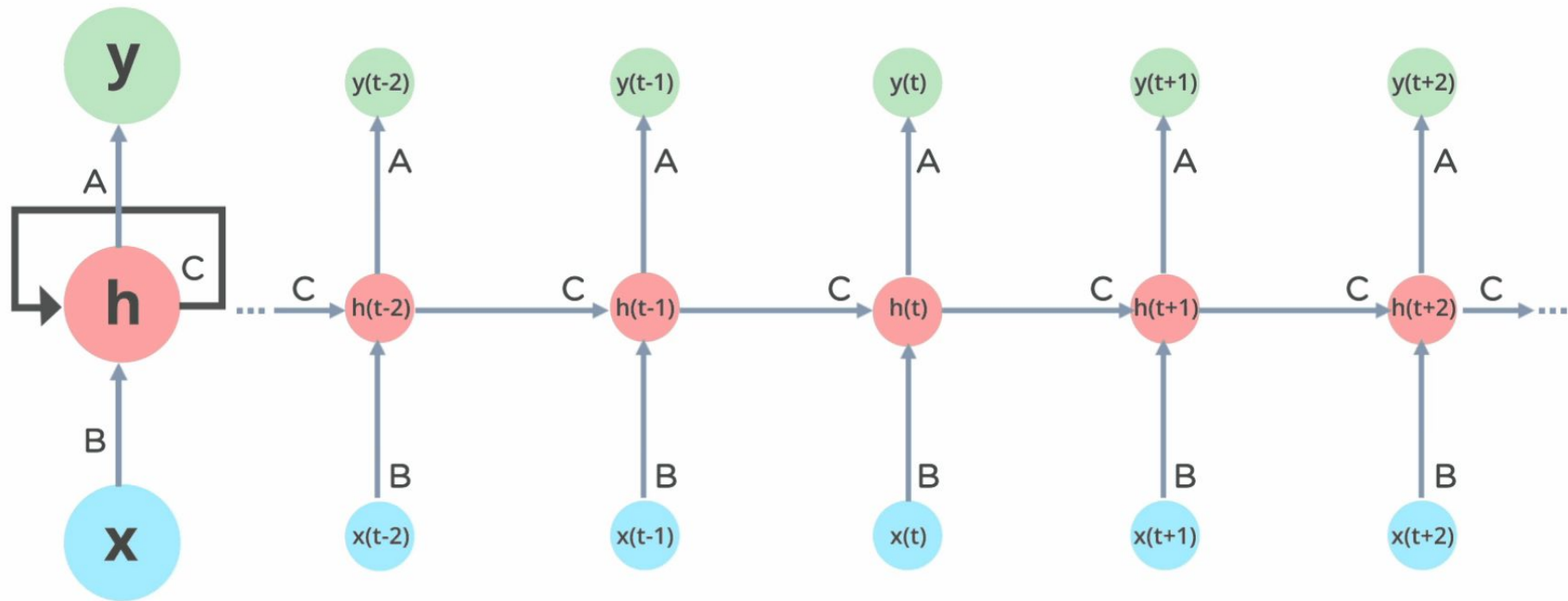


# Language Model and Emergence

소준섭

# RNN



**RNN이 할 수 있는 일들?**

# 주로 순서를 다루는 데이터들을 처리할 수 있습니다.

## Natural Language Processing



### Language Modelling

📊 61 benchmarks  
3322 papers with code



### Question Answering

📊 230 benchmarks  
2382 papers with code



### Machine Translation

📊 94 benchmarks  
1971 papers with code



### Text Generation

📊 266 benchmarks  
1244 papers with code



### Sentiment Analysis

📊 97 benchmarks  
1193 papers with code

# 주로 기계번역, 자연어처리에 사용되었습니다.

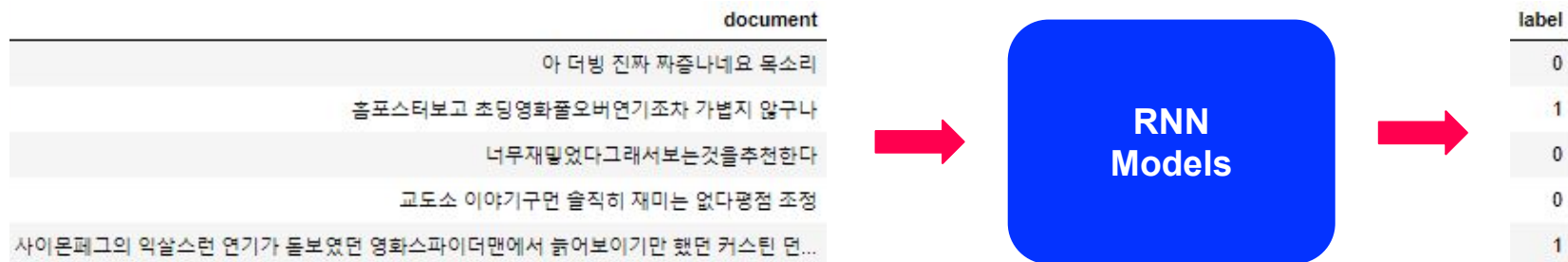
## Natural Language Processing



	id	document	label
0	9976970	아 더빙 진짜 짜증나네요 목소리	0
1	3819312	흠포스터보고 초딩영화풀어버연기조차 가볍지 않구나	1
2	10265843	너무재밌었다그래서보는것을추천한다	0
3	9045019	교도소 이야기구면 솔직히 재미는 없다평점 조정	0
4	6483659	사이몬페그의 익살스런 연기가 돋보였던 영화스파이더맨에서 늙어보이기만 했던 커스틴 던...	1

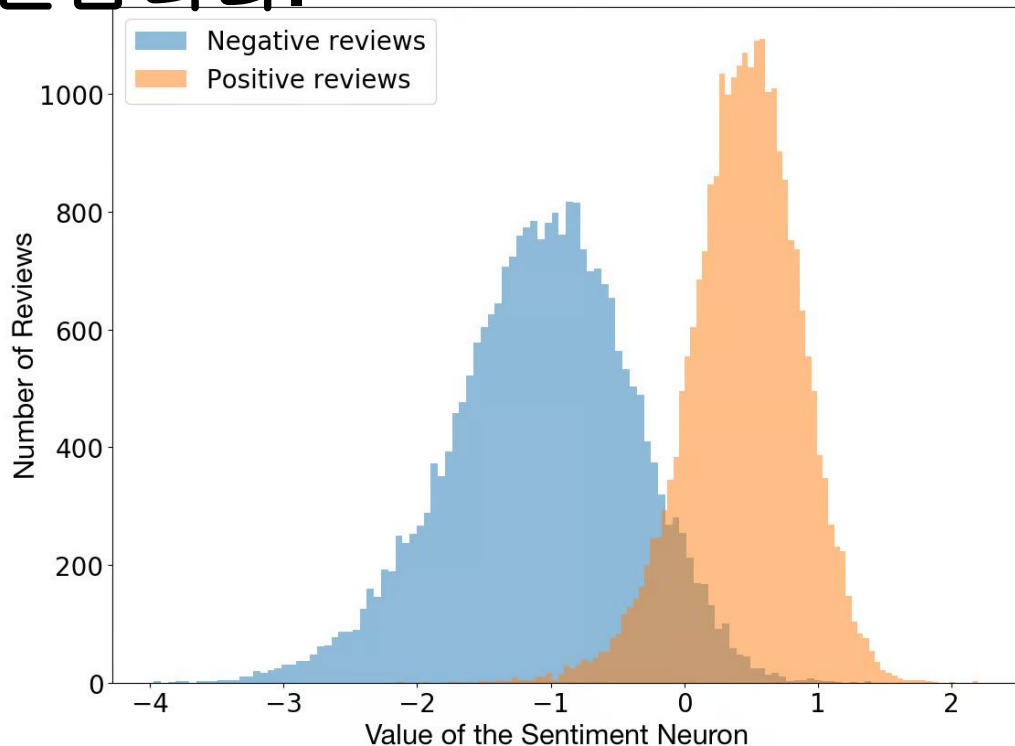
감정분석용  
데이터셋

주로 기계번역, 자연어처리에 사용되었습니다.



그런데, 자연어 비지도학습에서  
특이한 점이 발견됩니다.

# 학습된 RNN 모델에서 Sentiment neuron을 발견합니다.



**Sentiment neuron의 값에 따라서 부정, 긍정 리뷰가 구분되었습니다.**

**뉴런의 값을 고정 시키면 원하는 감정의 문장을 생성할 수 있습니다.**



# Sentiment neuron을 발견합니다.

---

## 긍정적인 감정으로 고정됨

내가 찾던 바로 그 것. 핏이 좋은 바지, 솔기가 내가 가지고 있는 다른 바지와 색상 대비가 정확히 일치합니다. 적극 추천하고 또한 매우 행복합니다!

이 제품은 예상되는 기능을 수행합니다. 혹시라도 교체 코드가 필요할 경우를 대비해 저는 항상 이 세 개를 주방에 보관합니다.

## 부정적으로 고정된 감정

받은 패키지는 비어 있고 바코드가 없습니다. 시간과 돈 낭비입니다.

훌륭한 작은 아이템. 특별한 장식 없이는 침대 위에 올려놓기가 어렵습니다. 내 추측은 내가 가지고 있던 나사 종류의 부착물과 같습니다.

---

\* 번역본입니다

## Sentiment neuron을 발견합니다.

This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

인공지능 모델이 우리가 사용하는 자연어를 학습 한 후  
따로 정답을 알려주지 않아도 스스로 감정을 학습했습니다.

인공지능 모델이 우리 자연어를  
이해한 것으로 봐도 될까요?

LLM, 초거대 AI의 시작

다양한 문제를 해결하기 위한 범용 AI 모델

# Foundation Models

다양한 문제를 해결하기 위한 범용 AI 모델

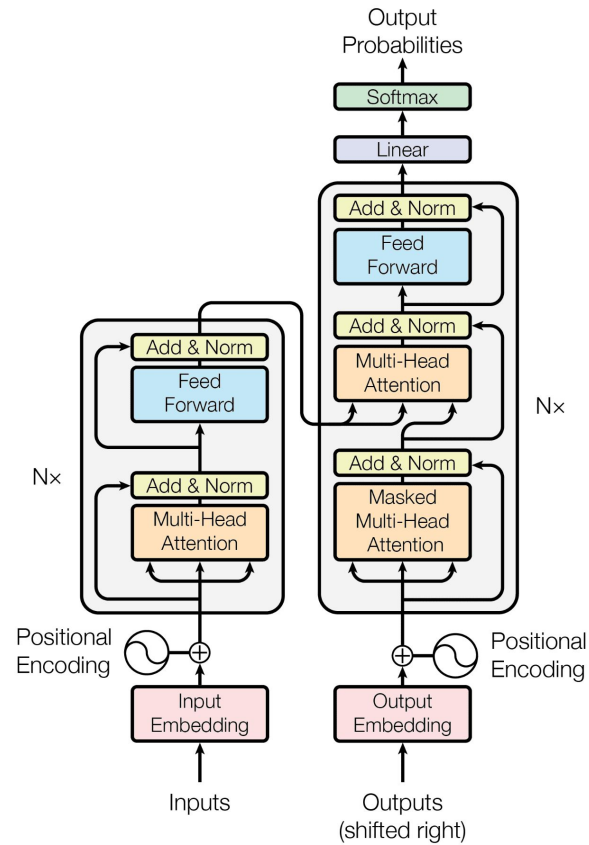
**Foundation Models**

**Transformer**

내부엔 Transformer를 기반으로 하는 모델이 존재합니다.

**Transformer는 RNN의 성능을  
강화시킨 새로운 모델입니다.**

# Transformer





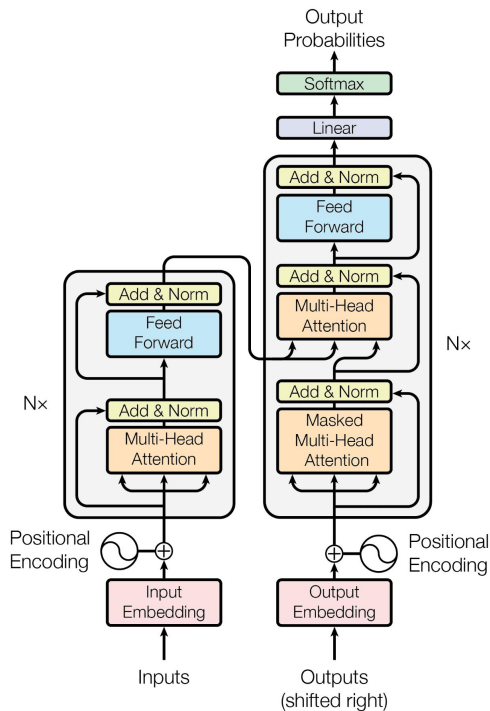
# Transformer

| 원래는 번역 모델이었습니다.

NLP (Machine Translation)  
Computer Vision  
RL  
Speech synthesis  
ETC ...

현재 다양한 분야에서 사용되고 있습니다.

모든 것은 Attention이 강력한 성능을 보여주기 때문입니다.



**Attention?**

NN을 이용해서 뭐가 중요한 단어인지 계산할 수 있을까?

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

The FBI is chasing a criminal on the run .

각 **vector**는 3가지 역할을 합니다.

Each vector receives three representations (“roles”)

$$\begin{bmatrix} W_Q \end{bmatrix} \times \begin{bmatrix} \text{green} \\ \text{green} \\ \text{green} \end{bmatrix} = \begin{bmatrix} \text{blue} \\ \text{blue} \\ \text{blue} \end{bmatrix}$$

**Query:** vector **from** which the attention is looking

“Hey there, do you have this information?”

$$\begin{bmatrix} W_K \end{bmatrix} \times \begin{bmatrix} \text{green} \\ \text{green} \\ \text{green} \end{bmatrix} = \begin{bmatrix} \text{yellow} \\ \text{yellow} \\ \text{yellow} \end{bmatrix}$$

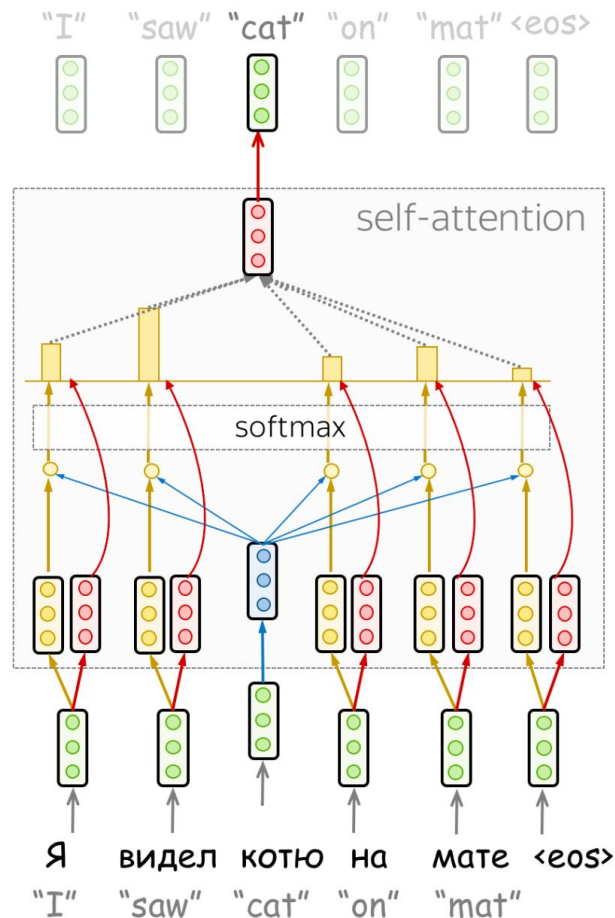
**Key:** vector **at** which the query looks to compute weights

“Hi, I have this information – give me a large weight!”

$$\begin{bmatrix} W_V \end{bmatrix} \times \begin{bmatrix} \text{green} \\ \text{green} \\ \text{green} \end{bmatrix} = \begin{bmatrix} \text{red} \\ \text{red} \\ \text{red} \end{bmatrix}$$

**Value:** their weighted sum is attention output

“Here’s the information I have!”



## query

내가 찾고 정보를 요청하는 것 ( 유튜브 검색어 )

## key

그 정보가 있는지 찾아보는 과정 ( 유튜브 영상의 제목 )

## value

제공된 정보가 얼마나 가치 있는지 ( 유튜브 콘텐츠의 정확도 )

# Foundational Architecture and Specific Model

## | BERT

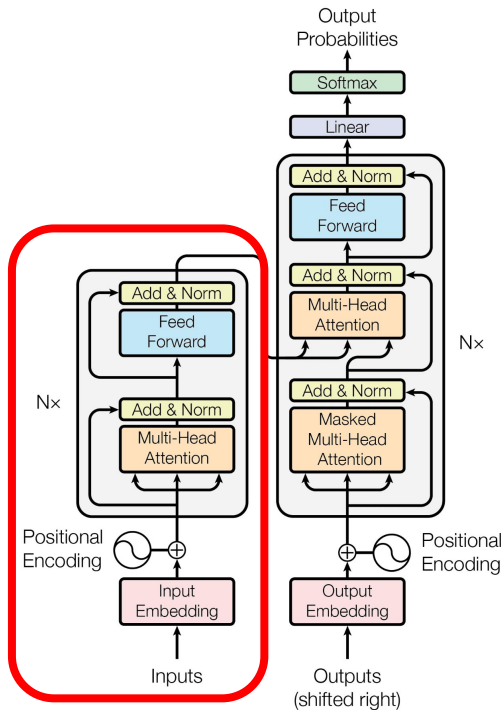
### BERT

Transformer의 기본 아키텍처를 기반으로 제작된 GPT

Decoder 파트만 사용해 아키텍처를 구성했습니다.

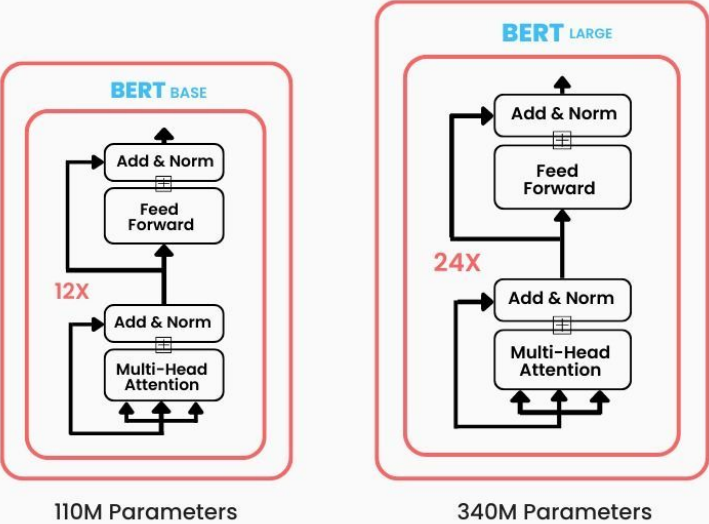
대용량의 데이터셋을 이용해서 미리 학습한  
(Pre-trained Transformer)를 이용

추후 버전에 개선되면서 구조가 조금 변하게 됩니다.



# BERT Architecture

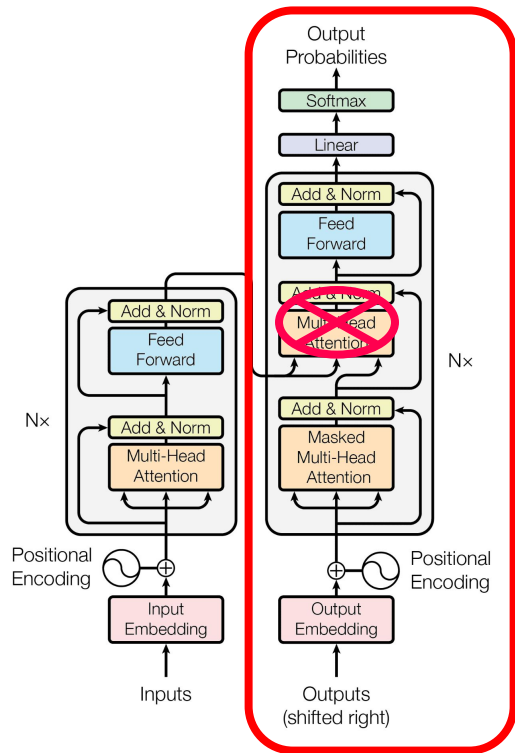
BERT Size & Architecture



Pre-trained Data

Input	[CLS]	my	[MASK]	dog	is	cute	[SEP]	he	[MASK]	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	$E_{my}$	$E_{[MASK]}$	$E_{dog}$	$E_{is}$	$E_{cute}$	$E_{[SEP]}$	$E_{he}$	$E_{[MASK]}$	$E_{likes}$	$E_{play}$	$E_{##ing}$	$E_{[SEP]}$
	+	+	+	+	+	+	+	+	+	+	+	+	+
Sentence Embedding	$E_A$	$E_A$	$E_A$	$E_A$	$E_A$	$E_A$	$E_A$	$E_B$	$E_B$	$E_B$	$E_B$	$E_B$	$E_B$
	+	+	+	+	+	+	+	+	+	+	+	+	+
Transformer Positional Embedding	$E_0$	$E_1$	$E_2$	$E_3$	$E_4$	$E_5$	$E_6$	$E_7$	$E_8$	$E_9$	$E_{10}$		

# Foundational Architecture and Specific Model



| GPT 1, 2, 3, 4

Generative Pre-trained Transformer (GPT)

Transformer의 기본 아키텍처를 기반으로 제작된 GPT

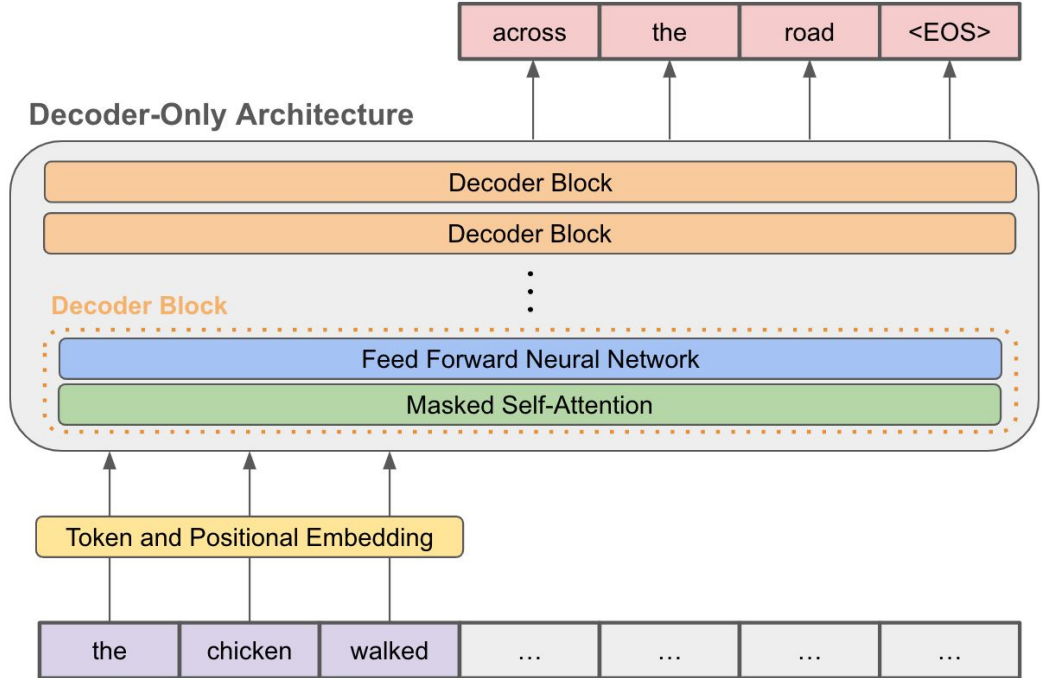
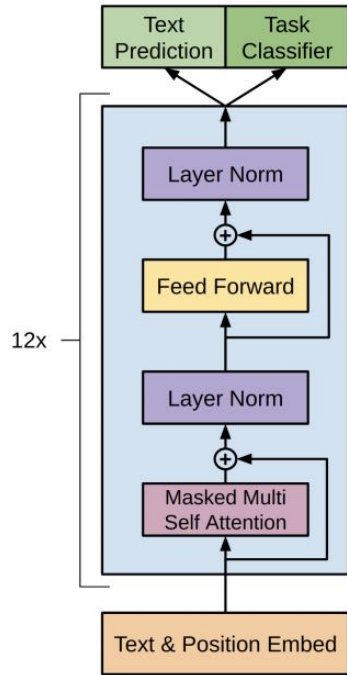
Decoder 파트만 사용해 아키텍처를 구성했습니다.

대용량의 데이터셋을 이용해서 미리 학습한  
(Pre-trained Transformer)를 이용

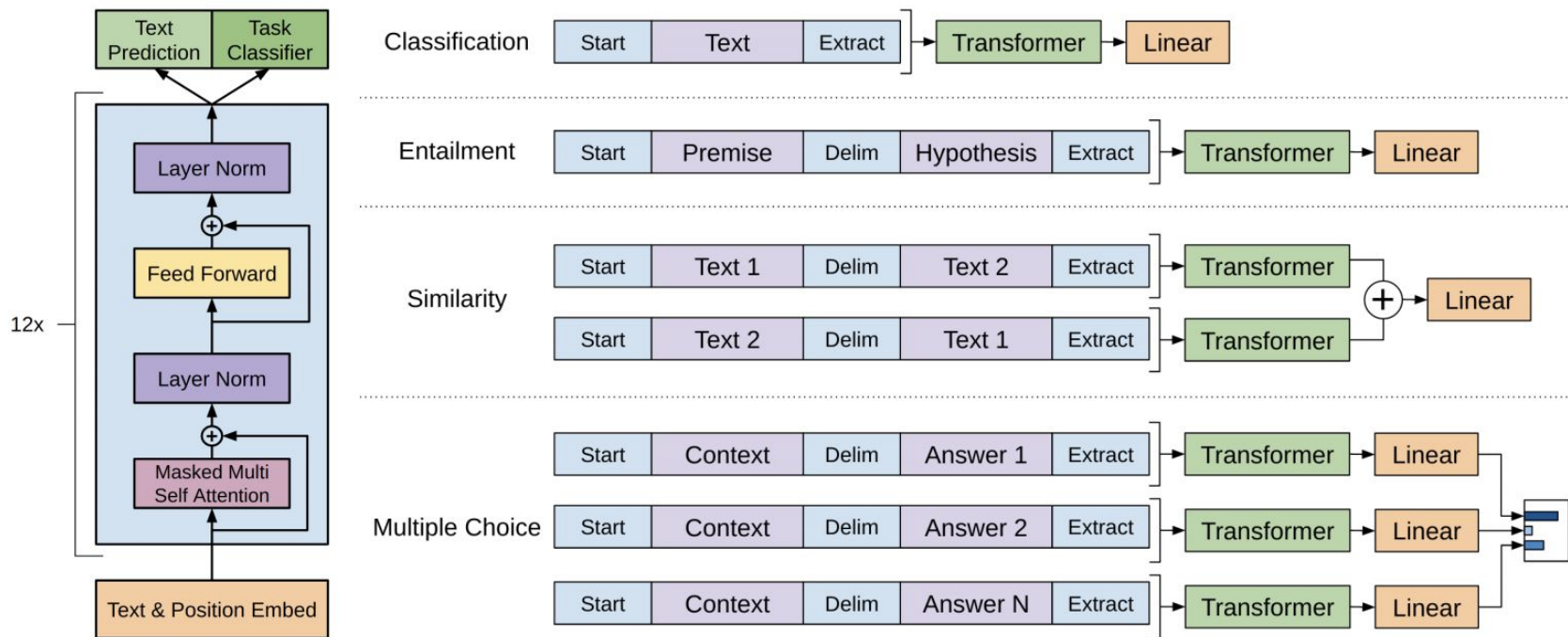
추후 버전에 개선되면서 구조가 조금 변하게 됩니다.



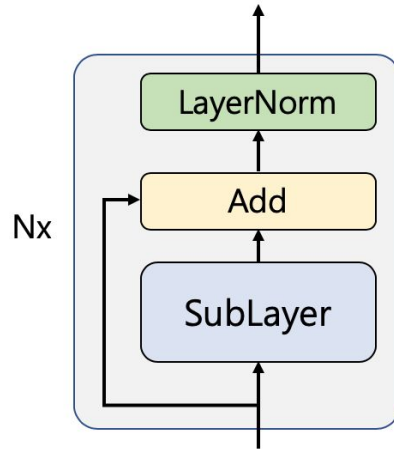
# GPT Architecture



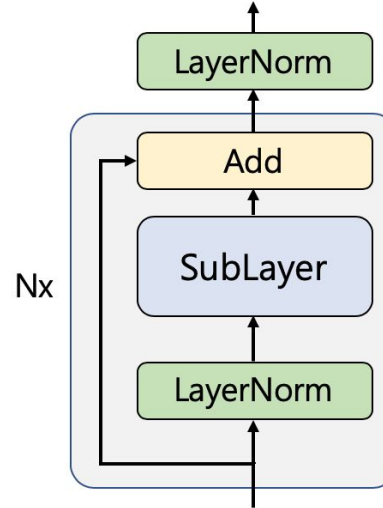
# GPT Architecture



# Adv. Transformer Architecture

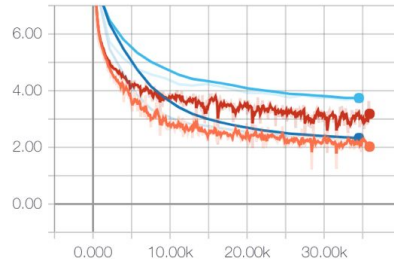


The paper ver.



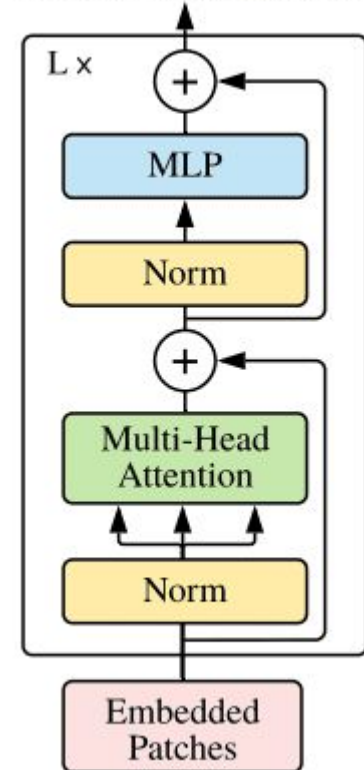
tensor2tensor

loss



- ✓ baseline - train
- ✓ baseline - validation
- ✓ Paper ver. - train
- ✓ Paper ver. - validation

## Transformer Encoder



**그래서 Transformer와 LLM과는  
무슨 관계가 있는걸까?**

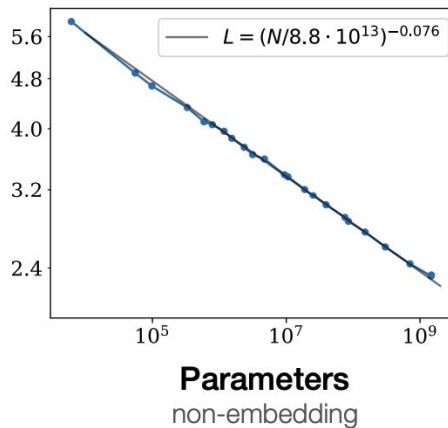
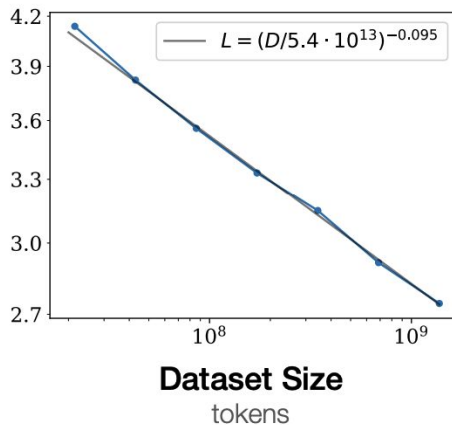
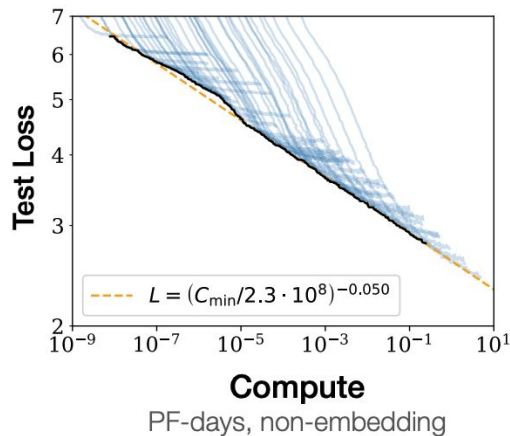
# Scaling Law & Emergence

# Scaling Law For Neural Language Models

데이터의 크기, 계산량, 모델 크기

세 가지 값에 따라 Scaling 했을 때,

성능이 달라지는 부분을 정리 (OpenAI, 2020)



# Scaling Law For Neural Language Models

1. 모델의 크기 (임베딩 제외), 데이터 크기
  - > 두 값을 같이 키우면, 성능은 예측 가능하게 증가
  - > 한 쪽을 고정하면, 어느 순간 성능 향상 중지
2. 모델의 크기 (임베딩 제외) 가 커지면,  
모델 크기에 따른 성능 향상이 보인다.
3. 큰 모델은 작은 모델에 보다 더 적은 데이터로 같은 성능에 도달한다.

# Chincilla, 2022

“Chincilla 이전의 LLMs이 underfitting 되어있다” 고 주장

동일 크기의 모델을 학습하기 위해

Open AI 가 제시한 데이터보다 더 많은 데이터가 필요



# Chinchilla, 2022

Table 1 | **Current LLMs**. We show five of the current largest dense transformer models, their size, and the number of training tokens. Other than LaMDA (Thoppilan et al., 2022), most models are trained for approximately 300 billion tokens. We introduce *Chinchilla*, a substantially smaller model, trained for much longer than 300B tokens.

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
<i>Chinchilla</i>	70 Billion	1.4 Trillion

70B Chinchilla에 1.4T Token을 학습했더니, Gopher보다 좋은 성능을 보였다.

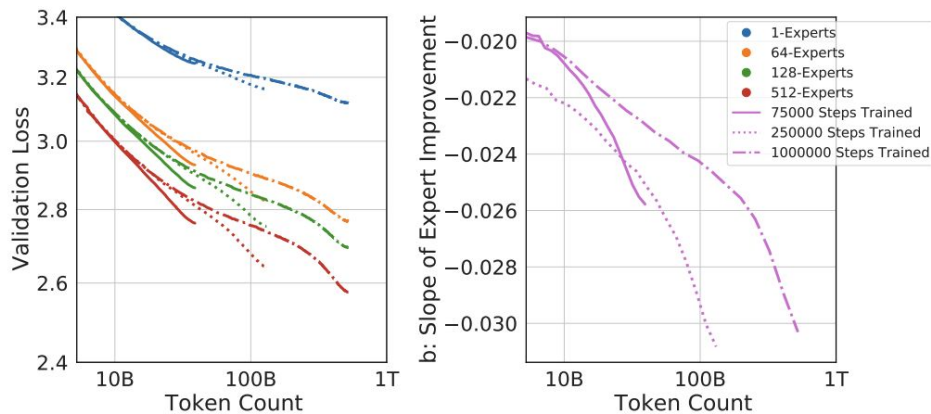
# LLaMA

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	<b>88.0</b>	82.3	-	83.4	<b>81.1</b>	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	<b>80.0</b>	<b>57.8</b>	58.6
	65B	85.3	<b>82.8</b>	<b>52.3</b>	<b>84.2</b>	77.0	78.9	56.0	<b>60.2</b>

Table 3: Zero-shot performance on Common Sense Reasoning tasks.

1.4T Token으로 학습한 33B LLaMA가 chinchilla 성능을 뛰어 넘었다.

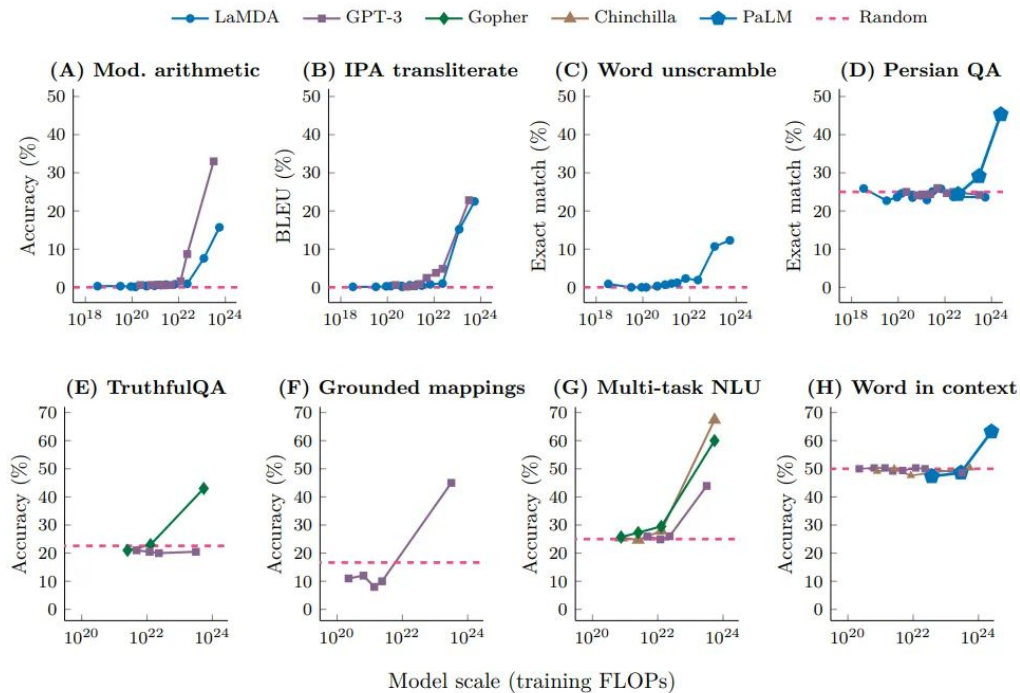
# HyperCLOVA



한국어 데이터셋 561.8B Token, 가장 큰 모델은 82B

# Emergence

## LLM에서 학습하지 않은 task들을 수행해내는 현상



# Emergence

	NSMC (Acc)	KorQuAD (EM / F1)		AI Hub (BLEU) Ko→En   En→Ko		YNAT (F1)	KLUE-STS (F1)
Baseline	89.66	74.04	86.66	40.34	40.41	82.64	75.93
137M	73.11	8.87	23.92	0.80	2.78	29.01	59.54
350M	77.55	27.66	46.86	1.44	8.89	33.18	59.45
760M	77.64	45.80	63.99	2.63	16.89	47.45	52.16
1.3B	83.90	55.28	72.98	3.83	20.03	58.67	60.89
6.9B	83.78	61.21	78.78	7.09	27.93	67.48	59.27
13B	87.86	66.04	82.12	7.91	27.82	67.85	60.00
39B	87.95	67.29	83.80	9.19	31.04	71.41	61.59
82B	88.16	69.27	84.85	10.37	31.83	72.66	65.14

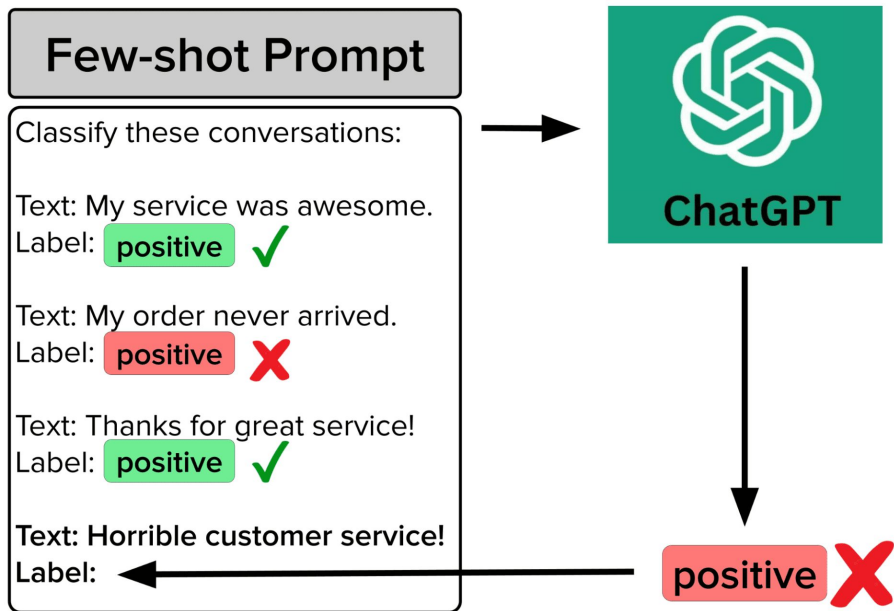
**HyperCLOVA,  
Few-shot Results**

Table 3: Results of in-context few-shot tasks on sentiment analysis, question answering, machine translation, topic classification, and semantic similarity per model size. As baselines, we report the results of BERT-base for NSMC and KorQuAD, and Transformer for AI Hub from [Park et al. \(2020\)](#). mBERT is used for YNAT and KLUE-STS from [Park et al. \(2021\)](#).

**760M > 1.3B > 6.9B > 13B**

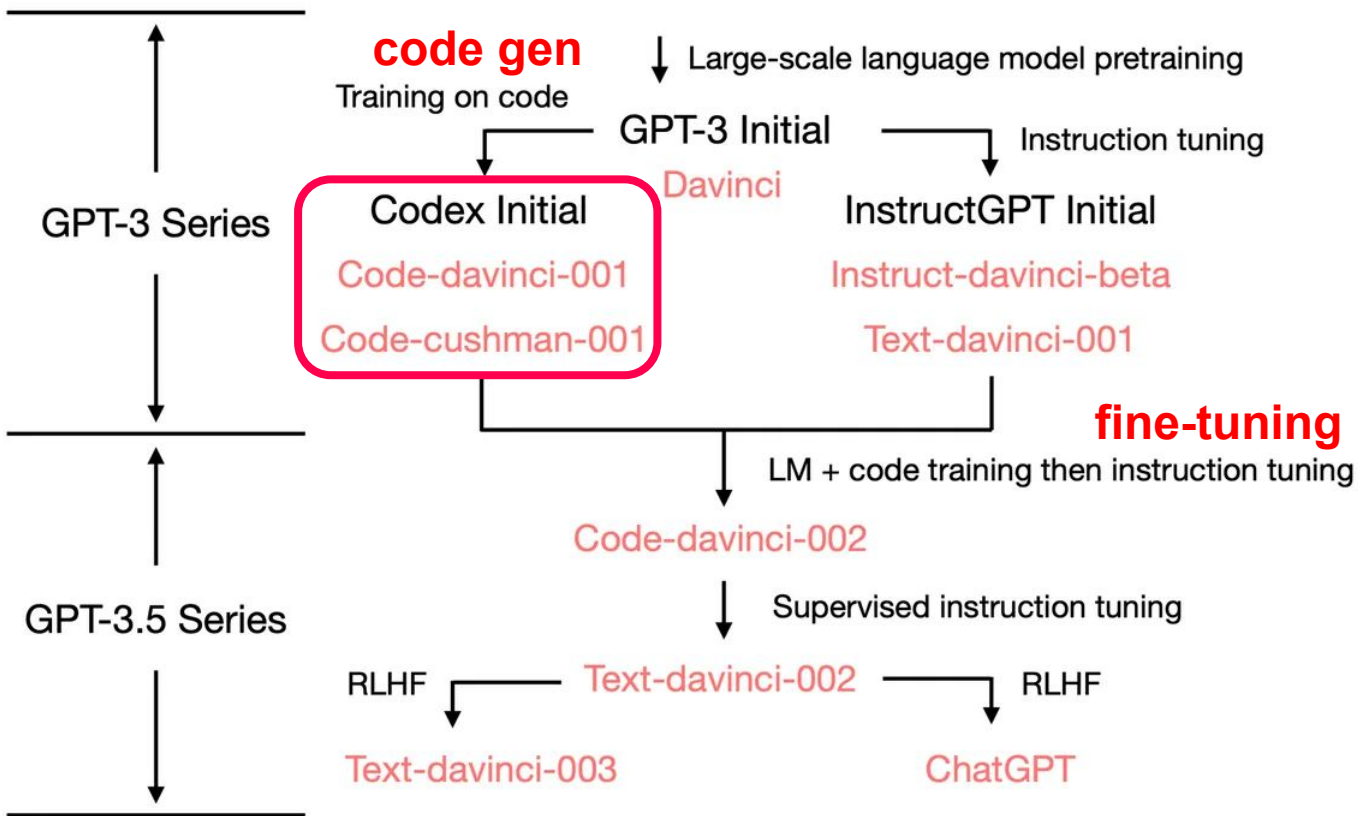
**부근에서 갑자기 성능 증가를 확인 할 수 있다.**

# Few / Zero Shot Learning



따로 모델의 **Weight Update** 없이 새로운 task를 수행할 수 있다.

# GPTs



# GPTs

## 일반 언어 모델

**Q> ChatGPT에 대해 설명해줘**  
A>Transformer에 대해 설명해줘  
GPT에 대해 설명해줘

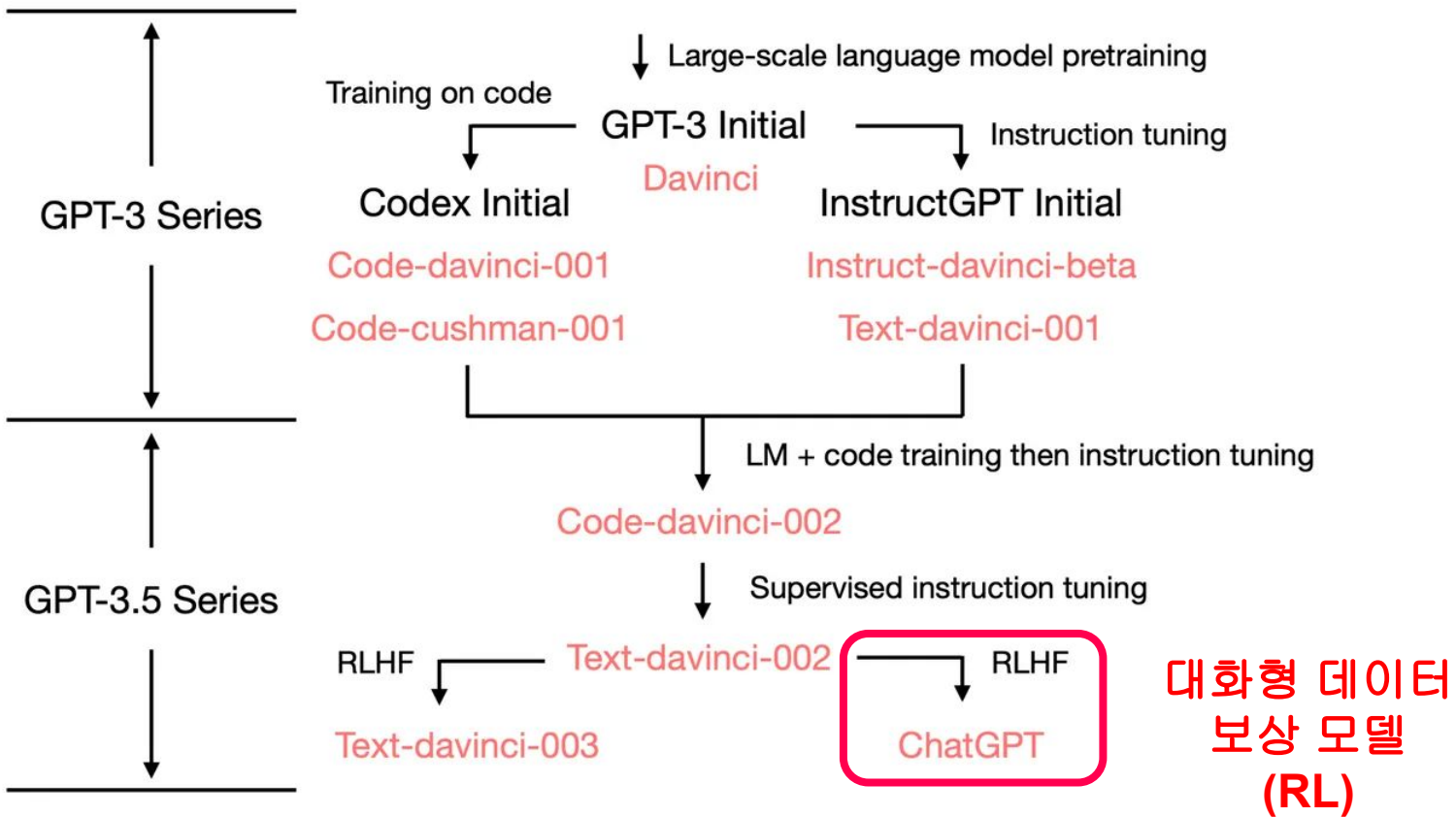
## Instruction 적용 모델

**Q> ChatGPT에 대해 설명해줘**  
A> ChatGPT는 OpenAI가 개발한  
대화형 인공지능 모델입니다. 이  
모델은 다음과 같은 특징들을 가지고  
있습니다:

자연어 처리 능력: ChatGPT는  
사람들이 사용하는 언어를 이해하고,  
이에 응답할 수 있습니다.



# GPTs



다양한 문제를 해결하기 위한 범용 AI 모델

**GPT-3 ➡ GPT 3.5 ➡ RLHF**  
**(Generative Pre-trained Transformer)**

**Transformer**

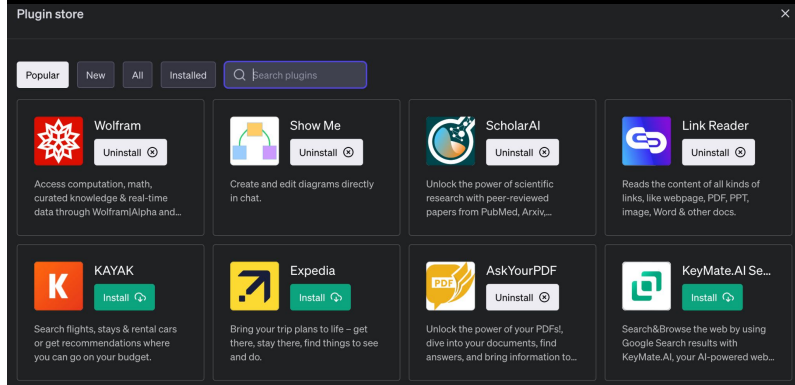
**빅 데이터**

## I 단 10일, LLM의 힘

GPT 3.5을 이용한 ChatGPT가 발표되었습니다.  
단 5일만에 하루 이용자가 100만명이 넘었습니다.

요약, 번역, 생성 등  
다양한 기능을 한번에 처리가 가능해

교육, 마케팅, 콘텐츠 제작 회사들은  
적극적으로 활용 중 입니다.



## | 알파고보다 강력한 파급력

바둑보다는 글을 쓰는 사람들이  
훨씬 많습니다.

또 쉬운 UI/UX로  
많은 사람들이 접근 가능했습니다.

모든 사람들이 사용하는  
채팅으로 사용이 가능했죠.

⚡ GPT-3.5

⚡ GPT-4

ChatGPT PLUS

## Foundation model의 사용

### Foundation Models

**Pre-trained**

자기 지도학습  
빅데이터



**Application**

처리된 데이터  
사용자 피드백

다양한 어플리케이션으로 연결 가능

정보 입력 ➡

LLMs

➡ 생성  
➡ 검색  
➡ 쇼핑  
➡ 인식  
⋮

## | 생산력의 상승

개발자가 chat-gpt를 활용하면  
개발 속도 향상을 체감할 수 있습니다.

물론 생성된 코드를 그대로 사용하진 않습니다.

대화형 UI, UX로 학습, 피드백 등을 이용해  
코드를 개선하거나 사용자가 학습을 할 수 있습니다.

SJ Unity로 테트리스 구현해줘



테트리스를 Unity로 구현하는 것은 재미있는 프로젝트입니다! 테트리스는 비교적 간단한 게임이므로 Unity를 사용하여 구현하는 것은 어렵지 않을 것입니다. 아래는 테트리스를 Unity로 구현하는 간단한 예제입니다. 참고로, 이 예제는 C# 스크립트를 사용합니다.

먼저, 게임 보드를 나타내기 위해 새로운 Unity Sprite를 만들고, 그 위에 테트리스 블록을 배치하는 방식으로 게임을 구현할 것입니다.

csharp

Copy code

```
using UnityEngine;

public class TetrisManager : MonoBehaviour
{
    public GameObject[] tetriminos; // 테트리스 블록 프리팹 배열
    public Transform spawnPoint; // 블록이 생성되는 위치

    private GameObject currentTetrimino; // 현재 떨어지고 있는 블록

    void Start()
    {
        SpawnTetrimino(); // 게임 시작 시 첫 번째 블록 생성
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            MoveTetrimino(Vector3.left); // 왼쪽으로 이동
        }
    }
}
```

## | 학습을 위해 이용하는 방법

직접 코드를 작성하는 것 보다  
그 코드가 구현된 배경을 학습할 수 있습니다.

대화형으로 모델과 이야기하면서  
해당 부분의 구현체 및 예제를 빠르고 쉽게  
학습할 수 있습니다.

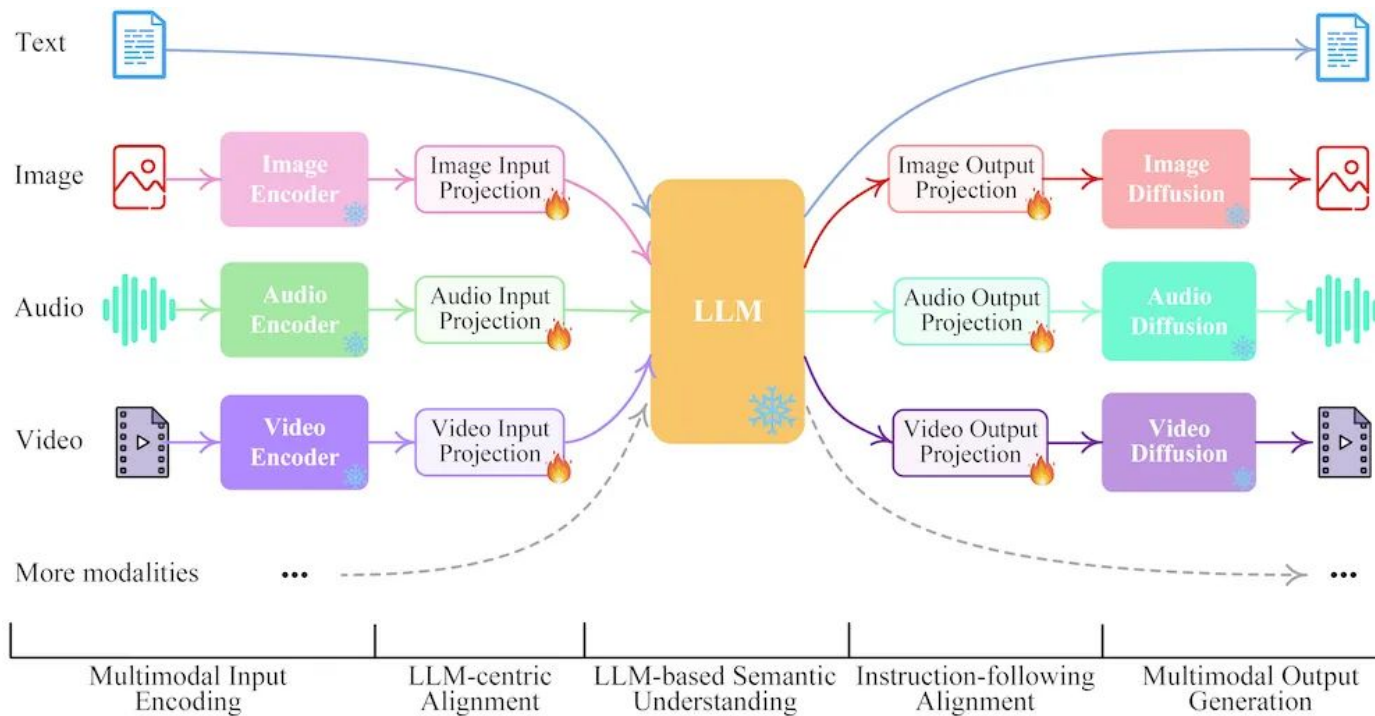
또한, 직접 작성한 코드의 리뷰 / 오류를  
잡아주기도하고, 번역된 글의 어색한 부분을  
수정해주기도합니다.



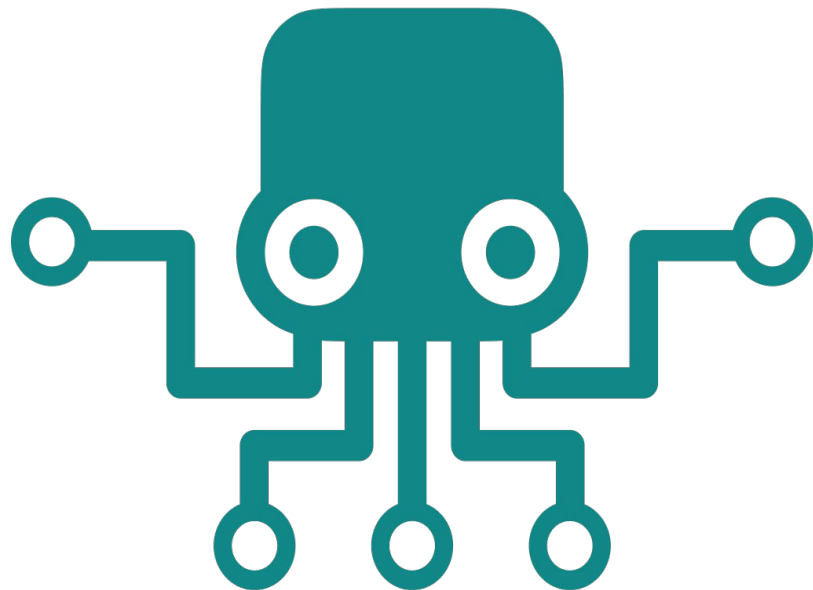




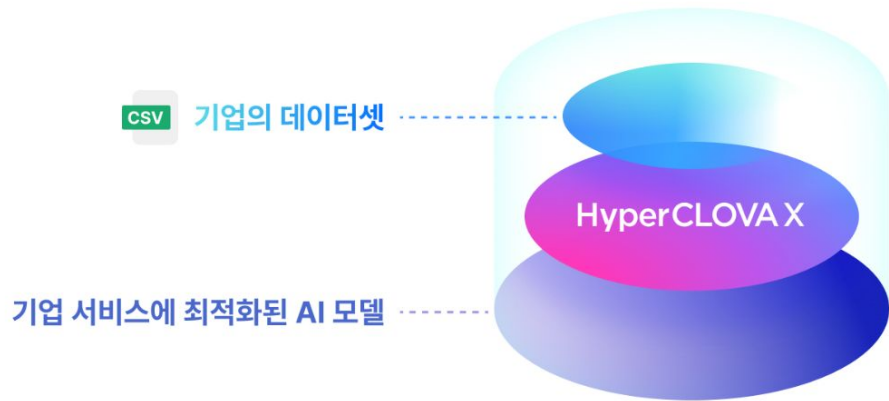
# GPTs & Multimodal



# GPTs & Other Models



# GPTs & Fine-tuning



## 튜닝 비즈니스에 최적화된 모델 만들기

기업이 보유한 데이터셋을 HyperCLOVA와 결합하여 특화 AI 언어 모델을 구축하고, AI 서비스를 개발할 수 있습니다.

# GPTs & Prompt

## **Prompt**

LLMs에서 원하는 출력 결과를 위해 사용자가 입력하는 Input

## **Prompt Engineering**

LLMs의 출력 결과의 품질을 올리기 위한 Input 입력 방법론

# Prompt Engineering