

GPT-2

Language Models are Unsupervised Multitask Learners

Alec Radford * 1 Jeffrey Wu * 1 Rewon Child 1 David Luan 1

[Submitted on 14 FEB 2019]

논문 구현

CloseAI팀 이상헌 이정훈 박준혁 김유철

목차

- GPT-1
- BERT
- GPT-2
- 구현결과
- Reference
- Q&A

GPT-1

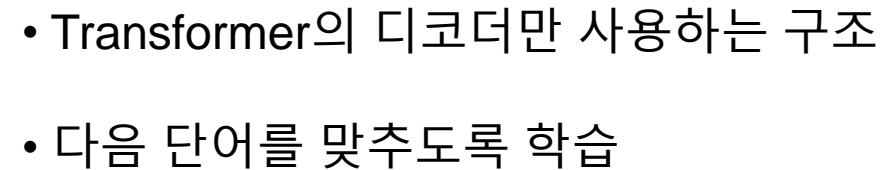
GPT (Generative Pre-training Transformer)

기존 방법의 문제점

- Labeled Data의 양이 많지 않다 → 데이터 부족
- Unlabeled Data는 많지만 활용할 수 있는 방법 X

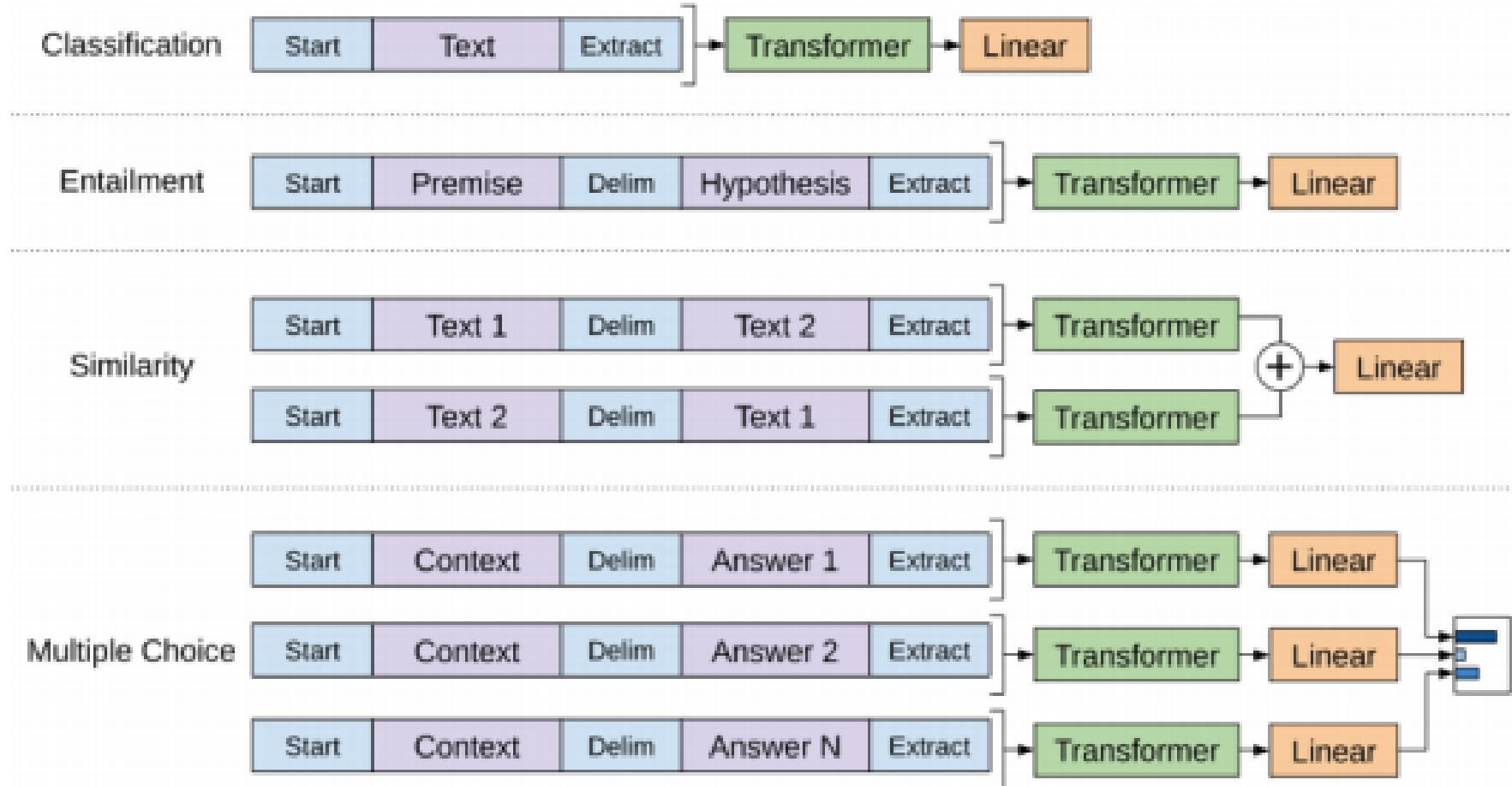
GPT-1가 제안한 방법

- Unlabeled Data를 사용한 Pre-Training
- Task에 맞게 Fine-Tuning



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

GPT (Generative Pre-training Transformer)



GPT (Generative Pre-training Transformer)

장점

- 전이학습의 효율성을 높였다
- Task 별 최소한의 엔지니어링
- 강력한 성능(SOTA)

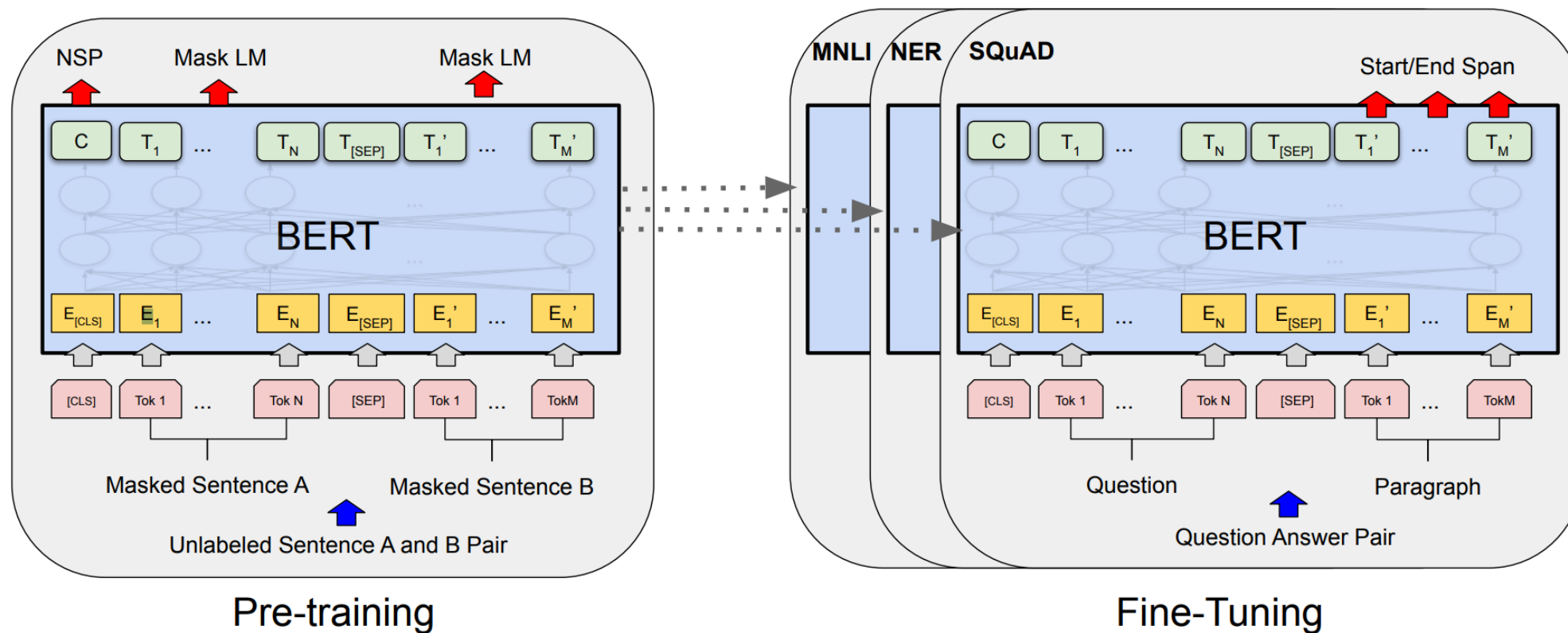
단점

- 많은 계산량에서 야기되는 비싼 계산 비용
- Task 별 Fine-Tuning의 어려움(상대적)

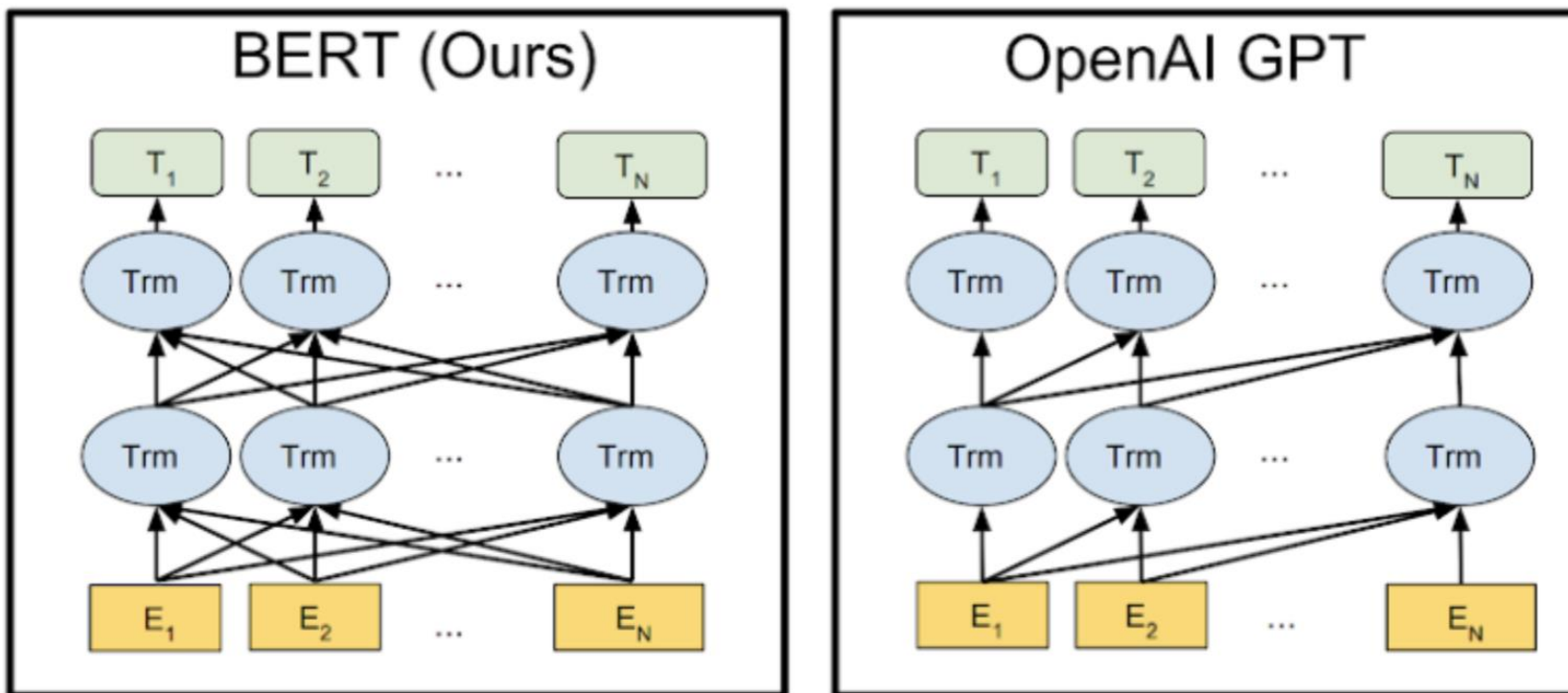
BERT

BERT

9



BERT (Bidirectional Encoder Representations from Transformers)

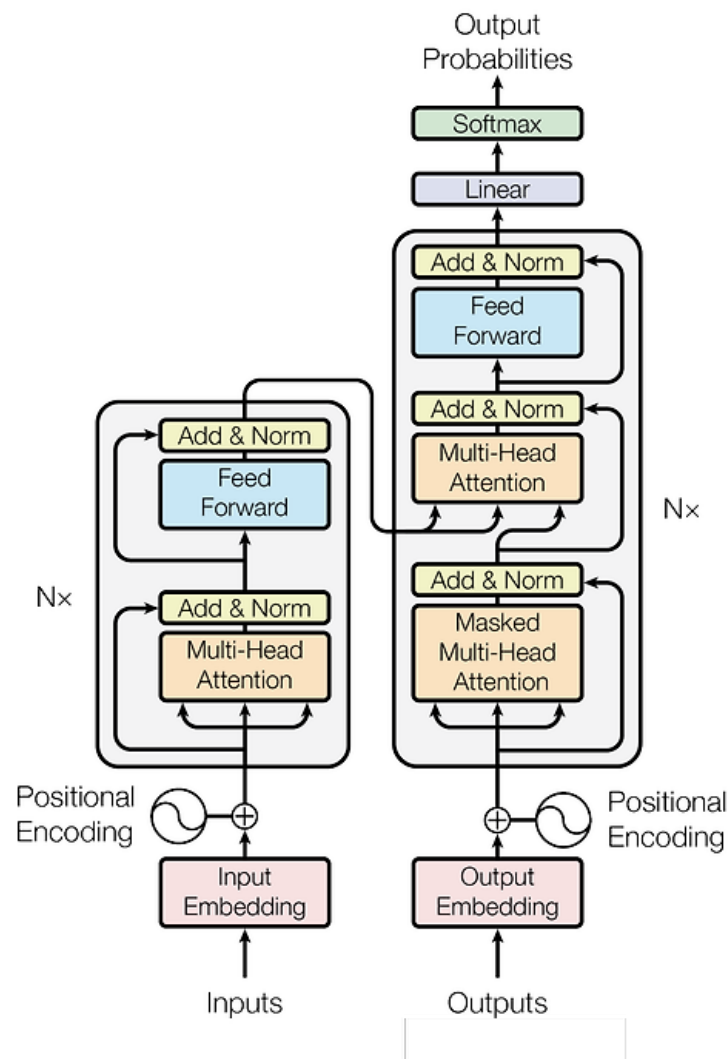


왼쪽에서 오른쪽으로 이동하도록 구성된 구조는 토큰이 이전에 나타난 토큰들에 대해서만 attend할 수 있다.

문맥을 양쪽에서 모두 이해해야 하는 question-answering task 에서 있어서 치명적

BERT

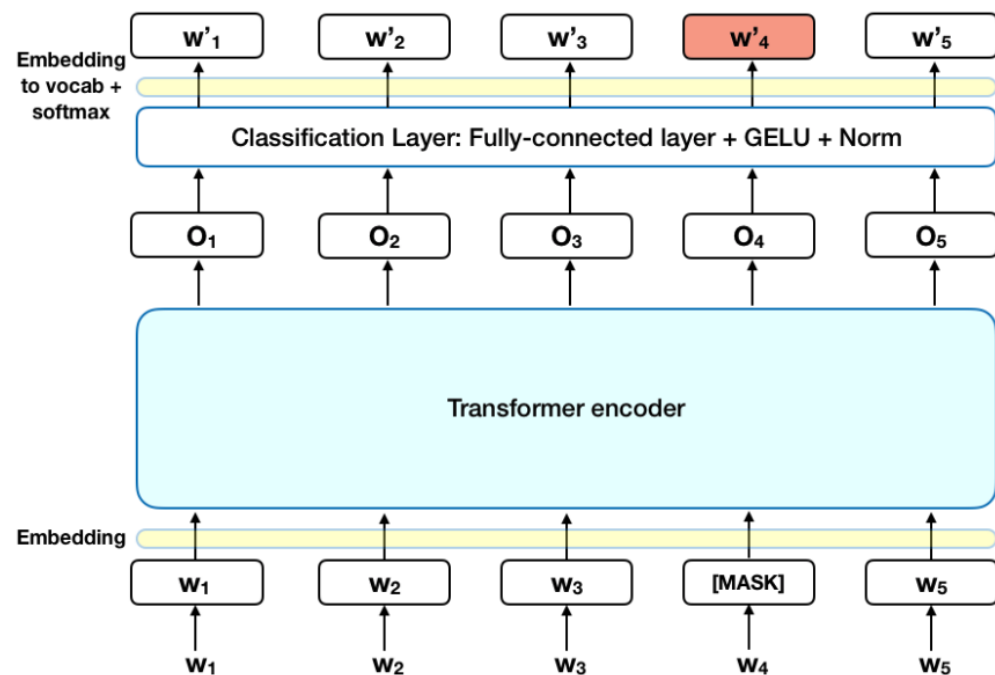
Encoder



GPT

Decoder

Masked LM (MLM)



input tokens의 일정 비율을 마스킹하고
마스킹 된 토큰을 예측하는 과정

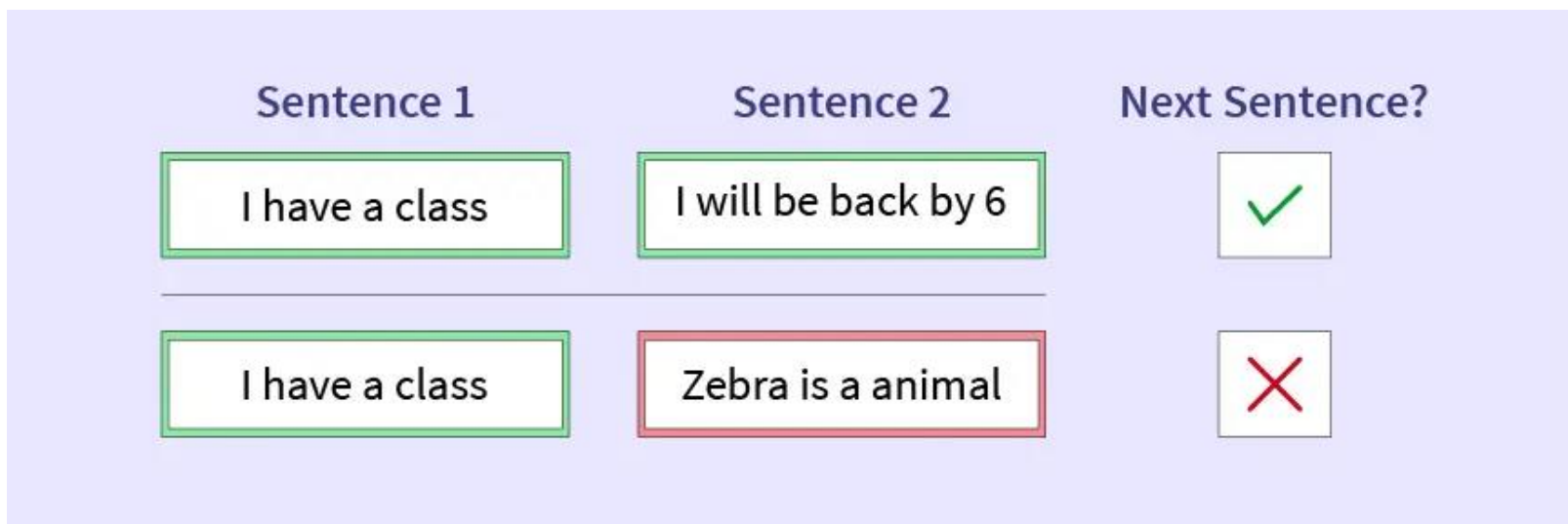
80% : token을 **[MASK] token**으로 바꾼다.
ex) my dog is hairy -> my dog is [MASK]

10% : token을 **random word**로 바꾼다.
ex) my dog is hairy -> my dog is apple

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

$$L_{MLM}(\theta) = - \sum_{t \in M} \log P(x_t | x_{\setminus t}; \theta)$$

Next Sentence Prediction (NSP)



Question-answering(QA), Natural Language Interference(NLI) 등의 두 문장 사이의 관계를 이해해야 하는 task에 대해 학습시키기위해 NSP를 학습

모델은 두 문장 A, B를 입력으로 제공받는데,
절반은 두 번째 문장인 B는 실제로 A의 다음 문장으로 구성되고,
나머지 50%의 경우 B는 전혀 관계가 없는 임의의 문장으로 제공된다.
B가 다음 문장인 경우 IsNext, 임의의 문장인 경우 NotNext 라고 라벨링 하게 된다

GPT-2

GPT-2 (Generative Pre-training Transformer)

GPT-1 문제점

- Pre-Training + Fine-tuning의 방법으로 개발된 모델들은 데이터의 분포가 바뀌면 불안정해진다
- 특정 Task에서만 뛰어난 능력을 발휘한다



많은 Task에 적용 가능한 더 범용적인 모델을 개발할 필요성

GPT-2 (Generative Pre-training Transformer)

1. 고품질의 데이터

사람에 의해 필터링된 글만을 사용했다.

- Reddit에서 3 karma 이상을 받은 글에 포함된 외부 링크의 글을 가져옴
- 45M 개의 링크를 가져옴
- 2017년 12월 이후의 글과 위키피디아 글은 제거함 :
위키피디아는 다른 dataset에서 흔하고, training과 test 단계에서의 데이터가 겹치는 문제로 인해 분석이 복잡해질 수 있기 때문에 제외했다.
- 중복제거 등을 거쳐 8M 개의 문서, 40GB의 텍스트를 확보

GPT-2 (Generative Pre-training Transformer)

2. Byte Pair Encoding (BPE) :

글자(byte)와 단어의 적당한 중간 단위를 쓰는 방식

자주 나오는 symbol sequence의 단어 수준 입력과

자주 나오지 않는 symbol sequence의 글자수준 입력을 적절히 보간

BPE는 subword 기반의 인코딩 방법으로 문자 단위로 단어를 분해하여 Vocabulary를 생성하고, 반복을 통해 빈도수가 높은 문자 쌍을 지속적으로 Vocabulary에 추가하는 bottom-up방법이다(Greedy). 예를 들자면,

$$Vocab_{word} = \{appel, avaiable, capable\}$$

$$Vocab_{char} = \{a, p, l, e, v, i, b, c, p\}$$

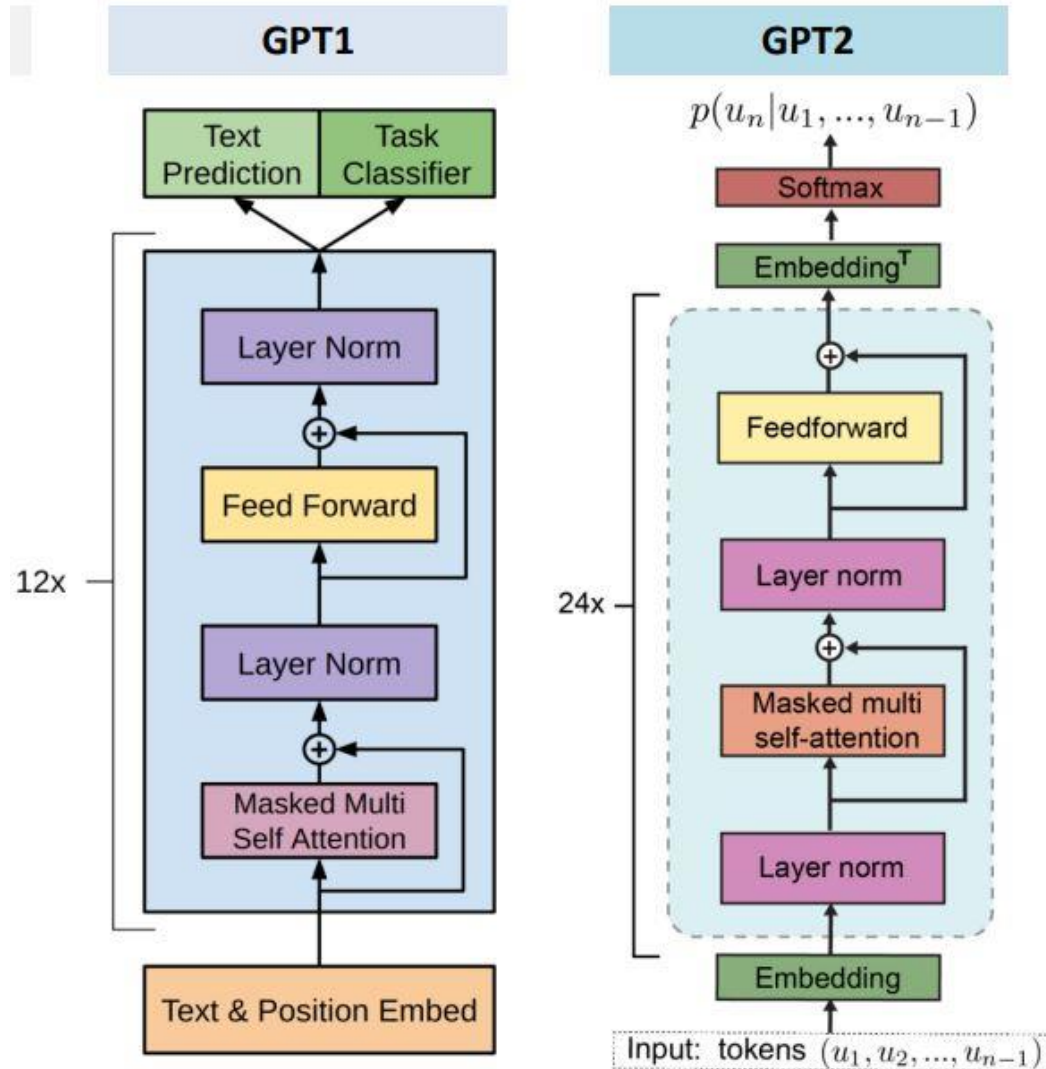
우선 {apple, available, capable} 로 구성된 Word Vocabulary로 부터 Character vocabulary {a,p,l,e,v,i,b,c,p}를 얻는다.

그 후, 매 회 반복을 통해 le, ble, able과 같이 함께 자주 등장하는 문자 쌍을 Character vocabulary에 greedy 하게 추가하는 방법이다.

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

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

GPT-2 (Generative Pre-training Transformer)



3. 모델 수정

- Layer Norm의 위치를 어텐션 블록 앞으로 이동
- 사전의 단어수 50,257개로 확장
- 문맥 고려 범위(context size)가 512 → 1024 개의 token으로 증가
- batch size = 512로 증가

논문 구현

```

147 def model(hparams, X, past=None, scope='model', reuse=False):
148     # 모델 스코프 정의, 재사용 설정 가능
149     with tf.variable_scope(scope, reuse=reuse):
150         results = {} # 결과를 저장할 딕셔너리 초기화
151         batch, sequence = shape_list(X) # 입력 x의 배치 크기와 시퀀스 길이 추출
152
153         # 위치와 토큰 임베딩을 위한 변수 생성
154         wpe = tf.get_variable('wpe', [hparams.n_ctx, hparams.n_embd],
155                                initializer=tf.random_normal_initializer(stddev=0.01))
156         wte = tf.get_variable('wte', [hparams.n_vocab, hparams.n_embd],
157                                initializer=tf.random_normal_initializer(stddev=0.02))
158         # 과거 시퀀스 길이 계산
159         past_length = 0 if past is None else tf.shape(past)[-2]
160         # 입력 x에 대한 토큰 및 위치 임베딩 적용
161         h = tf.gather(wte, X) + tf.gather(wpe, positions_for(X, past_length))
162
163         # 트랜스포머 레이어를 위한 준비
164         presents = [] # 각 레이어의 출력을 저장할 리스트
165         pasts = tf.unstack(past, axis=1) if past is not None else [None] * hparams.n_layer # 과거 데이터 언패킹
166         assert len(pasts) == hparams.n_layer # 레이어 수 검증
167         for layer, past in enumerate(pasts):
168             # 각 레이어를 순회하면서 트랜스포머 블록 적용
169             h, present = block(h, 'h%d' % layer, past=past, hparams=hparams)
170             presents.append(present) # 현재 레이어의 출력 저장
171         results['present'] = tf.stack(presents, axis=1) # 모든 레이어의 출력을 스택하여 결과에 추가
172         h = norm(h, 'ln_f') # 마지막 레이어의 출력 정규화
173
174         # 언어 모델 로스 계산을 위한 로짓 생성
175         h_flat = tf.reshape(h, [batch*sequence, hparams.n_embd]) # 2D로 재배열
176         logits = tf.matmul(h_flat, wte, transpose_b=True) # 임베딩 가중치를 사용한 로짓 계산
177         logits = tf.reshape(logits, [batch, sequence, hparams.n_vocab]) # 원래 차원으로 재배열
178         results['logits'] = logits # 로짓 결과 저장
179         return results # 최종 결과 반환

```

논문 구현(1)

```
1  import os
2  from dotenv import load_dotenv
3  import openai
4
5  # Load environment variables from .env file
6  load_dotenv()
7
8  # Set API key
9  openai.api_key = os.getenv("OPENAI_API_KEY")
10
11 response = openai.ChatCompletion.create(
12     model="gpt-3.5-turbo",
13     messages=[
14         {"role": "system", "content": "You are a helpful assistant."},
15         {"role": "user", "content": "스타벅스 창업자의 정보를 알려줘"}
16     ]
17 )
18
19 print(response['choices'][0]['message']['content'])
```

구현 결과

스타벅스의 창립자는 제리 볼드윈, 고든 보워커, 조 슈르로더 세 명입니다. 처음으로 스타벅스는 1971년에 미국 시애틀에서 창립되었습니다. 이후 스타벅스는 전 세계적으로 유명한 커피 체인으로 성장하였습니다.

PS C:\Playground> █

Reference

- **Improving Language Understanding by Generative Pre-Training (GPT-1)**

Alec Radford/ 11 JUN 2018

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

Jacob Devlin/ 24 May 2019

- **Language Models are Unsupervised Multitask Learners**

Alec Radford/ 14 FEB 2019

Q & A