# Gpt, Gpt-3 모델(개념정리)

GPT의 정의

📍

1. GPT는 거대한 양의 텍스트 데이터를 통해서 사전 훈련된 언어모델이며 다양한 분야의 언어 태스크가 가능
2. GPT는 딥러닝을 통해 주어진 텍스트 기반의 입력에 대해서 사람과 유사한 텍스트를 생성한다
3. GPT는 OpenAI에서 지능형 시스템을 위해 개발한 언어모델이며 ChatGPT 프로젝트에 활용되었다

▼ 연도별 모델정리표

**2018년**

- **6월: GPT-1 (OpenAI)**

**2019년**

- **2월: GPT-2 (OpenAI)**
- 11월: BERT (Google)

**2020년**

- 5월: T5 (Google)
- **6월: GPT-3 (OpenAI)**

**2021년**

- 1월: DALL-E (OpenAI)
- 4월: CLIP (OpenAI)
- 6월: Codex (OpenAI)
- 10월: Gopher (DeepMind)
- 11월: MT-NLG (Microsoft & NVIDIA)

**2022년**

- 4월: GPT-3.5 (OpenAI)
- 7월: BLOOM (BigScience)
- 11월: ChatGPT (OpenAI)

**2023년**

- 3월: GPT-4 (OpenAI)
- 6월: LLaMA (Meta)
- 11월: Claude 2 (Anthropic)

**2024년**

- 2월: GLaM (Google)
- 2월: SOLAR (Meta)
- 5월: PaLM 2 (Google)

---

## GPT 모델 기본 구조

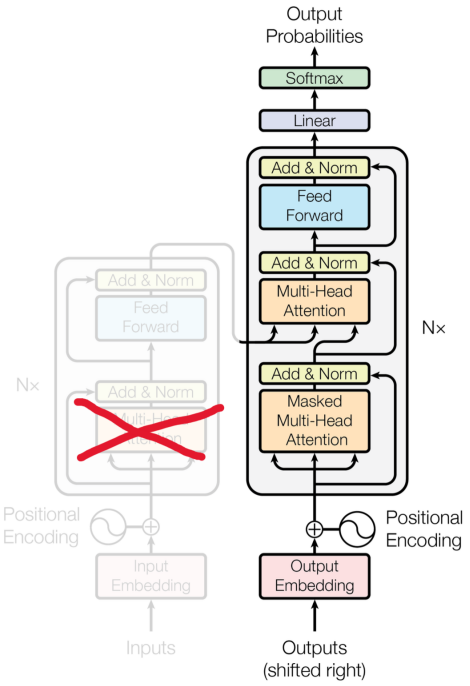
-GPT는 사람의 언어 구사를 흉내내도록 학습된 딥러닝 모델.

-Autoregressive Model 사용

> 자기 자신이 생성한 텍스트를 다시 자신의 입력으로 받아서 다시 출력 텍스트를 생성하는 자기회귀적인 특성.

-신경망 구조

> GPT모델의 구조에는 기본적으로 Transformer모델이 사용되지만, Decoder 구조만(약간 변형)으로 구성됨.




Transformer 모델과의 차이점:

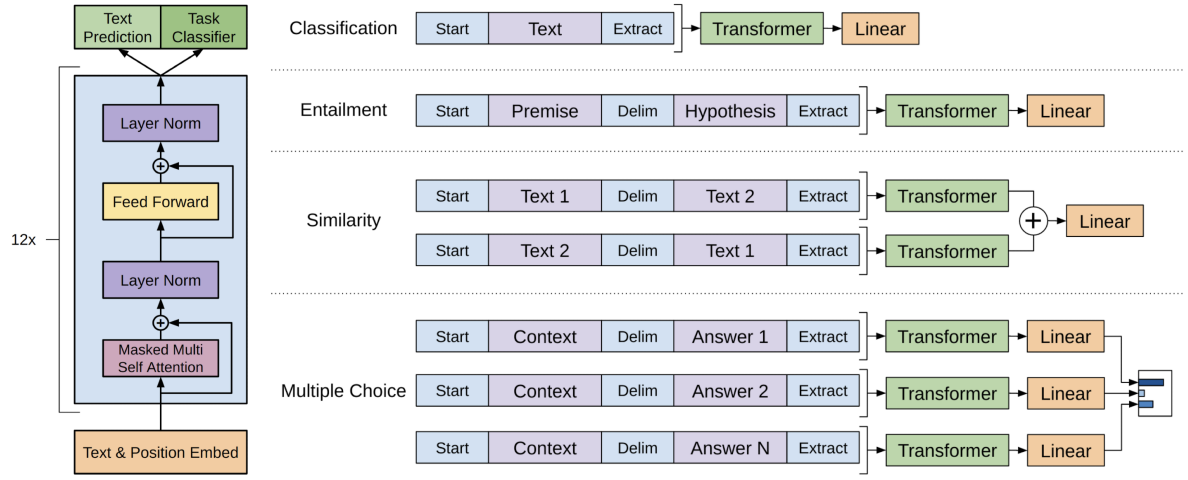
Transformer의 Encoder " Multi-Head Attention메커니즘"

: 시퀀스의 Self Attention과정에서 미래 시점의 토큰을 예측하도록 학습을 수행하는 구조

→ GPT에서는 Encoder시퀀스에 대한 Attention을 수행할 필요가 없어서 Multi-Head Attention메커니즘 계층이 제거됨

Multi-Head Attention!!!!
"GPT에서의 Multi-Head Attention은 여전히
**이전의 문맥 토큰들에 대한 정보**를 종합하는 중요한 역할을 하며, 다만 **미래의 토큰들을 참조하지 않도록 제한**된 방식으로 변형되어 사용됩니다."

**미래의 토큰을 참조하지 못하도록 막는 것**은 GPT와 같은 자기 회귀적 모델이 자연스럽고 논리적인 문장을 생성하는 데 필요한 중요한 원칙이다.




Transformer의 변형된 Decoder구조 + 입력방식 → GPT의 구조

→ Task별로 구조가 다르게 형성되어 있다. (input의 형식이 다르다. )

## GPT와 BERT의 차이

공통점 : Transformer활용 모델

차이점 :

GPT → 입력 토큰 시퀀스에 대해서 미래 시점은 확인하지 않고 과거 시점들에 대해서만 Attention을 수행하여 현재시점에 대한 결과물을 출력한다. **디코더-only** 모델로, 문장을 **생성**하는 데 중점을 둔다. GPT는 **자기 회귀적(autoregressive)** 방식으로 문장을 생성하며, 이전에 나온 단어들만 참조해서 다음 단어를 예측한다.

BERT → 인코더의 역할을 수행하는 모델로서, 디코더와 다르게 입력 토큰 시퀀스에서 현재 시점의 토큰에 대해 인코딩을 하는 경우 과거와 미래 시점의 토큰 시퀀스를 Attention에 모두 활용한다. **인코더-only** 모델로, 문장 전체를 한꺼번에 보고 **이해**하는 데 중점을 둔다. 이는 **양방향(bidirectional)** Attention을 사용해, 문장의 앞뒤 단어 모두를 참조하여 문맥을 해석한다.

## GPT의 비지도 사전훈련

GPT는 사람이 생성한 말뭉치 데이터의 언어 모델링을 수행하도록 학습된다.

$$\mathcal{U} = \{u_1, \cdots, u_n\}.$$

주어진 식을 비지도 말뭉치의 토큰이라고 정의한다.

$$L_1(\mathcal{U}) = \sum_i \log p\left(u_i | u_{i-k}, \cdots, u_{i-1}\right).$$

-$L_1(u)$는 GPT의 비지도 사전 훈련에 사용된 표준적인 언어 모델링 목적함수이다. (Likelihood 함수형태)

-$k$개의 이전 토큰들을 참고하여 다음 토큰을 예측. ($k$ : 문맥 윈도우 크기)

다층 Transformer디코더로 구성되어 언어 모델링을 수행하면서 입력 문맥 토큰들에 Multi-Head Attention을 수행하고 이 결과물을 위치별 완전 연결 순전파 망을 활용하여 타켓 토큰에 대한 출력 분포를 생성한다.

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{u}\mathbf{W}_e + \mathbf{W}_p. \\ \mathbf{h}_l &= \text{transformer-block}(\mathbf{h}_{l-1}),\ \forall i \in [1, n]. \\ p(u) &= \text{softmax}\left(\mathbf{h}_n \mathbf{W}_e^{\mathrm{T}}\right). \end{aligned}$$

언어모델링 실행 식

## 식1. 입력 문맥 처리

먼저 GPT는 입력된 단어들을 벡터로 변환한 후, 각 단어의 위치 정보를 더하여 모델이 이해할 수 있는 형태로 만든다. 이 과정은 모델이 단어의 *의미뿐만 아니라 단어의 위치 정보도 함께 이해할 수 있도록* 해준다. 예를 들어, "I love deep learning"이라는 문장이 있다면, 각 단어는 숫자 벡터로 변환된다.

## 식2. Transformer 블록을 통해 처리

GPT는 여러 개의 층으로 이루어진 구조를 가지고 있으며, 각 층에서는 **Multi-Head Attention**을 사용해 입력된 문맥에서 중요한 단어들을 찾아내고, 단어들 간의 관계를 분석한다. 이 과정을 여러 층에서 반복하면서 문맥을 점점 더 깊이 있게 이해하게 된다.

## 3. 다음 단어 예측

마지막으로, GPT는 주어진 문맥에서 다음에 나올 단어를 예측한다. 주어진 문장에서 다음 단어가 될 확률을 계산하여 가장 적합한 단어를 선택하는 방식이다.

▼ 수식기호 설명

$u = (u_{-k}, \cdots, u_{-1})$ : 토큰들의 문맥 벡터

"I love deep learning"이라는 문장에서 "learning"을 예측하려 한다면, $u_{-1}$은 "deep", $u_{-2}$는 "love", $u_{-3}$는 "I" 이다.

$n$ : 디코더 계층 개수

Transformer는 여러 층(layer)으로 구성되어 있는데, 여기서 $n$은 모델의 깊이를 나타내며, 디코더 블록의 개수이다.

$W_e$ : 토큰 임베딩 행렬

단어 "apple"이 주어졌을 때, $W_e$는 이를 벡터로 변환해 모델이 처리할 수 있게 합니다.

$W_p$ : 위치 임베딩 행렬

Transformer 모델은 시퀀스 순서를 내재적으로 이해하지 못하기 때문에, 토큰이 등장한 위치 정보를 추가해야 함. $W_p$는 각 토큰이 시퀀스에서 어느 위치에 있는지 알려주는 벡터를 제공함.

## GPT의 지도 Fine Tuning

비지도 사전 훈련 이후에는, NLP태스크를 수행하기 위해 지도 Fine Tuning을 진행한다.

**지도 파인 튜닝**은 이미 학습된 모델을 새로운 데이터에 맞춰 미세하게 조정하는 과정이다.

입력 시퀀스가 사전 훈련된 모델을 통과하면 마지막 Transformer 블록의 활성 값이 출력되고(h), 이 벡터는 파라미터가 $W_y$인 추가적인 선형 계층에 통과되어 레이블 y를 예측하게 된다.

$$p(y|x^1, \cdots, x^m) = \text{softmax}\left(\mathbf{h}_n^m \mathbf{W}_y\right).$$

모델은 입력된 데이터를 처리하여 예측을 하고, 그 예측 결과와 실제 정답 사이의 차이를 손실 함수로 계산한다.

추가적으로, 보조 목표 함수를 도입해 모델의 성능을 더 향상시킬 수 있다.

---

## GPT모델 등장 전의 한계점

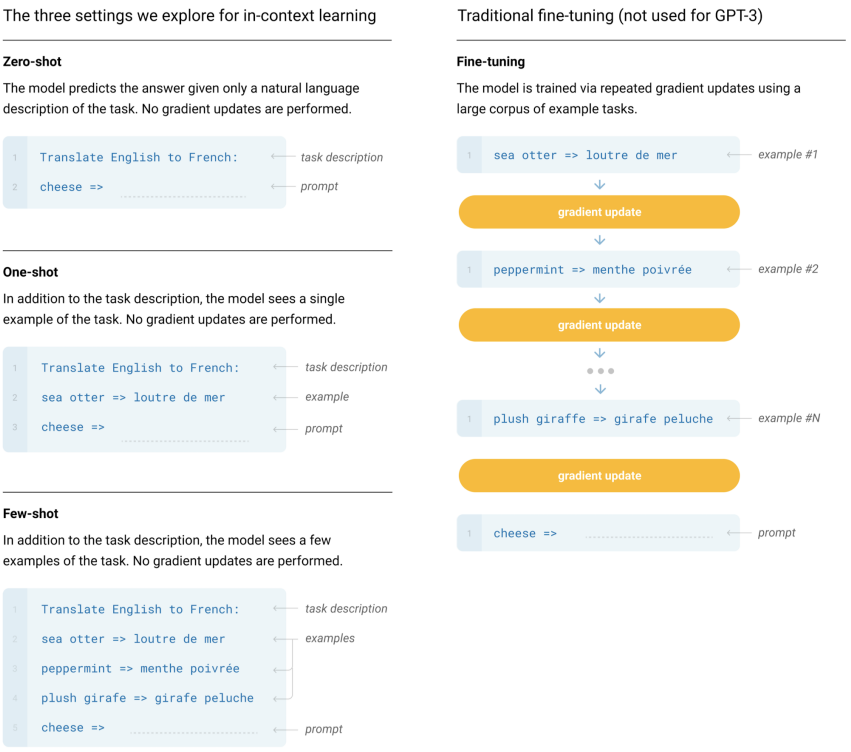
GPT모델 이전에는 대량의 라벨링 데이터가 필요했었다. 따라서 자신이 훈련된 데이터셋을 넘어서는 다른 태스크는 수행될 수 없었다. 라벨링 된 데이터를 구하기 힘들다는 한계도 많았었다.

## GPT-1

GPT-1은 OpenAI가 2018년에 제안한 최초의 생성형 AI 모델로, *라벨링되지 않은 대량의 텍스트 데이터를 활용해 사전 훈련을 할 수 있었다*. 이를 통해 감정 분석, 분류, 질의응답 등 다양한 NLP 작업에서 적은 양의 지도 학습 데이터로도 우수한 성능을 발휘할 수 있었다.

GPT-1은 **BooksCorpus 데이터셋**을 사용해 **12계층의 Transformer 디코더 구조**로 학습되었으며, **Zero-Shot 학습** 성능을 보여주었다. 이는 GPT-1이 특정 태스크에 특화되지 않고도 *다양한 작업을 수행*할 수 있는 일반적인 AI 모델임을 증명했다.

## GPT-3




GPT-3의 학습 방식과 전통적인 파인 튜닝 방식의 차이

GPT-3는 2020년에 발표된 매우 거대한 언어 예측 및 생성 모델로, 1750억 개의 파라미터를 가지고 있으며, 5000억 개의 단어를 학습했다. 문맥 윈도우 크기가 2049~4096 토큰에 이르러 긴 문맥도 처리할 수 있다. GPT-3는 자연스러운 언어 생성뿐만 아니라, 수식 처리와 코드 구현 같은 새로운 태스크도 수행할 수 있으며, 비즈니스 활동에도 신속하고 정확하게 활용될 수 있다.

GPT-3는 **Common Crawl** 데이터를 기반으로 여러 필터링 및 정제 작업을 거쳐 학습되었으며, 고품질 말뭉치인 WebText, Books1, Books2, 위키피디아 등의 데이터를 추가했다. 또한, GPT-3는 **Zero-Shot, One-Shot, Few-Shot 학습**을 통해 추가적인 학습 없이 프롬프트 설명만으로도 다양한 NLP 태스크에서 높은 성능을 보여준다.

## 1. Zero-shot 학습

- 모델이 태스크 설명만 주어졌을 때, 추가적인 학습 없이 바로 결과를 예측한다.
- 예: "cheese"를 번역하는 태스크 설명만 주어지고 모델이 답을 예측함.

## 2. One-shot 학습

- 모델이 태스크 설명과 더불어 한 가지 예시를 보고, 그 예시를 바탕으로 추가적인 입력에 대해 결과를 예측한다.
- 예: "sea otter"를 번역한 예시를 보고, "cheese"도 번역한다.

## 3. Few-shot 학습

- 모델이 태스크 설명과 몇 가지 예시들을 보고, 이를 바탕으로 추가적인 입력에 대해 결과를 예측한다.
- 예: "sea otter", "peppermint", "plush giraffe"를 번역한 예시를 보고 "cheese"를 번역한다.

### 전통적인 파인 튜닝

- 기존의 파인 튜닝 방식은 여러 개의 예시를 모델에 입력하고, 각 예시마다 모델의 가중치가 업데이트된다. 예시를 많이 주면 줄수록 모델이 그 태스크에 맞춰서 학습하게 된다.
- GPT-3는 이 방식 대신 Zero-shot, One-shot, Few-shot 학습을 통해 추가적인 파인 튜닝 없이도 높은 성능을 발휘할 수 있다.