ChatGPT & LLM

한대희 @ 뉴럴웍스랩

daehee@neuralworks.io

010-2101-0255

http://neuralworks.io/

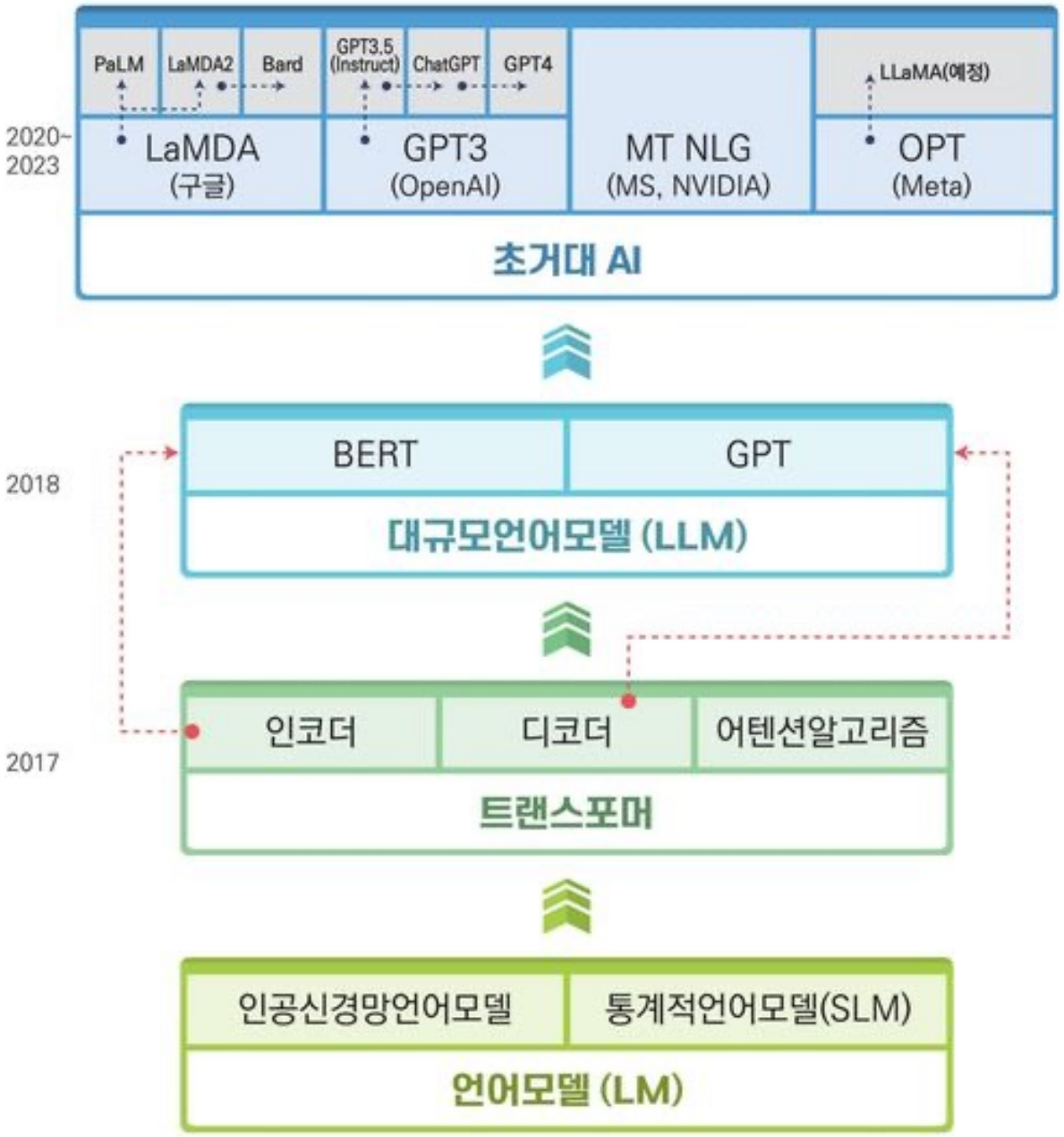
NeuralWorks

GPT란

- Generative Pre-trained Transformer
- 대규모 언어 모델(LLM, Large Language Model)
- **G**enerative(생성, 또는 생성모델): '그럴듯한 가짜'를 만들어낸다는 의미
- **P**re-trained : 전세계 모든 문서를 모두 미리 (unsupervised) 학습 (8백만 문서, 40G(100억개) 단어, 499B토큰, 753Gbytes, A100 GPU 1만개이상 사용 등)
- **T**ransformer : 단어를 입력하면 다음 나올 단어의 확률 계산기
- ChatGPT : 대화형으로 만든 GPT

GTP 관련 기술 히스토리

- Word2Vec
- LSTM, GRU
- Transformer
- BERT
- GPT
- InstructGPT
- ChatGPT
- RLHF(Reinforcement Learning from Human Feedback)


PaLM
LaMDA2
Bard
GPT3.5 (Instruct)
ChatGPT
GPT4
LLaMA(예정)
2020~2023
LaMDA (구글)
GPT3 (OpenAI)
MT NLG (MS, NVIDIA)
OPT (Meta)
초거대 AI
2018
BERT
GPT
대규모언어모델 (LLM)
2017
인코더
디코더
어텐션알고리즘
트랜스포머
인공신경망언어모델
통계적언어모델(SLM)
언어모델 (LM)

GPT의 진화

모델	GPT1	GPT2	GPT3	GPT3.5 (instructGPT)	chatGPT	GPT4
출시일	2018년 7월	2019년2월	2020년 5월	2022년 1월	2022년 11월	2023년내
파라미터수	1.17억개	15억개	1750억개			100조개
Layer / Dimension	12 / 768	48 / 1600	96 / 12288			
신규 기술 특징			Few-shot learning (단순한 큰 언어 모델 + 빅데이터)	RLHF	RLHF 대화 모델	

-RLHF(Re-enforcement Learning from Human Feedback) : 인간에 의한 강화 학습, 인간이 원하지 않는 응답(폭력적/선정적/자해/혐오/차별 관련) 처리
-Few-shot learning : 아주 적은 수의 데이터 만으로 학습 가능하게 하는 알고리즘
-대화모델(Dialogue model) : OpenAI가 선택한 대화 모델 규칙은 Sparrow 대화 모델(2022년 9월)에 DeepMind 가 적용한 규칙 23가지 규칙과 유사, 예를 들면 위협적인 진술 금지 등

GPT-1, GPT-2, GPT-3

GPT-1

- **Training Data:** BookCrawl
- **Parameters:** 117M
- **Layers:** 12
- **Dimensions:** 768

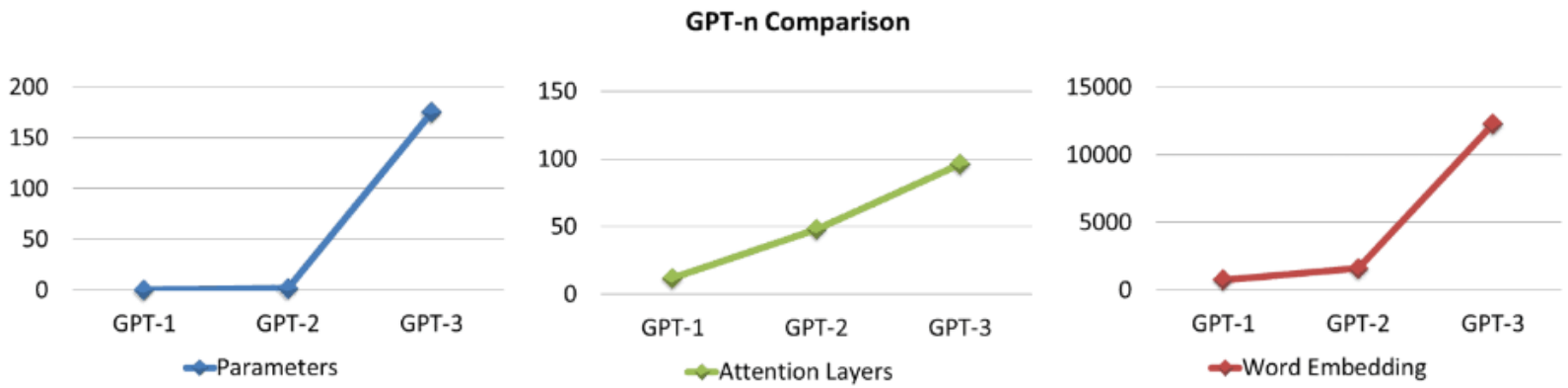
GPT-2

- **Training Data:** WebText
- **Parameters:** 1.5B
- **Layers:** 48
- **Dimensions:** 1600

GPT-3

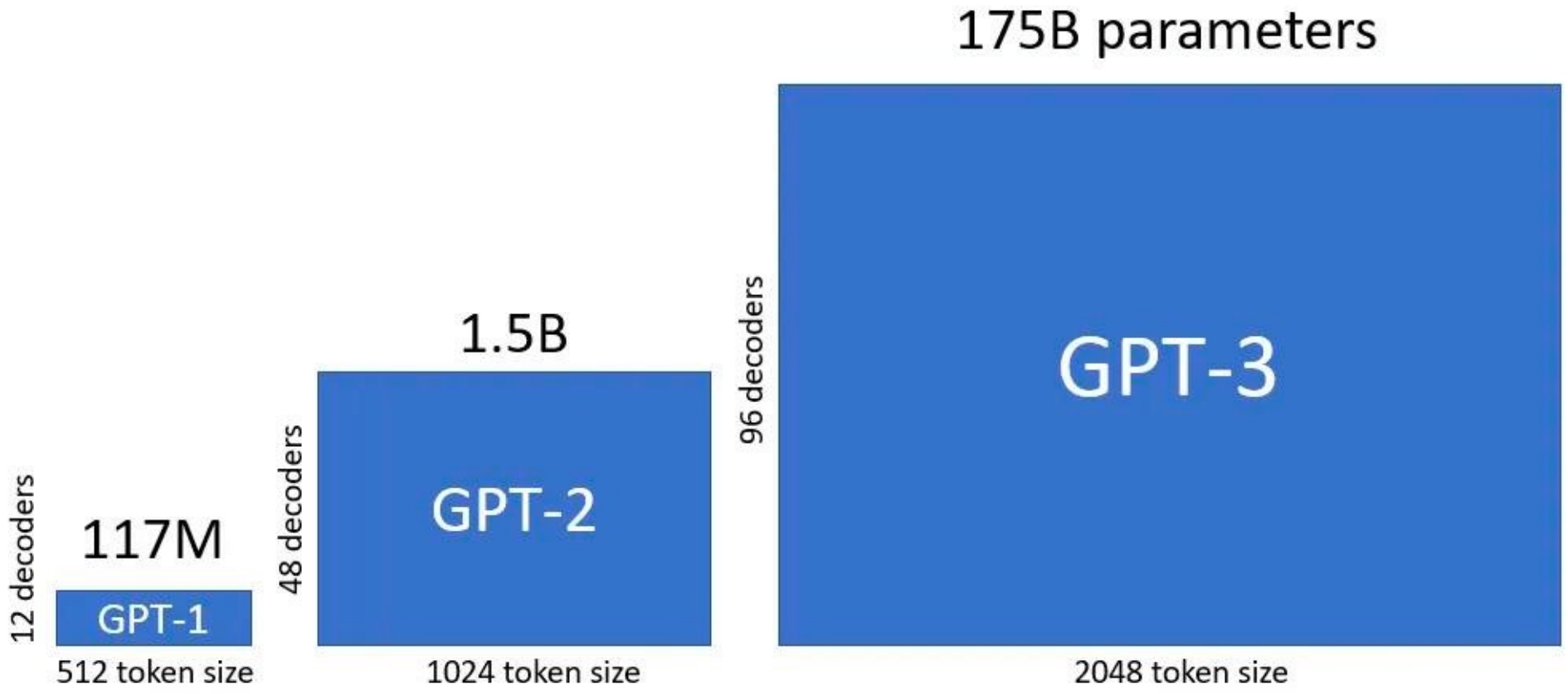
- **Training Data:** CommonCrawl
- **Parameters:** 175B
- **Layers:** 96
- **Dimensions:** 12288

GPT-n Timeline



GPT의 파라미터 개수, 사람 뇌의 비교

- 사람의 뉴론 개수: 1000억, 100B,
- 연결 시냅스 - 1조~10조. 1,000B = 1T, 1T ~ 10T



GPT-1, GPT-2, GPT-3

Model	Launch Year	Training Data	Training Parameters	Attention Layers	Word Embedding	Attention Heads
GPT-1	2018	7000 Books ~5GB	117M	12	768	12
GPT-2	2019	8 million documents ~40GB	1.5B	48	1600	48
GPT-3	2020	Multiple Source ~45TB	175B	96	12288	96

GPT-3 학습 데이터 규모
-총 499B토큰

GPT-3 Training Data

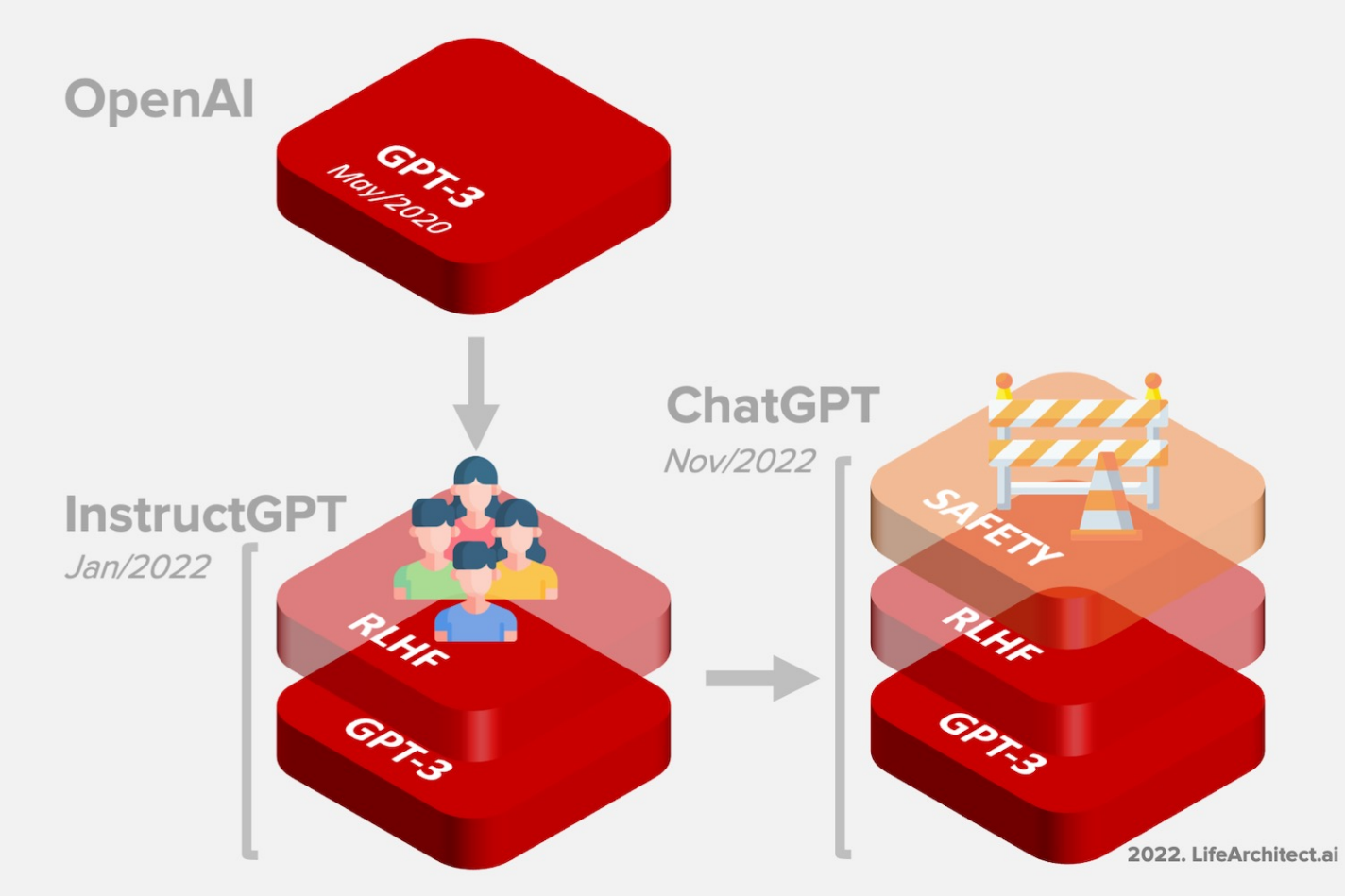
Dataset	# Tokens	Weight in Training Mix
Common Crawl	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Word vector의 크기

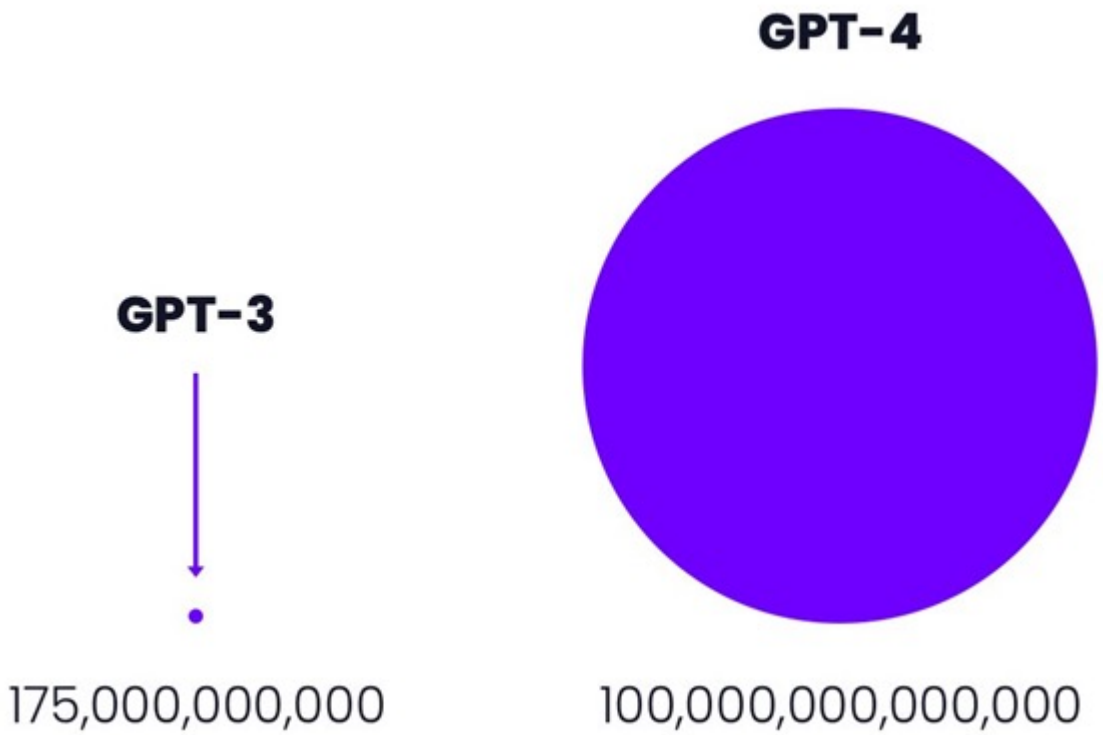
- BERT-base, they have 768 dimensions

1. GPT-3 125M: 768 dimensions
2. GPT-3 350M: 1024 dimensions
3. GPT-3 1.3B: 2048 dimensions
4. GPT-3 13B: 4096 dimensions

GPT3, GPT3.5 그리고 chatGPT



GPT-4?



참고문헌

- 전이학습(Transfer Learning)이란? https://dacon.io/forum/405988
- 자연어 처리 – Transfomer, Bert, GPT-3 https://heave.tistory.com/73
- https://junsik-hwang.tistory.com/61 자연어처리 - 수집 정제 tokenization
- 자연어와 트랜스포머, BERT, GPThttps://blog.testworks.co.kr/natural-language-and-transformer-bert-gpt/ 자연어와 트랜스포머, BERT, GPT
- https://dreamgonfly.github.io/blog/gan-explained/ 쉽게 씌어진 GAN
- https://yseon99.tistory.com/127 사전학습모델
- https://heytech.tistory.com/341 언어모델(Language Model)의 개념 및 특징
- https://lifearchitect.ai/chatgpt/#chatgpt GPT-3.5 + ChatGPT: An illustrated overview
- https://www.kdnuggets.com/2020/06/gpt-3-deep-learning-nlp.html GPT-3, a giant step for Deep Learning and NLP?
- https://www.mk.co.kr/news/it/10619669 "나는 혁신기술이지만 인간 대체하지 않아"
- https://littlefoxdiary.tistory.com/44 [논문리뷰] GPT3 - Language Models are Few-Shot Learners
- https://littlefoxdiary.tistory.com/62 [논문리뷰] Small Language Models Are Also Few-Shot Learners
- https://jiho-ml.com/chatgpt-intro/ ChatGPT에 화들짝 놀라신 분 들어오세요 - 원리 편
- https://soundprovider.tistory.com/entry/GPT3-%EC%A3%BC%EC%9A%94-%EB%82%B4%EC%9A%A9-%EC%A0%95%EB%A6%AC [GPT3] 주요 내용 정리

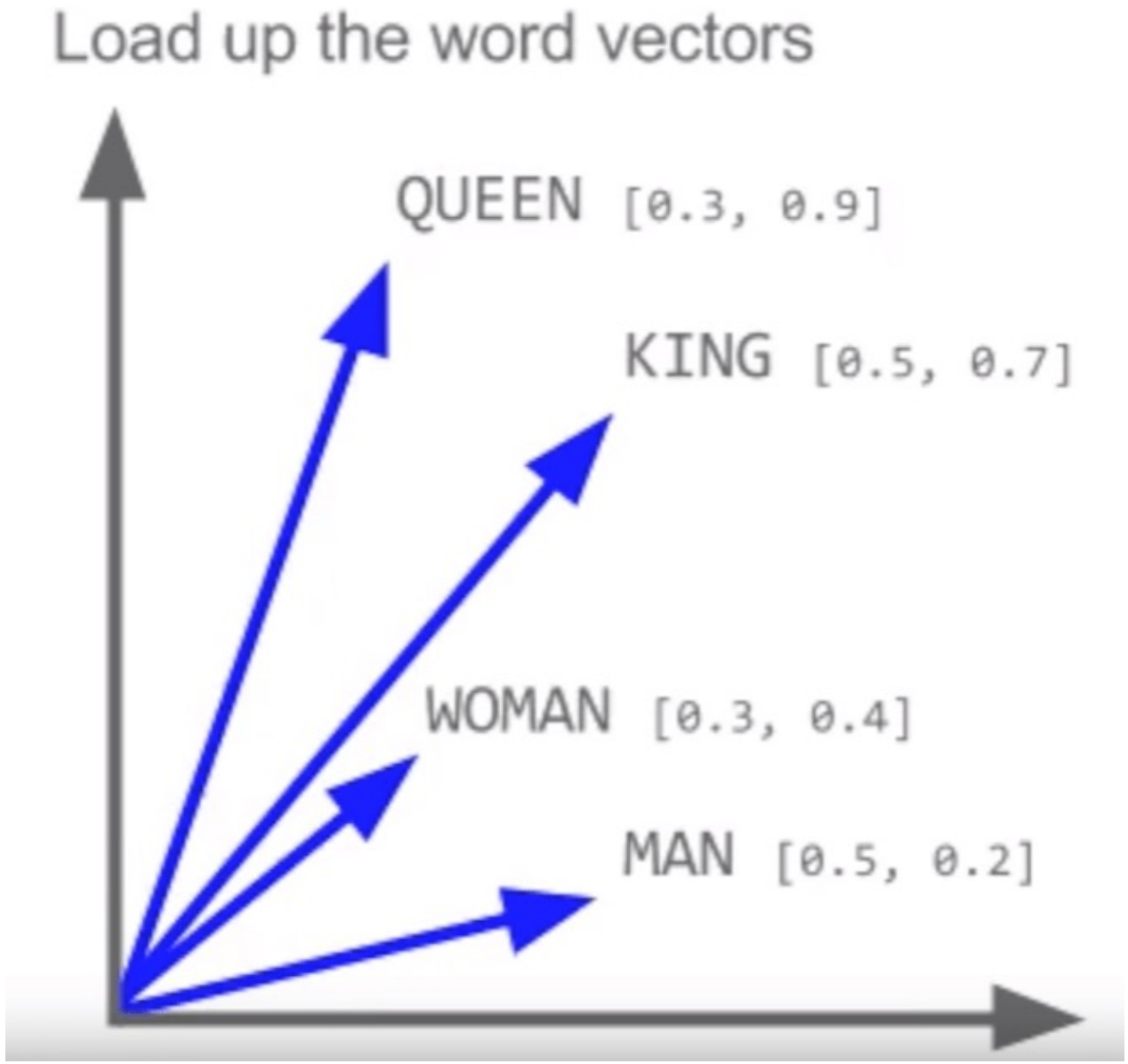
Few-shot learning 원리와 특징

- 참고문헌들
- https://blog.naver.com/PostView.naver?blogId=dh0985&logNo=222364043528 [AI 스터디] GPT-3 본격 분석 - BERT / 초거대언어모델 / 사전학습 / 미세조정 / 퓨샷러닝 / 메타러닝 / in-context learning / 셀프어텐션

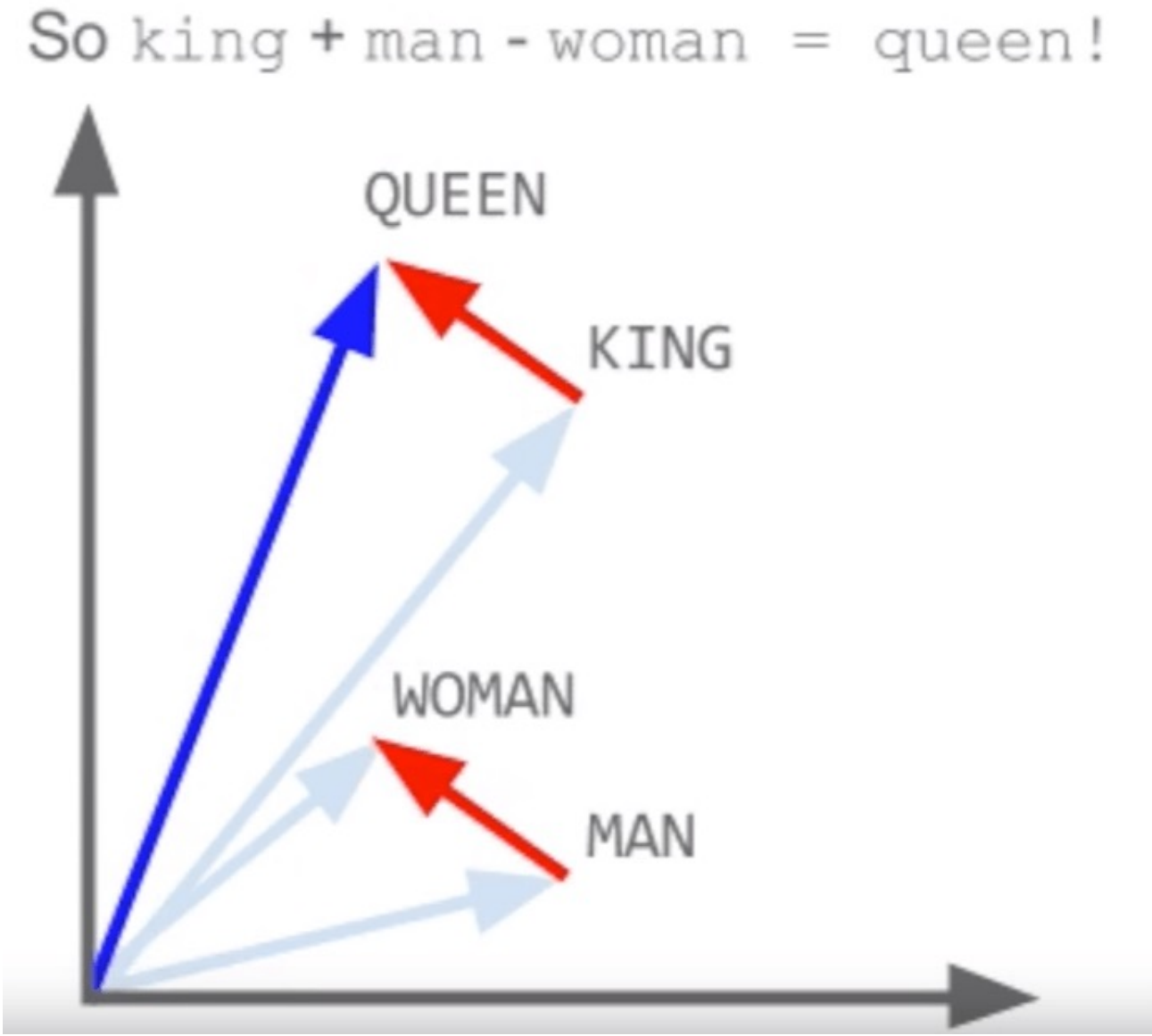
Word2Vec(Embedding)
RNN
Seq-to-Seq
Attention
Transformer
BERT
GPT
ChatGPT

Word2Vec – Each word is a vector.

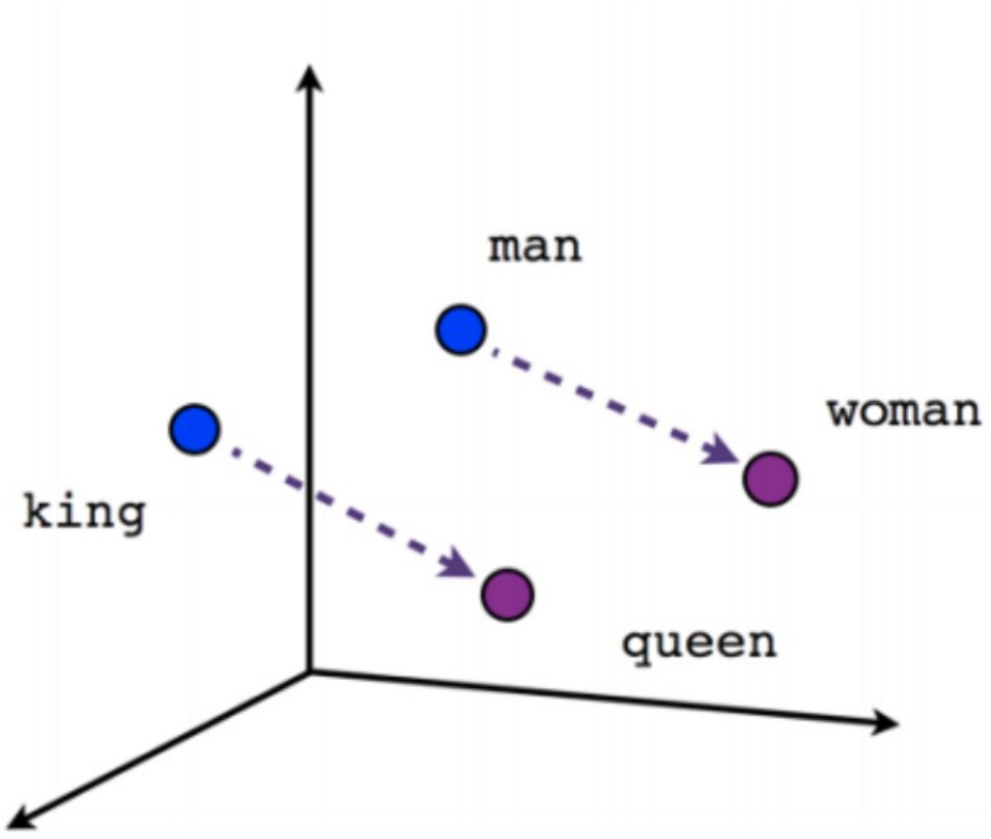
http://www.lifestyletrading101.com/word2vec-deep-learning/



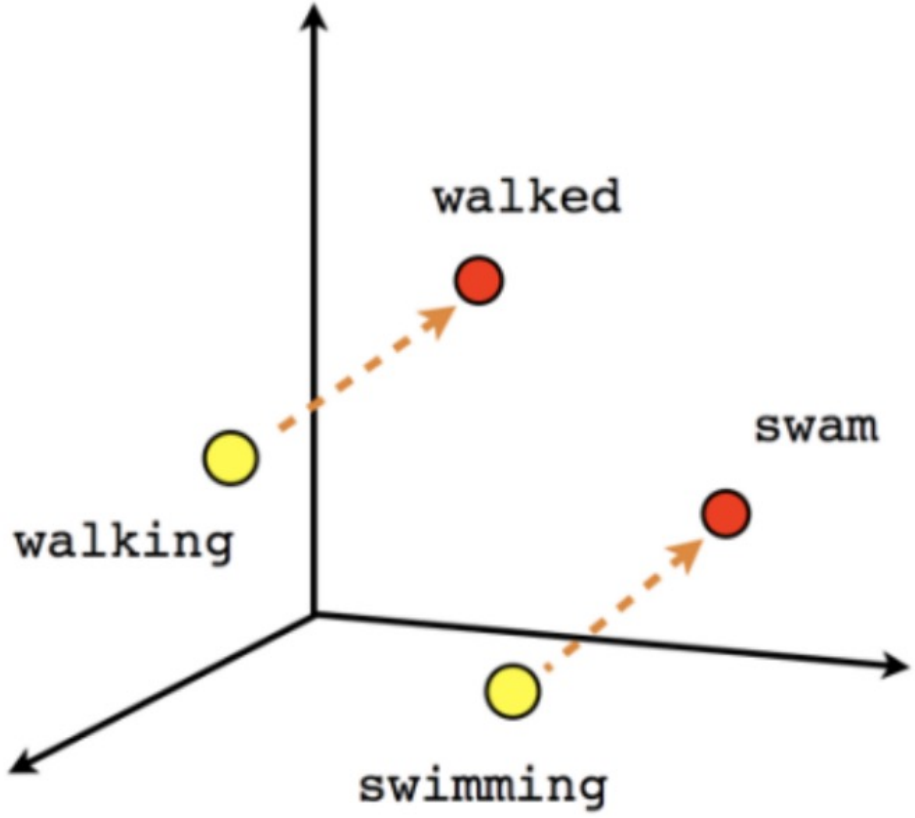
Word2Vec: King + (man – woman) = queen



Word2vec



Male-Female



Verb tense

does BERT use word2vec internally?

- No, BERT (Bidirectional Encoder Representations from Transformers) does not use Word2Vec internally. **BERT and Word2Vec are different** techniques for learning word representations.

- BERT, on the other hand, is a deep learning model based on the **Transformer** architecture. It is **pre-trained** on a large corpus of text using masked language modeling and **next sentence prediction** tasks. BERT learns contextualized word representations, meaning that it captures information about the context in which words appear, resulting in **different representations for the same word in different contexts**.

Unigram 언어 모델

- 가장 단순 (단어주머니(bag of words) 모델)
- 각각의 단어가 출현할 확률이 독립이라고 가정하고 모두 따로따로 계산
- $P_{uni}(w1,w2,w3,\cdots)=P(w1)P(w2)P(w3)\cdots$

n-gram 언어 모델

- 이전 단어를 고려하여 다음 단어의 확률을 계산
- n-gram 모델에서는 다음 단어를 예측하기 위해 이전의 n−1개의 단어를 활용
- n=2인 경우,
- $P_2(w1,w2,w3,\cdots)=P(w1)P(w2|w1)P(w3|w2)\cdots$
- 3-gram 모델에서는 ab?의 빈도를 세고, 그 중 abc의 빈도를 셈으로써 ab 다음에 c가 등장할 확률
- 단점 : 모델 크기와 빈도 조회 시간의 문제, 학습 말뭉치에서 등장하지 않은 단어열에 대해서는 확률 계산이 불가능 ->Smoothing 기법이 등장

Kneser-ney 언어 모델

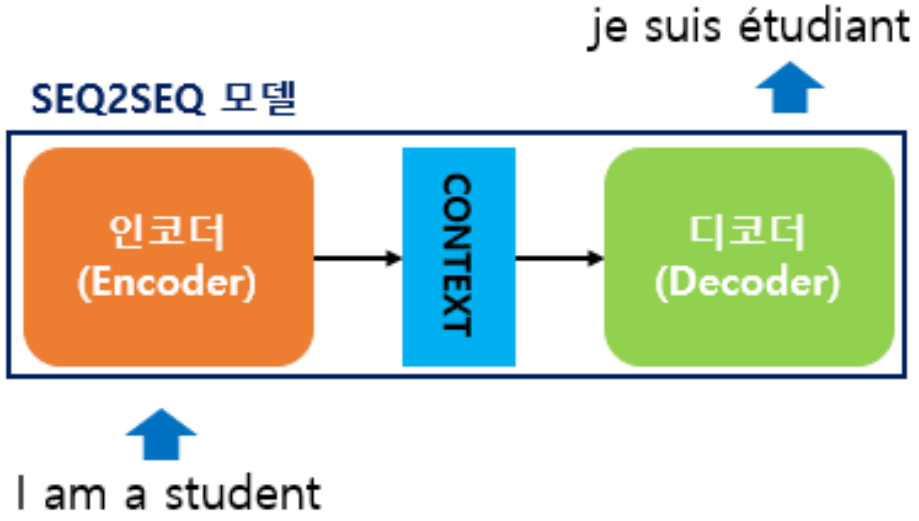
- n-gram 모델의 여러 smoothing 기법 중에서 가장 높은 성능
- ab 다음에 c가 등장할 확률을 계산할 때, b다음에 c가 등장할 확률도 함께 고려, 또 b 다음 c가 등장할 확률을 계산할 때도 그냥 c가 등장할 확률도 고려

RNN 계열 언어 모델

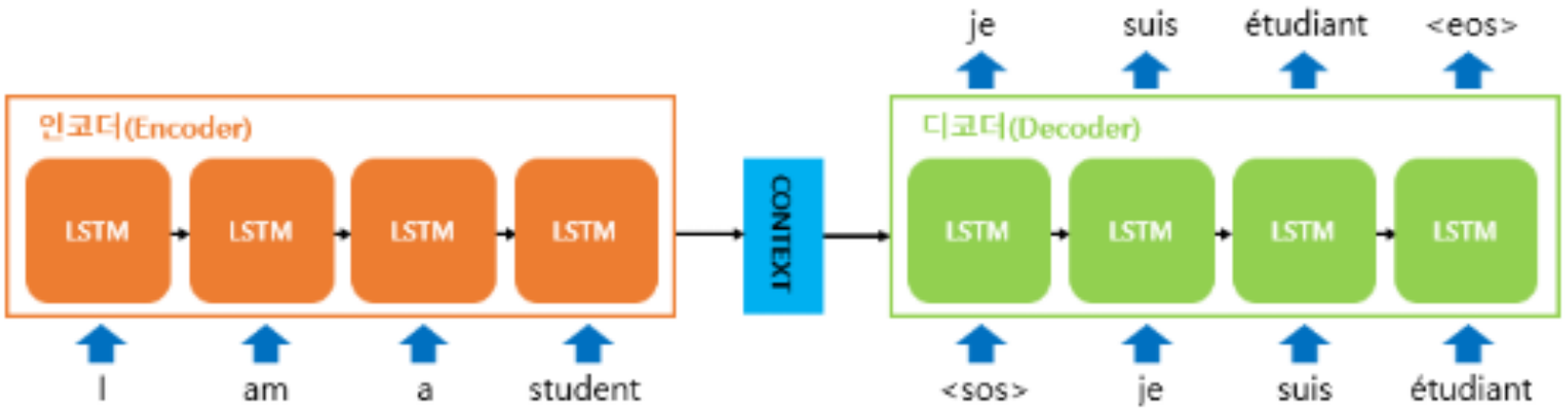
- RNN은 이전 상태값과 현재 입력을 조합하여 새로운 상태값을 계산해 내는 신경망
- RNN(f) 언어 모델은 언어의 생성의 패러다임을 암기에서 계산으로 변화
- $h_t = f(x_t; h_{t-1})$
- n-gram 모델과 같이 빈도 기반 모델은 학습용 말뭉치에서 등장하는 패턴을 외워서 이를 그저 적용하는 것에 불과했다면, RNN은 암기하는 대신에 그저 f를 반복 계산하는 것을 통해 특정 단어나 문장이 등장할 확률을 계산
- 어휘에 임의의 벡터를 대응시키는 걸 임베딩(embedding)이라 함
- RNN의 출력 또한 마찬가지로 다음에 등장할 단어의 확률이 아니라 h_t 라는 임의의 벡터

시퀀스-투-시퀀스(seq2seq) 모델

- 한 문장(시퀀스)을 다른 문장(시퀀스)으로 변환하는 모델
- 인코더(Encoder)와 디코더(Decoder)로 구성
- 번역 seq2seq모델 예시 (영어 -> 불어)



- 인코더나 디코더는 RNN으로 구성(LSTM 등)



RNN + Attention 모델

- Attention :
  - 기계번역 연구
  - 짧은 문장에 대해서는 번역을 성공적으로 수행할 수 있었으나 문장이 길어질 수록 정확도가 크게 떨어지는 문제가 발생
  - 디코더에서 문장을 생성할 때 인코더의 최종 벡터 hh 말고도 중간 벡터들도 모두 살펴볼 수 있는 구조 [Bahdanau (2014)]
  - 모델이 필요한 부분을 다시 주목(attend)해서 그 부분에서 정보를 꺼내오는 메커니즘을 Attention
- Attention은 꼭 인코더-디코더 구조가 아니더라도 일반 RNN 모델에도 적용 가능

Transformer 모델

- Transformer 는 RNN + Attention 모델에서 RNN을 빼버린 모델
- Self-attention : RNN + Attention에서는 각 단어를 RNN에 통과시켜서 h를 얻은 다음, 현재의 hn값을 이용해 주목해야할 과거의 hh 값들을 찾아내고, 최종적으로 이 hh들을 조합하여 다음 단어를 예측해냅니다. 여기서 다음 단어를 예측하기 위해 과거의 hh 중 어떤 것에 주목할 지가 중요한 것이지, RNN을 통과시키는 게 중요한게 아니라는 거죠. 그래서 아예 RNN구조를 빼버리고, hn이 스스로 h 중 주목해야할 대상들을 찾아내는 것에만 집중
- Transformer 구조에서 인코더 부분만 따로 떼어서 사용하는 게 요즘 널리 쓰이는 BERT모델
- 디코더 부분만 따로 떼어서 사용하는 게 GPT 모델

GPT모델

- 현재 시점에서 특정 문장이 등장할 확률을 계산하거나, 주어진 단어들 다음에 등장할 단어를 예측하는 작업을 아주 뛰어나게 수행
- GPT-1에서부터 시작하여 현재는 GPT-3까지 등장
- 모델의 크기가 어마어마하게 커져서 일반 컴퓨터에서는 돌리기도 어려워졌다는 문제가 있긴 하지만 매우 강력한 성능
- GPT-3의 경우 모델의 크기가 매우 커지면 단순히 이전 단어들로부터 다음 단어를 예측하는 식으로 학습을 시켰음에도 그 안에서 놀라운 능력들이 생겨난다는 점
- 르쳐준적이 없는데도 모델이 스스로 번역을 한다든지, 몇개의 예제와 함께 문제를 던져주면 예제를 파악해서 문제를 풀어준다든지, 수학 계산을 해주기도 합니다. 이 모든게 그저 이전 단어들이 주어졌을 때 다음 단어를 예측하는 과정

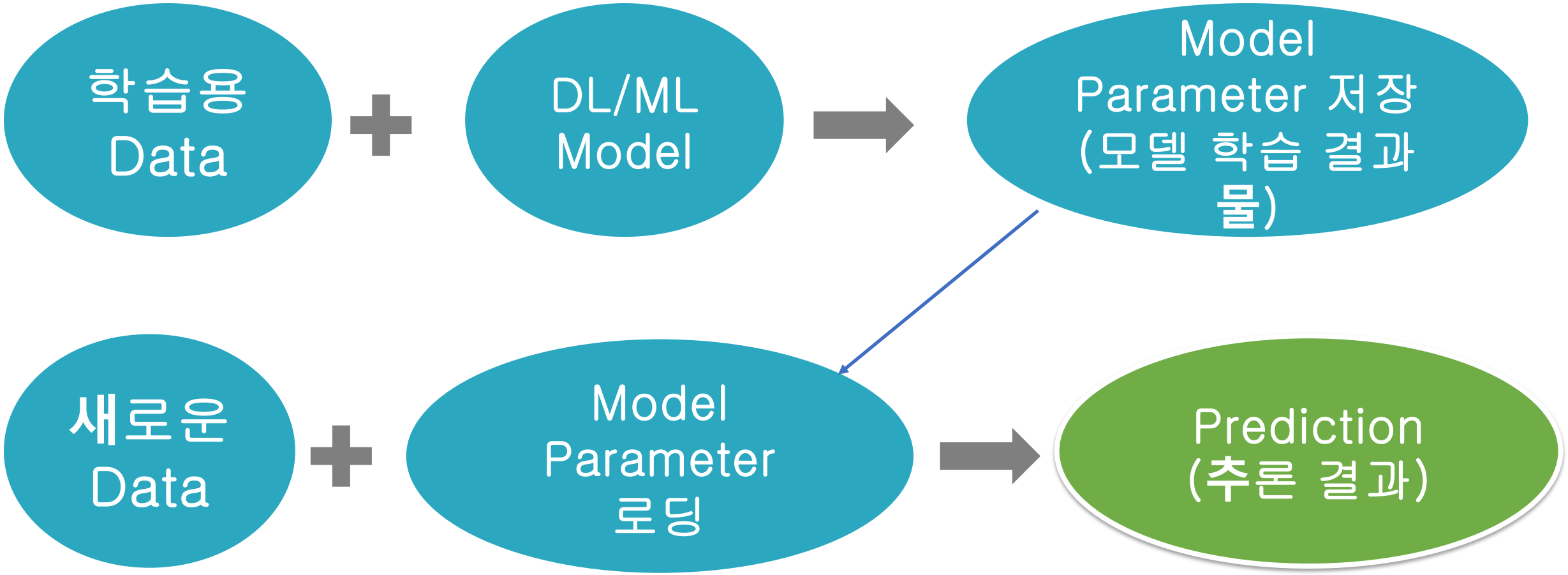
ChatGPT Fine-tuning

나만의 데이터로 나만의 모델을 만들기

용어

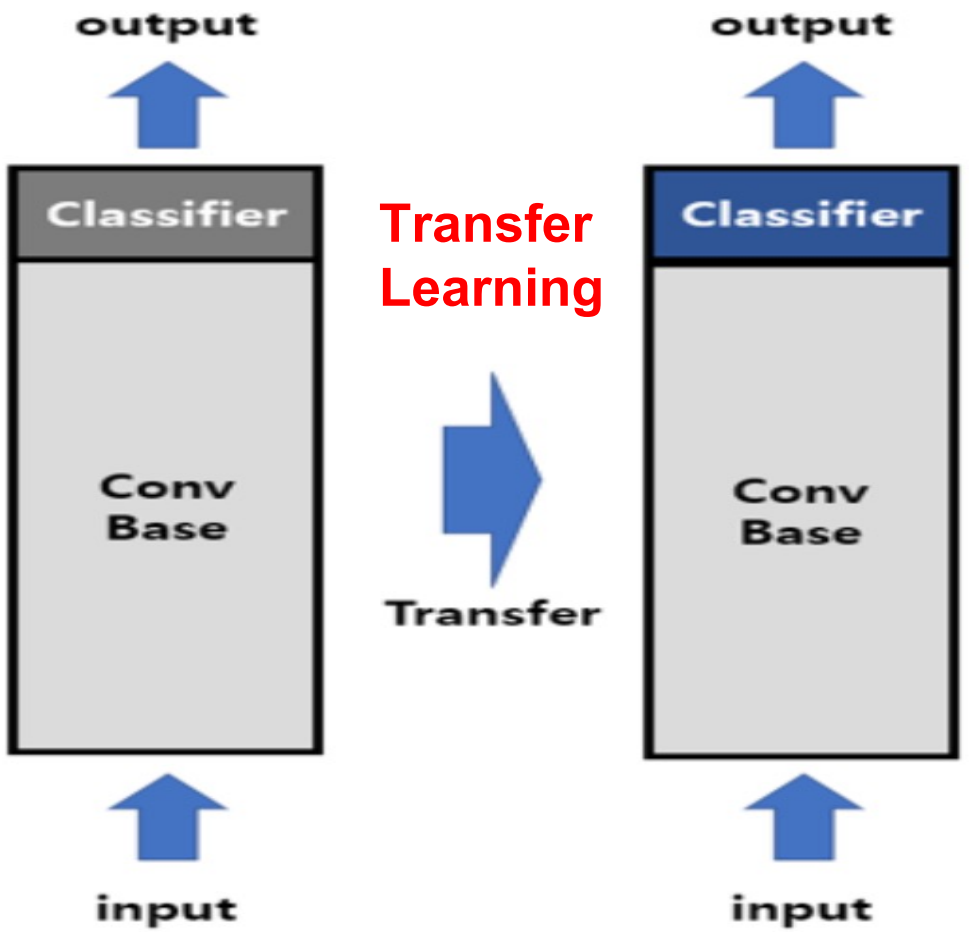
- **Transfer Learning : 뇌 이식. 뇌의 끝단만 조정**
- **Fine-tuning: 미세 조정 (세뇌 ?)**
- **In-Context Learning (Zero shot learning)**

학습(Training)과 추론(inference)



Transfer Learning (전이 학습)

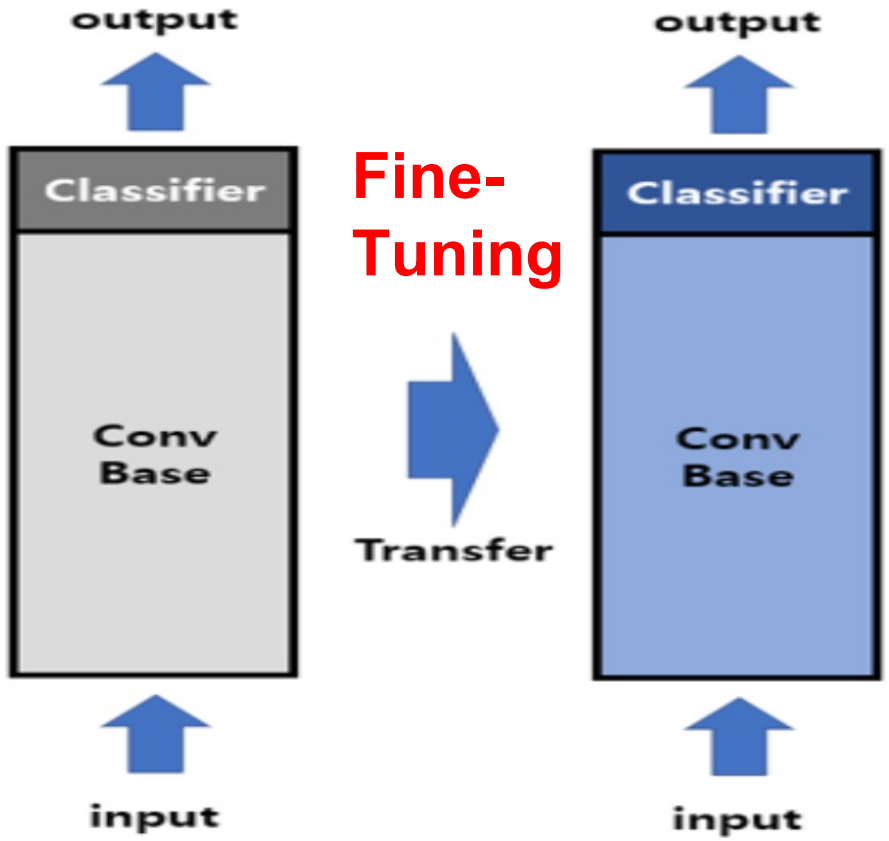
- 기존 데이터를 학습한 기존 뇌를 사용
- 새 데이터로 추가 학습하면서 **기존 뇌의 일부(출력단)만 수정**



이미지 출처

Fine-tuning

- 기존 데이터를 학습한 기존 뇌를 사용
- 새 데이터로 추가 학습하면서 기존 뇌의 전체를 미세조정함
- 예) 서양인들 얼굴 10억개를 학습 모델에 한국 얼굴 10만개를 추가학습



GPT3.5, GPT4 Price(Query, 추론)

- 토큰 수: 단어개수에 비례
- Prompt: 질문,지시
- Completion: 응답

GPT 3.5

Model	Usage
gpt-3.5-turbo	$0.002 / 1K tokens

GPT 4

Model	Prompt	Completion
8K context	$0.03 / 1K tokens	$0.06 / 1K tokens
32K context	$0.06 / 1K tokens	$0.12 / 1K tokens

ChatGPT 커스텀 학습/훈련, fine-tuning 비용

- ChatGPT 3.0: training $ 0.03, inference $ 0.12

Create your own custom models by fine-tuning our base models with your training data. Once you fine-tune a model, you'll be billed only for the tokens you use in requests to that model.

Learn more about fine-tuning ↗

Model	Training	Usage
Ada	$0.0004 / 1K tokens	$0.0016 / 1K tokens
Babbage	$0.0006 / 1K tokens	$0.0024 / 1K tokens
Curie	$0.0030 / 1K tokens	$0.0120 / 1K tokens
Davinci	$0.0300 / 1K tokens	$0.1200 / 1K tokens

커스텀 학습/훈련, fine-tuning 방법

- 커스텀 학습 - https://platform.openai.com/docs/guides/fine-tuning

1. 학습용 txt 파일로 jsonl 파일을 만듭니다. neural_studio_guide_1.txt --> neural_studio_guide_1.jsonl
2. openai tools fine_tunes.prepare_data -f neural_studio_guide_1.jsonl. --> neural_studio_guide_1_prepared.jsonl
3. openai api fine_tunes.create -t neural_studio_guide_1_prepared.jsonl -m davinci

   --> fine-tuning ID를 부여받음 ex) ft-cA5TB40jReezSXO46LNALXWg
4. openai api fine_tunes.get -i ft-cA5TB40jReezSXO46LNALXWg

Fine-tuning 진행 상황 확인하기

$ openai api fine_tunes.follow -i ft-cA5TB40jReezSXO46LNALXWg

[2023-03-14 04:46:19] Created fine-tune: ft-cA5TB40jReezSXO46LNALXWg
[2023-03-14 05:02:31] Fine-tune costs $0.72
[2023-03-14 05:02:31] Fine-tune enqueued. Queue number: 0
[2023-03-14 05:02:35] Fine-tune started
[2023-03-14 05:05:20] Completed epoch 1/4
[2023-03-14 05:05:38] Completed epoch 2/4
[2023-03-14 05:05:55] Completed epoch 3/4
[2023-03-14 05:06:13] Completed epoch 4/4
[2023-03-14 05:06:50] Uploaded model: **davinci:ft-neuralworkslab-2023-03-13-20-06-49**
[2023-03-14 05:06:51] Uploaded result file: file-vfvq9FENGjtZ3xYo5QVw2ruB
[2023-03-14 05:06:51] Fine-tune succeeded

Job complete! Status: succeeded

Try out your fine-tuned model:

Fine-tuning(학습)된 모델 사용하기(추론)

- $ openai api completions.create -m davinci:ft-neuralworkslab-2023-03-13-20-06-49 -p <YOUR_PROMPT>
-

감사합니다!