기본 구조이며 GPT-1의 구조를 대부분 따름.

- Layer normalization이 각 Sub-block (Self-Attention 및 Feed-Forward)의 입력 단계로 이동.
- Residual Layer의 가중치 초기화 방법이 Residual Layer의 깊이(N)에 따라, 각 Residual Layer의 가중치에 $1/\sqrt{N}$을 곱하는 방식으로 변경됨.

- Vocab size: 40,000 → 50,257
- Context size: 512 → 1024 tokens (maxlen)
- Batch size: 64 → 512

Part.04

코드 구현



질의응답

```python
from transformers import pipeline

# Pre-trained 모델 불러오기 (질문-응답을 위한 모델)
qa_pipeline = pipeline("question-answering", model="distilbert-base-cased-distilled-squad")

# 본문과 질문 설정
context = """
Tom goes everywhere with Catherine Green, a 54-year-old secretary. He moves around her office at work and goes shopping with her.
"Most people don't seem to mind Tom," says Catherine, who thinks he is wonderful. "He's my fourth child," she says.
She may think of him and treat him that way as her son. He moves around buying his food, paying his health bills and his taxes,
but in fact Tom is a dog.

Catherine and Tom live in Sweden, a country where everyone is expected to lead an orderly life according to rules laid down by the government,
which also provides a high level of care for its people. This level of care costs money.
"""

questions = [
    "How old is Catherine?",
    "Where does she live?",
    "Who is Tom?",
    "What country does Catherine live in?"
]

# 각 질문에 대해 답변 생성
for question in questions:
    result = qa_pipeline(question=question, context=context)
    print(f"Q: {question}")
    print(f"A: {result['answer']}\n")
```

질의응답 - 결과

```
Q: How old is Catherine?
A: 54

Q: Where does she live?
A: Sweden

Q: Who is Tom?
A: a dog

Q: What country does Catherine live in?
A: Sweden
```

텍스트 생성

```python
from transformers import GPT2LMHeadModel, GPT2Tokenizer

# GPT-2 모델과 토크나이저 불러오기
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# 모델을 평가(evaluation) 모드로 설정
model.eval()

def generate_text(context, max_length=150, temperature=0.7, top_k=50):
    """
    GPT-2를 사용해 텍스트를 생성하는 함수.

    Args:
        context (str): 입력 텍스트 (프롬프트).
        max_length (int): 생성될 텍스트의 최대 길이.
        temperature (float): 생성의 랜덤성을 조절.
        top_k (int): 상위 k개의 단어만 샘플링에 사용.

    Returns:
        str: 생성된 텍스트.
    """
    # 입력 텍스트를 토큰화
    input_ids = tokenizer.encode(context, return_tensors="pt")

    # 텍스트 생성
    output = model.generate(
        input_ids,
        max_length=max_length,
        temperature=temperature,
        top_k=top_k,
        do_sample=True,  # 샘플링 활성화
        pad_token_id=tokenizer.eos_token_id  # 문장이 끝났음을 알려주는 토큰
    )

    # 생성된 토큰을 텍스트로 디코딩
    generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
    return generated_text

# 예시 사용
context = """Today, I wake up early as usual and start the same day.
But it snowed so much that the roads were frozen solid.
        """
generated = generate_text(context, max_length=200)
print("Generated Text:\n", generated)
```

텍스트 생성 - 결과

```
Generated Text:
 Today, I wake up early as usual and start the same day.
But it snowed so much that the roads were frozen solid.

This was just like the day I had been waiting for...
And yet, I kept getting colder...
And the snow was so thick that I couldn't even see my face...
I was exhausted from all the sweat and tears...
I had to look past the snow and towards the other side of the mountain.
Then, I saw the way that this mountain had turned back.
I was going to have to stop.
I was going to get up and get in the car.
The speed was going to be slow, but I could
```

번역

```python
from transformers import MarianMTModel, MarianTokenizer

# 번역 모델과 토크나이저 설정 (English → French)
model_name = "Helsinki-NLP/opus-mt-en-fr"  # English to French
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)

def translate_text(text, tokenizer, model):
    """텍스트를 번역하는 함수"""
    # 입력 텍스트를 토큰화
    tokenized_text = tokenizer.prepare_seq2seq_batch([text], return_tensors="pt", padding=True)
    # 모델로 번역 수행
    translated = model.generate(**tokenized_text)
    # 번역 결과 디코딩
    translated_text = tokenizer.batch_decode(translated, skip_special_tokens=True)[0]
    return translated_text

# 영어 문장 입력
english_text = "This re-release, titled The Next Day Extra, was presented in the form of three disks: the original album, \
    unpublished studio sessions and remixes, plus a DVD containing the four clips that have already been unveiled."

# 영어 → 프랑스어 번역
french_translation = translate_text(english_text, tokenizer, model)
print("French Translation:", french_translation)
```

번역

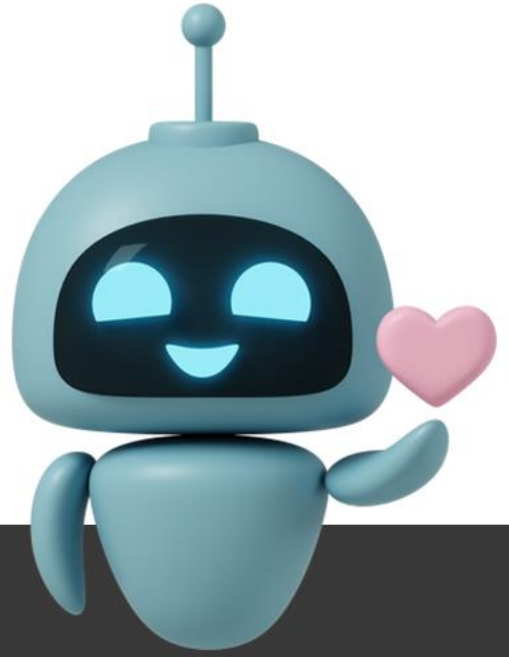
```
# 프랑스어 → 영어 모델 로드 (French → English)
model_name_fr_en = "Helsinki-NLP/opus-mt-fr-en"  # French to English
tokenizer_fr_en = MarianTokenizer.from_pretrained(model_name_fr_en)
model_fr_en = MarianMTModel.from_pretrained(model_name_fr_en)

# 프랑스어 입력
french_text = "Un homme expliquait que le fonctionnement de la hernia fonctionnelle qu'il avait reconnu avant de faire, \
    le fonctionnement de la hernia fonctionnelle que j'ai réussi, j'ai réussi."

# 프랑스어 → 영어 번역
english_translation = translate_text(french_text, tokenizer_fr_en, model_fr_en)
print("English Translation:", english_translation)
```

번역 - 결과

```
French Translation: Cette réédition, intitulée The Next Day Extra, a été prése
ntée sous la forme de trois disques : l'album original, des sessions de studio
 inédites et des remixes, ainsi qu'un DVD contenant les quatre clips qui ont d
éjà été dévoilés.
English Translation: One man explained that the functioning of the functional
hernia that he had recognized before doing, the functioning of the functional
hernia that I succeeded, I succeeded.
```



감사합니다
Q&A