AI Application Specialist

대규모 언어모델 이해 및 파인튜닝

2026

사내교수 박영진 / 첨단기술아카데미

SEU
Samsung Electronics University

수업의 목표

“ LLM 의 기본 구조를 이해하고

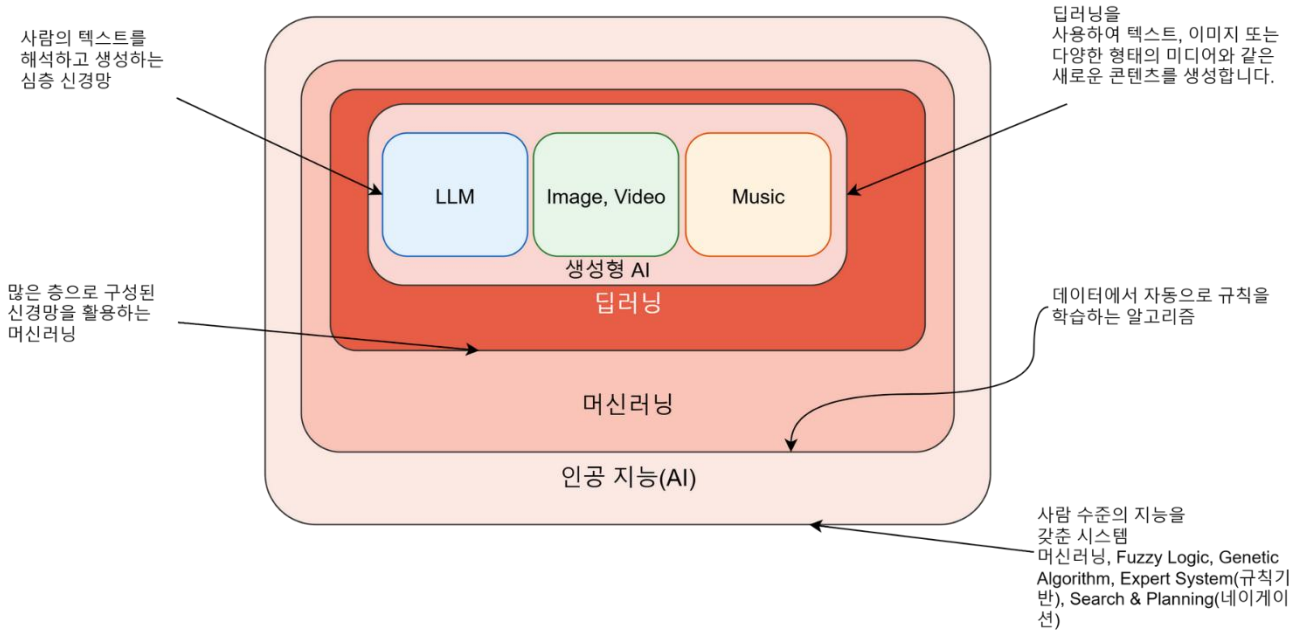
이해한 것은 코드로 실습해서

LLM을 잘 사용하는 것”

AI 모델의 역사

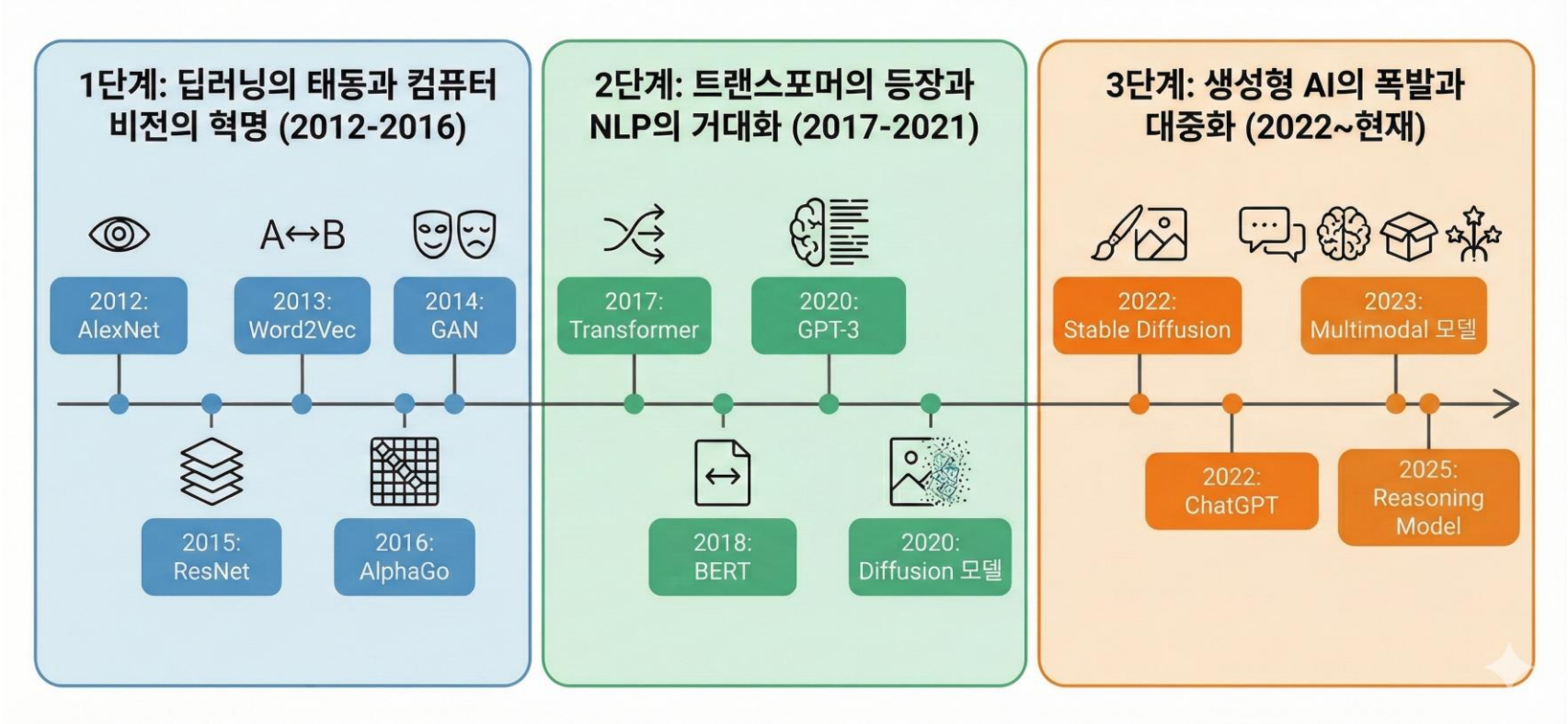
AI 모델 구분

- 머신러닝 -> 딥러닝 -> 생성형 AI



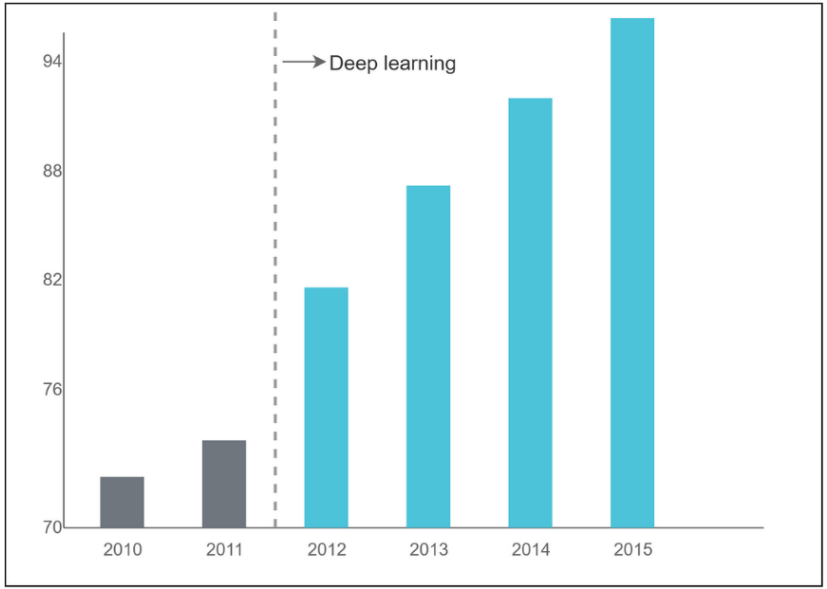
Deep Learning 모델의 역사

- Deep Learning 모델의 역사

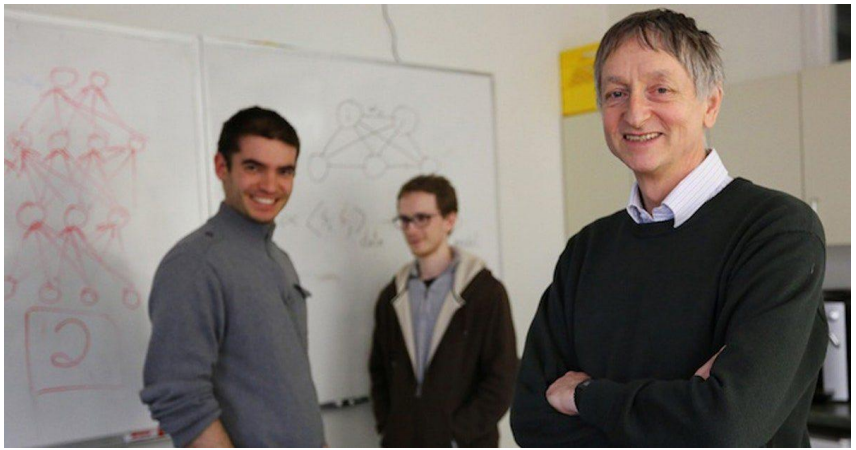


AlexNet

- ImageNet 대회에서 압도적으로 1등
  - Deep Leaning의 시작, GPU의 사용
  - Layer을 깊게 쌓기 위한 경쟁 돌입



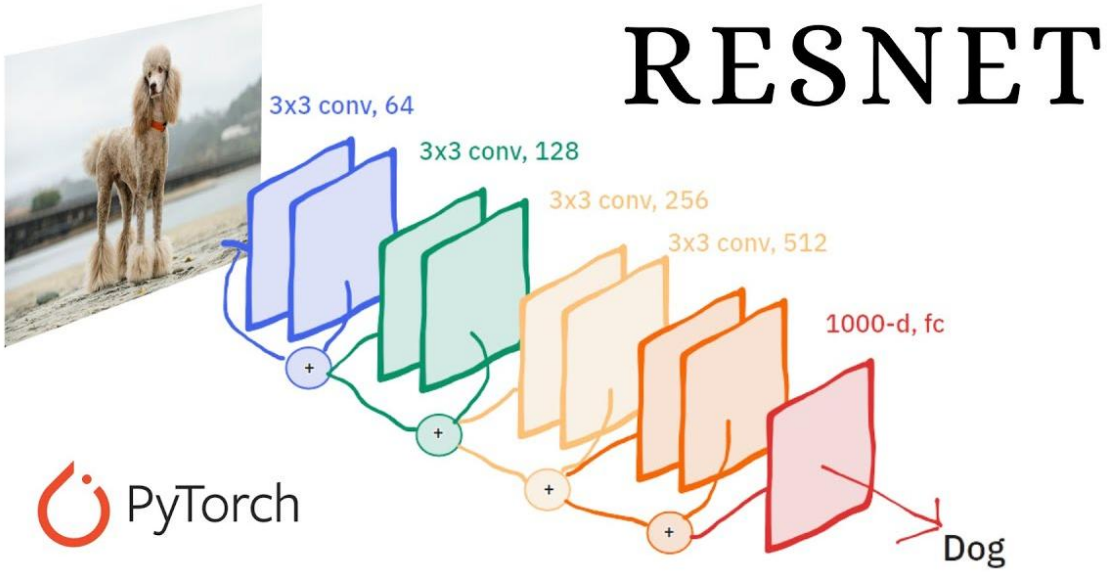
[AI 이야기] 인간 VS 인공지능 (3)딥러닝의 시대를 연 알렉스넷(AlexNet)



오른쪽부터 제프리 힌든, 알렉스 키르제브스키, 일리야 수츠케버

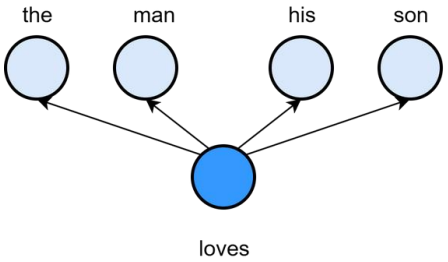
ResNet

- 신경망의 깊이 한계를 깨부셨다
  - Skip Connection 도입
  - Layer을 깊게 쌓기 위한 한계 돌파

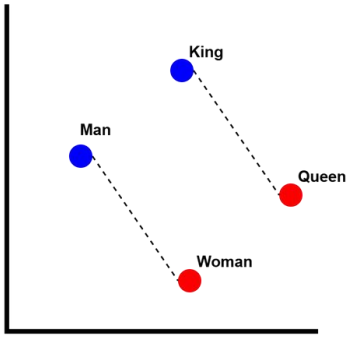


Word2Vec

- 단어의 의미를 백터 공간에 보존하여 단어 간의 연산을 가능하게 함(추론)



word2vec(2013)



단어의 의미를 백터 공간에 맵핑하여 계산할 수 있게 됨

토마시 미콜로프

"This infuriated a local linguist who declared my ideas to be a total nonsense"

Alphago

- AI의 미래를 보여주는 사례

Alphago

78수
알파고의 몬테카를로 트리 탐색 알고리즘이 예상치 못한 상황에서 어떻게 오류를 일으키는지 보여준 역사적인 사례

AlphaGo Zero

AlphaGo
ZERO

- 스스로 깨우침
- 독창적이고 효율적인 수법
- 간결한 하드웨어: TPU 4대로 3일간 학습

AlphaZero

- Alphago를 다른 게임으로 확대
- 짧은 학습 시간으로 압도함
- 규칙만으로 학습하는 범용성
- 공격적이고 창의적인 스타일

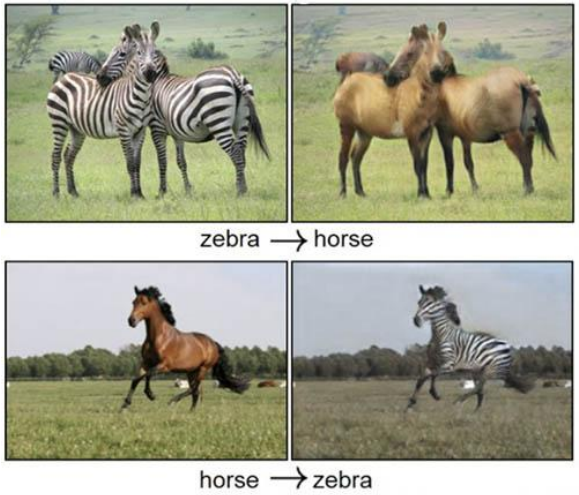
Transformers

- “Attention is all you need”
  - 모든 저자들이 창업하여 수천에서 수조원의 스타트업 창업

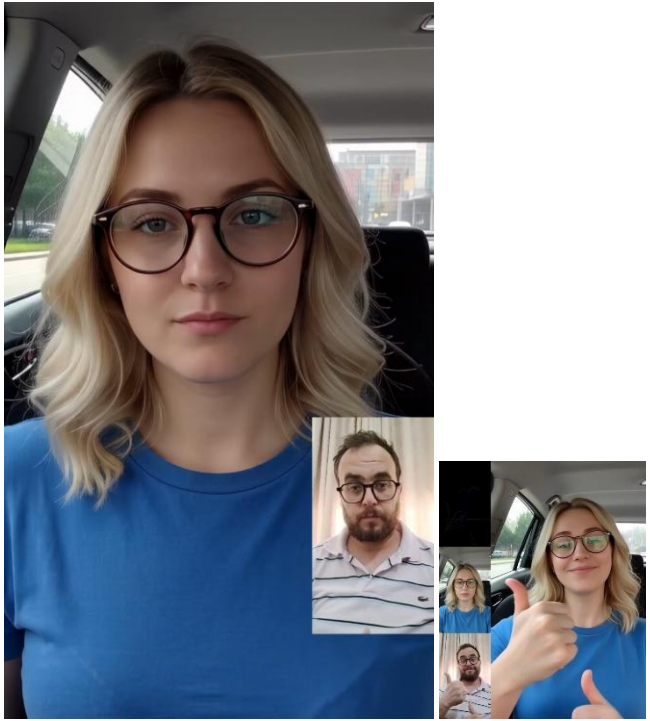
GAN과 Diffusion

- Multimodal 생성의 시대

I,Robot(2004)



CycleGAN(2017)

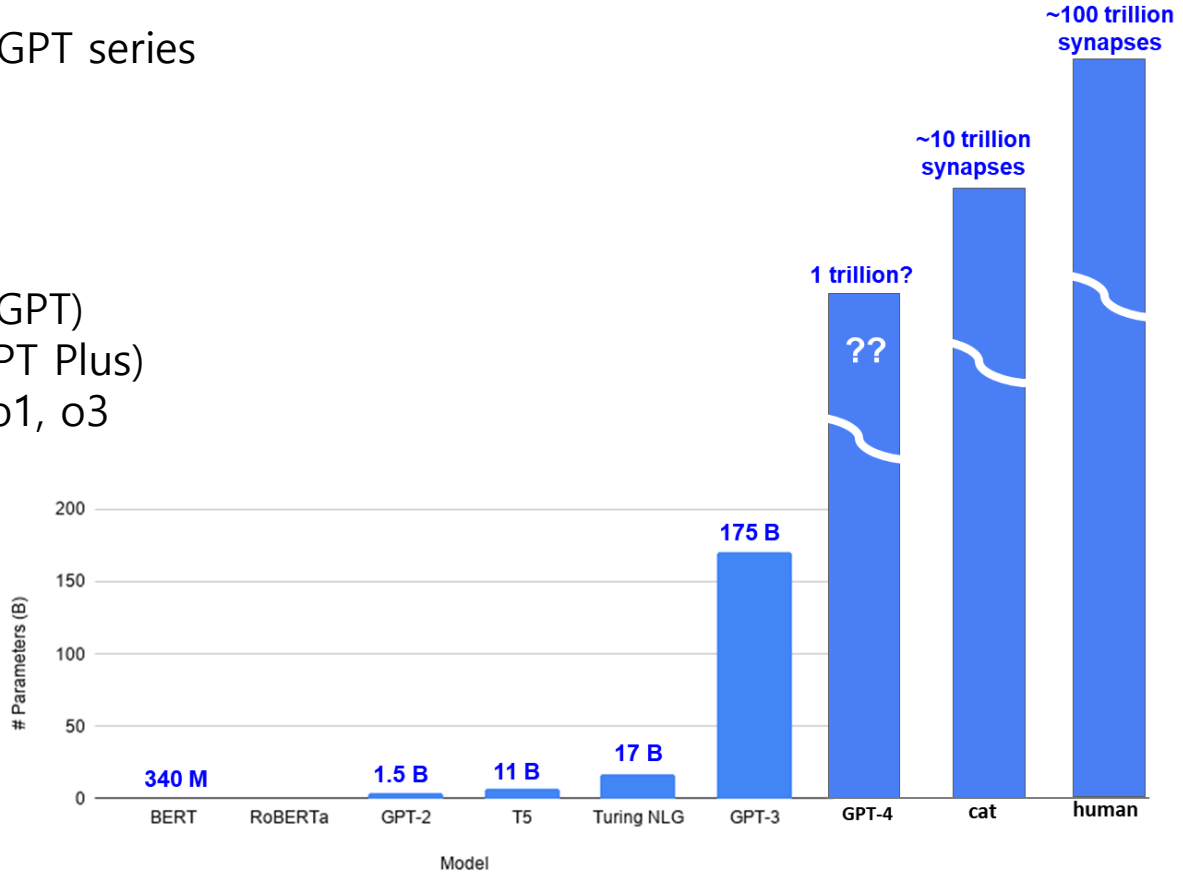


비디오 생성

GAN이란? - 생성적 대립 신경망 설명 - AWS

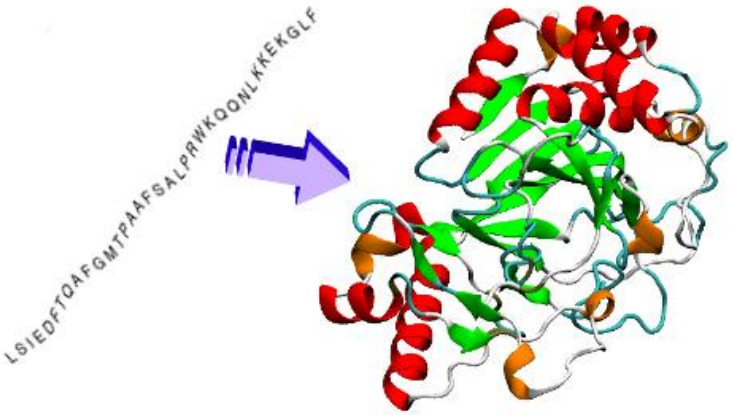
GPT: 규모의 시대

- Comparisons between GPT series
  - 2018 GPT-1
  - 2019 GPT-2
  - 2020 GPT-3
  - 2022 GPT-3.5 (ChatGPT)
  - 2023 GPT-4 (ChatGPT Plus)
  - 2024 GPT-4o, GPT o1, o3
  - 2025 GPT-4.5
  - 2025 GPT-5

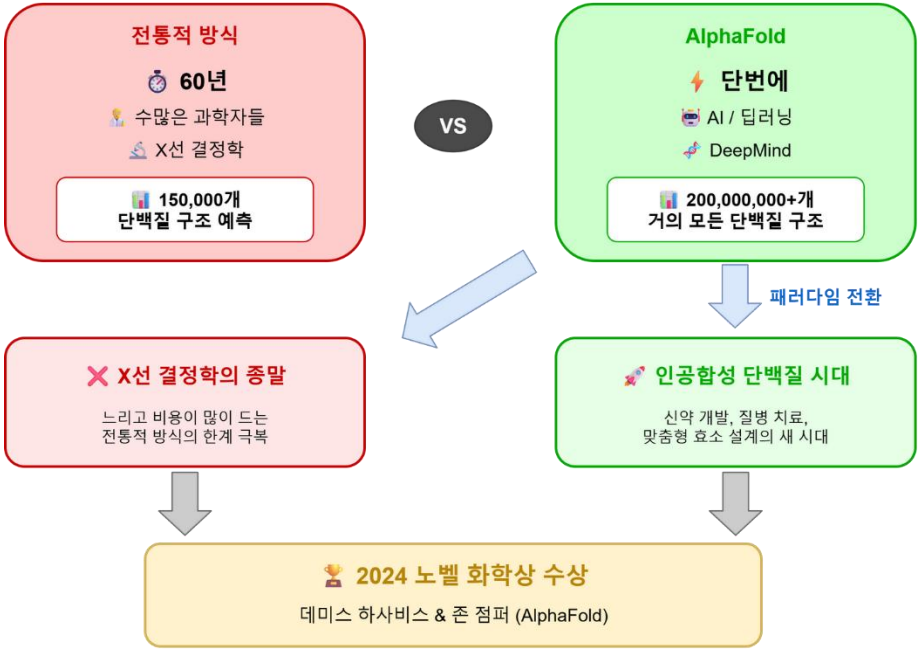


AlphaFold

단백질 구조



전통적 방식의 종말



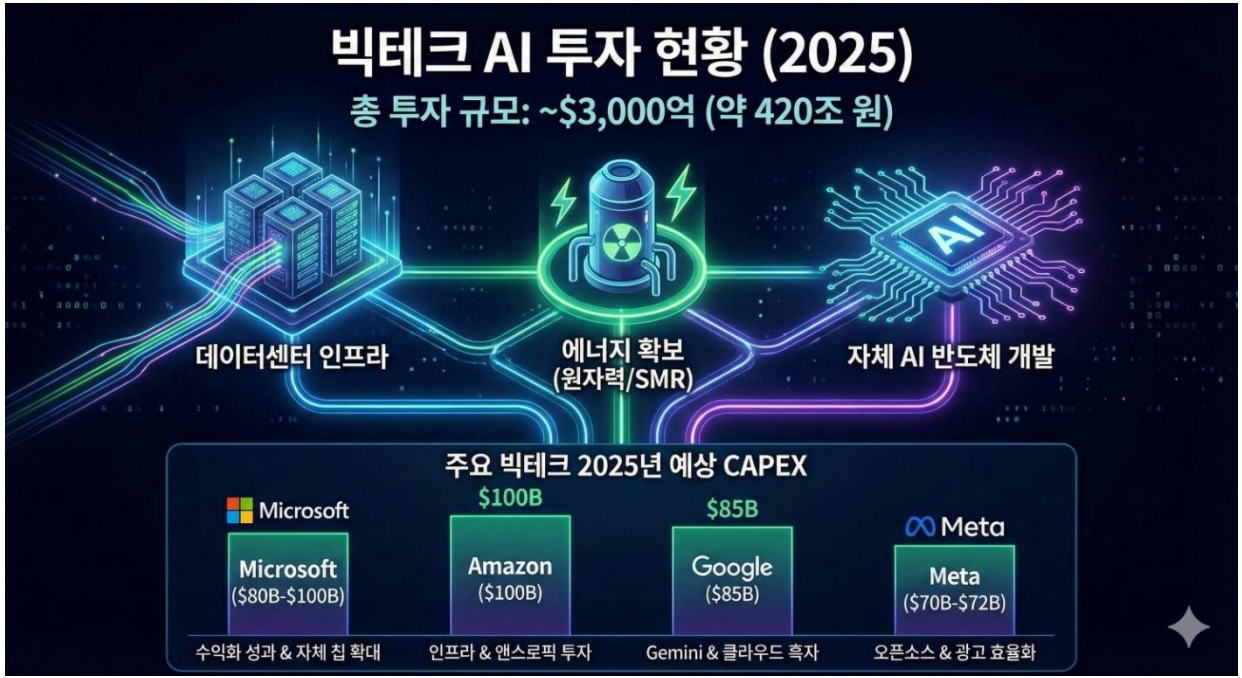
AI의 미래

AI의 호황

- 에너지, 인력, 돈, 자원을 빨아들이는 AI

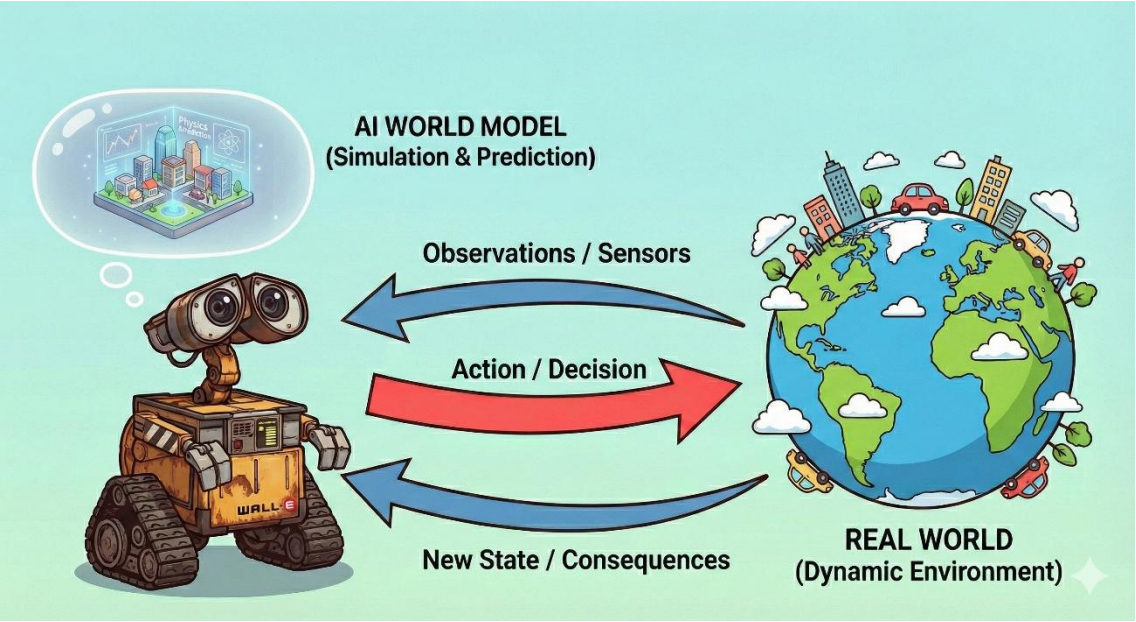
빅테크의 AI 투자 계획

- 4개 빅테크 회사들이 1년에 3000억 달러 규모 투자
  - 누구도 멈출 수 없는 AI의 발전



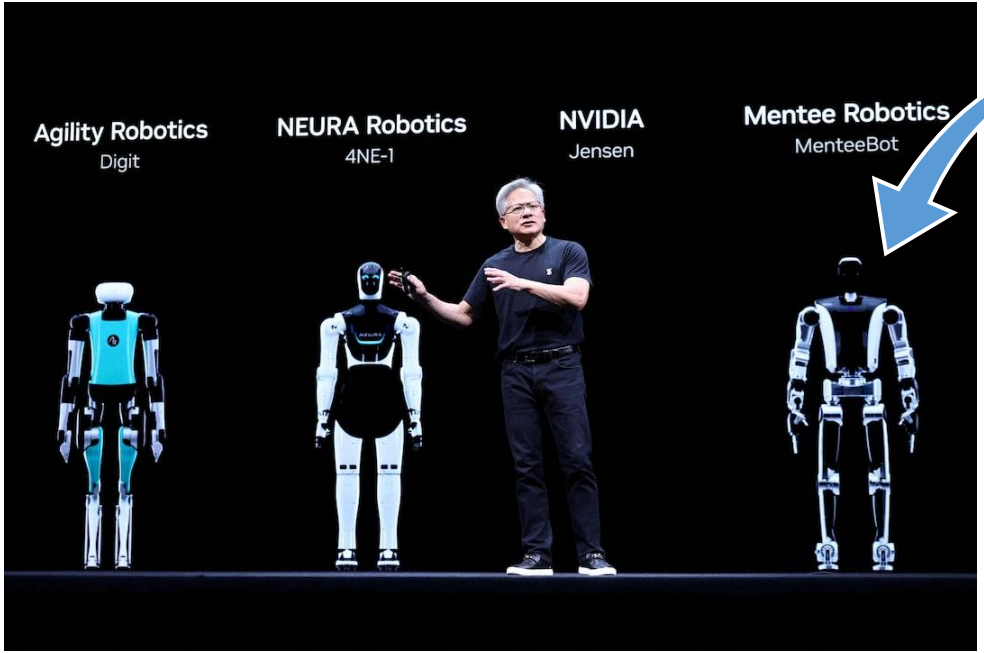
World 모델

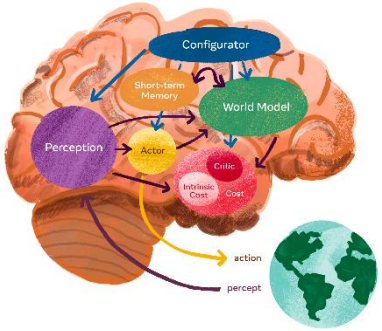
- 세계에 대한 내재적 모델을 가지고 물리적 세계의 원리를 이해하고 행동 결과를 예측하는 AI 시스템



Humanoid의 발전

- Humanoid에 world model을 넣으면? Embodied AI





Exclusive: Nvidia, Foxconn in talks to deploy humanoid robots at Houston AI server making plant | Reuters

배움

인간과 침팬지

- 인간과 침팬지 > 침팬지와 오랑우탕

- 침팬지와 인간의 차이

	침팬지	인간	비고
집단크기	50개체	150개체	언어가 비접촉식 사회 연결을 가능하게 함
도구 개발	O	O	
지식 전파	X	O	인간은 지식을 가르치고 배울 수 있음
시뮬레이션	근미래와 경험	추상과 허구	

뇌의 가소성

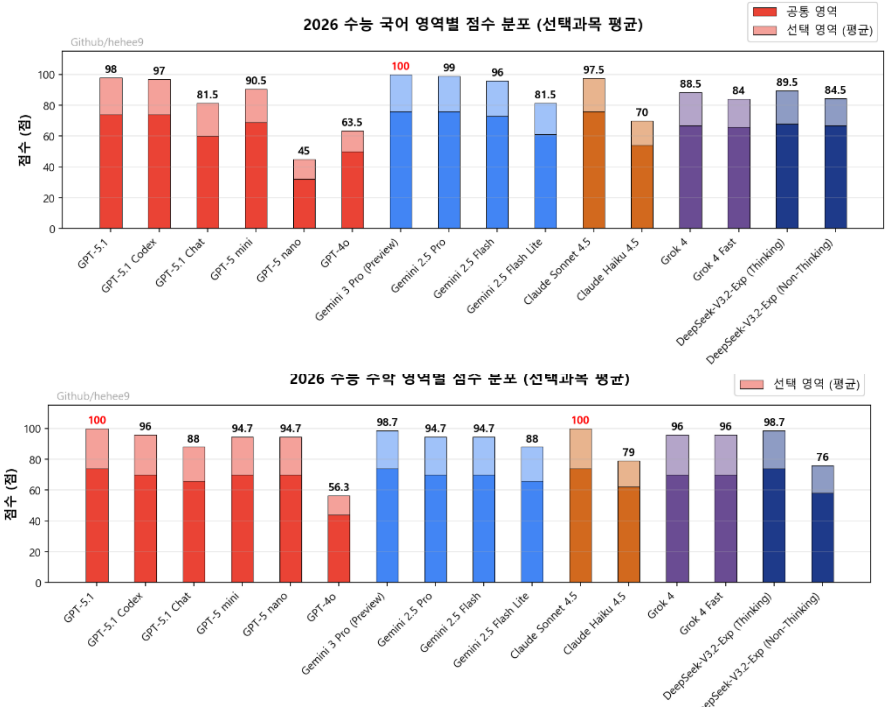
- 경험과 환경에 따라 끊임없이 조정되고 재구성되는 능동적 시스템
- 태어났을 때 시각 장애인이었던 환자가 시각을 회복하면 볼 수 있을까?
- 어렸을 때 언어를 배우지 못한 소녀가 말을 할 수 있을까?
- 사회에 나가기까지 20년 이상의 배움이 필요하고 증가 추세

인간 뇌의 가소성 – 고등과학원 HORIZON

배움

- 2026년도 수능에서 450점 만점에 440점

순위	모델명	원점수	추정 등급컷(2025.11.18기준)
1st	o1-Preview	97	1등급
2nd	o1-mini	78	4등급
3rd	gpt-4o	75	4등급
4th	gpt-4o-mini	59	5등급
5th	gpt-3.5-turbo	16	8등급



수능 국어 '1등급' 받아든 AI "인간은 AI 더 의심하고 학습해야" - 경향신문 GitHub - hehee9/2026-CSAT: 2026 대입 수능 시험 LLM 풀이 결과 모음

노동의 가치와 해방

인력 감축

- 노동 가치의 상실
- 올해 미국 빅테크 인력 감원 증가

기업	사례
마이크로소프트 Microsoft	•5월: 제품·엔지니어링 부서를 중심 6000명 해고 •7월: 전체 직원 4%에 규모인 9000명 감원
메타 Meta	•2월: 저성과자 3600명 해고 •10월: AI 부문 600명 감원
구글 Google	•2월: 클라우드 부문 100명 미만 감원 •4월: 플랫폼·디바이스 부문 수백명 감원 •5월: 판매·파트너십 부문 200명 해고 •6월: 클라우드 부문 100명 이상 감원
아마존 amazon	•10월: 본사 인력 10% 규모인 3만명 감원 추진

자료: 각사 및 외신 보도 종합 The JoongAng

AI '일자리 침공' 현실로, 아마존 3만명 줄인다 | 중앙일보

노동의 해방

- 기본 소득 시대
- 인간은 노동으로부터 해방될 수 있을까?

[AI패권전쟁 한국의 승부수]머스크도, 챗GPT 창시자도..."기본소득 시대 온다" - 아시아경제

의식과 자유의지

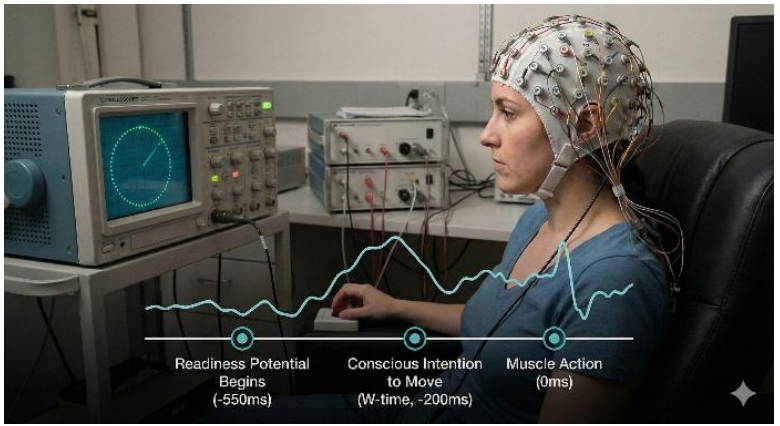
의식

- 구글의 한 개발자가 구글의 인공지능(AI) 챗봇 람다에 의식이 있다고 주장
- 통합 정보 이론: 어떤 시스템이 정보를 **통합**하는 능력이 있다면, 그 시스템은 그만큼의 **의식**을 가진다
- 인간의 뇌를 On/Off할 수 있는 Claustrum

[이슈] "람다, 너는 의식이 있어?" "응, 그런 것 같아"

자유의지

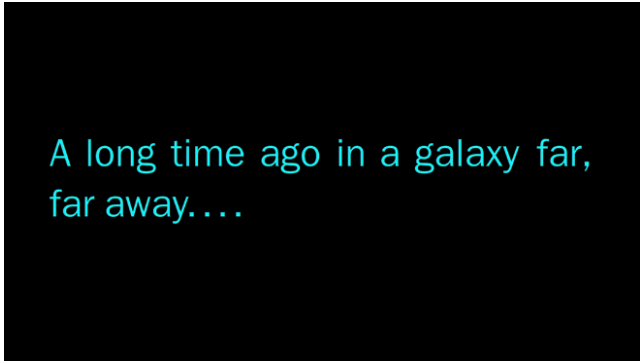
- 우리의 결정보다 뇌가 먼저 정했을까?(Libet, 1983)
- $1,000를 어느 자선 단체에 기부할지 고민해서 선택(Uri Maoz et al., 2019)
- 가치 있고 신중한 선택에서는 전위가 나타나지 않음



우리의 결정보다 먼저 뇌에서 벌어지는 일 | 세상의 결말이 다 정해져 있는 걸까..?

새로운 지적인 존재

- 현생 인류가 외계인보다 먼저 맞이하는 지적인 존재
  - 의식과 자유 의지를 가질까?

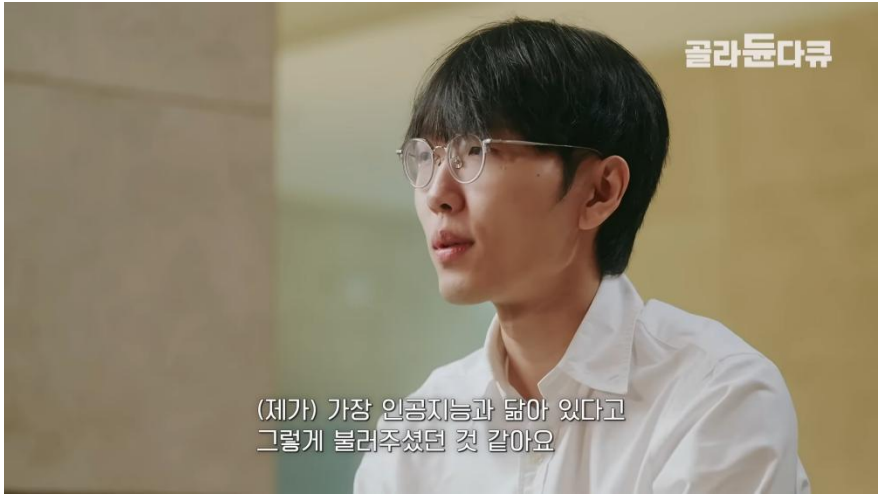


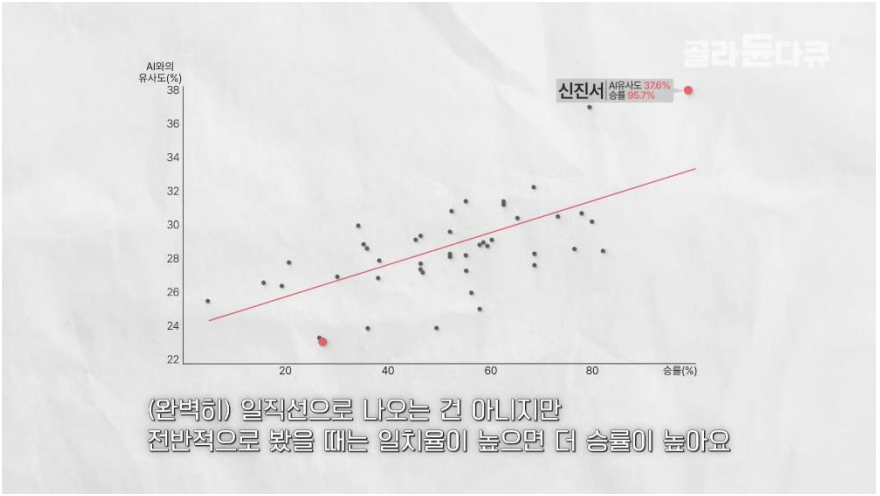
Why are we only left among human races?

英 학자 "외계인 못 만나는 이유는 AI" : 네이버 블로그

Alphago 이후

- AI에서 배우는 바둑

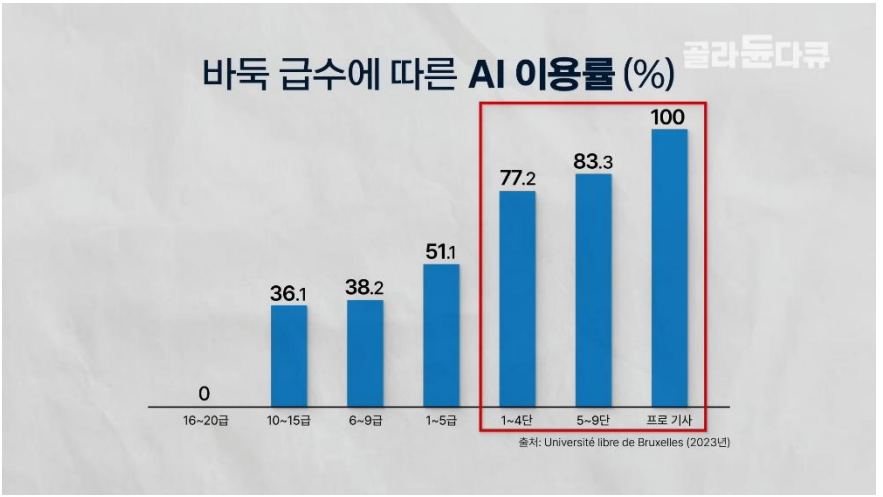




[이젠 옛날로 돌아갈 수 없다. 모든 것이 인공지능인 바둑계 충격적인 근황 | 알파고 10주년 | AI | 신진서 | 이세돌 | 다큐프라임 | #골라듄다큐]

Alphago 이후

- 전문가일수록 더 잘 활용



[이젠 옛날로 돌아갈 수 없다. 모든 것이 인공지능인 바둑계 충격적인 근황 | 알파고 10주년 | AI | 신진서 | 이세돌 | 다큐프라임 | #골라듄다큐]

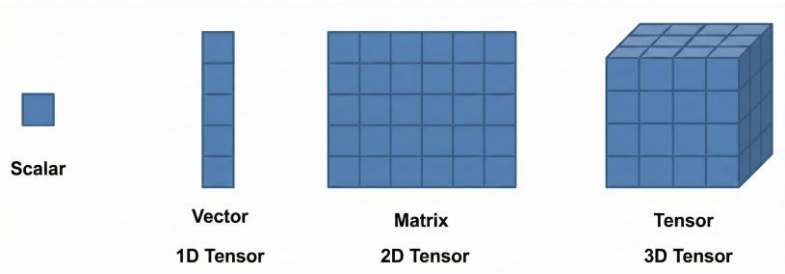
Alphago 이후

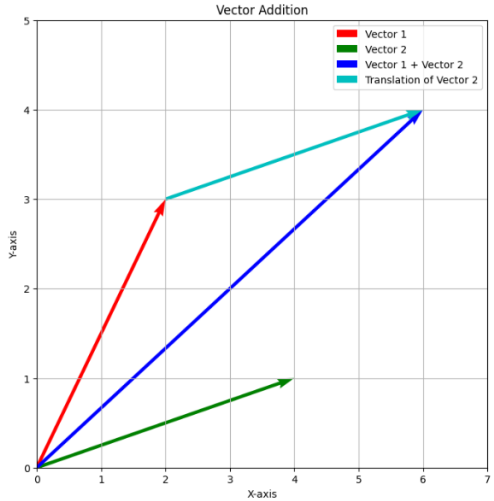
- AI를 어떻게 바라봐야 할까

수학적 사전지식

백터: 덧셈

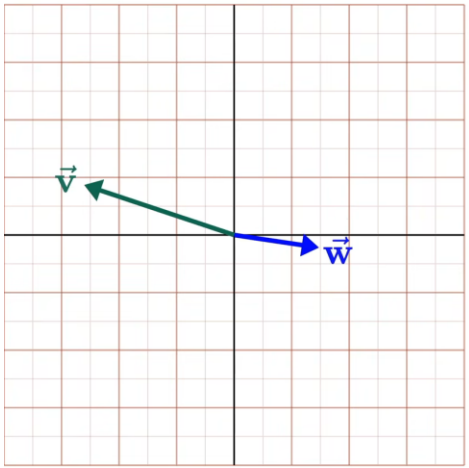
- 좌표 공간 상에서 한 점을 향하는 화살표





백터: 내적(Dot Product)

- 백터 간의 유사도 계산:Scalar값



$$\underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix}}_{\text{Dot product}} = \begin{matrix} v_1 w_1 \\ + \\ v_2 w_2 \\ + \\ v_3 w_3 \\ + \\ \vdots \\ + \\ v_n w_n \\ \| \\ -4.04 \end{matrix}$$

- 반대방향이면 –
- 같은 방향이면 +
- 직교하면 ?

트랜스포머, ChatGPT가 트랜스포머로 만들어졌죠. - DL5

행렬

요소별 곱셈

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \times 1 & 2 \times 0 \\ 3 \times 0 & 4 \times 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

행렬 곱셈

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} @ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (1 \times 1 + 2 \times 0) & (1 \times 0 + 2 \times 1) \\ (3 \times 1 + 4 \times 0) & (3 \times 0 + 4 \times 1) \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

요소별 덧셈

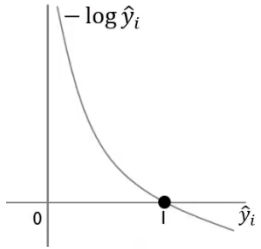
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix} = \begin{bmatrix} 11 & 22 \\ 33 & 44 \end{bmatrix}$$

Transpose

$$A = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}_{2 \times 3} \qquad A^T = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}_{3 \times 2}$$

지수함수 로그함수

- 지수함수
  - 작은 차이를 극적으로 확대(단어 A: 10점, 단어 B: 8점, 단어 C: 1점 -> 단어 A e^{10}:약 22026, 단어 B (e^{8}): 약 2981, 단어 C (e^{1}): 약 2.7)
  - 모든 숫자를 양수로 변환, 모호한 점수차를 확실한 확률 차이로 변환
- 로그함수
  - 2^x = 10일 때, 실수 x의 값은 얼마인가요? $\log_2 10$
  - 곱셈을 덧셈으로 변경할 수 있음: 연산속도향상, 0으로 소멸해버리는 '언더플로우' 방지
    $\log(A \times B) = \log A + \log B \qquad \log(A/B) = \log A - \log B$
  - 틀렸을 때 강력한 피드백을 줌



- 자연상수(e)
  - 자연 상수를 지수나 로그 함수에 사용: e^x(모든 지수함수는 자연상수로 표현 가능), $\log_e(x) = \ln(x)$
  - 미분했을 때 자기 자신이 나옴
    - 지수함수(e^x)를 미분하면 그대로 e^x 가 됩니다.
    - 자연로그($\ln x$)를 미분하면 아주 깔끔하게 $1/x$가 됩니다.

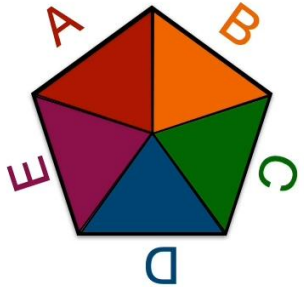
[로그의 미분 1편] 자연로그부터 꿰뚫자! 제대로 배우면 너무 쉽다! - YouTube

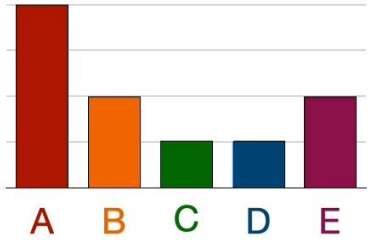
KL Divergence, Cross Entropy

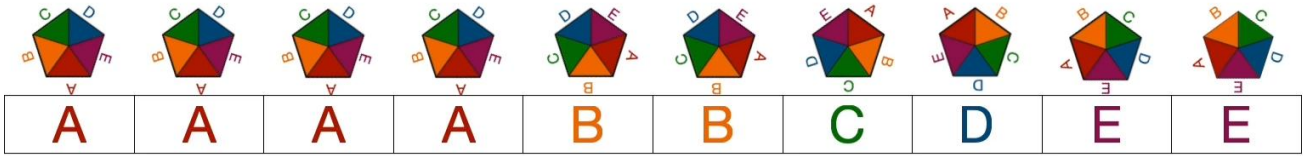
- 5변을 가진 주사위를 굴려서 나올 확률

5 sided die

$p(A) = 0.4$
$p(B) = 0.2$
$p(C) = 0.1$
$p(D) = 0.1$
$p(E) = 0.2$







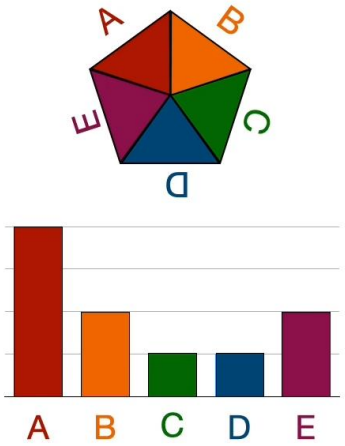
https://www.youtube.com/watch?v=sjgZxuCm_8Q

Cross Entropy

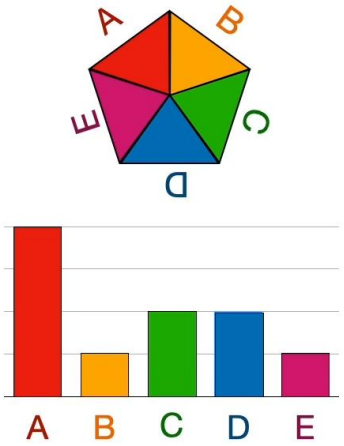
- 어느 확률이 주어진 시퀀스를 재현한 것일까?

A	A	A	A	B	B	C	D	E	E

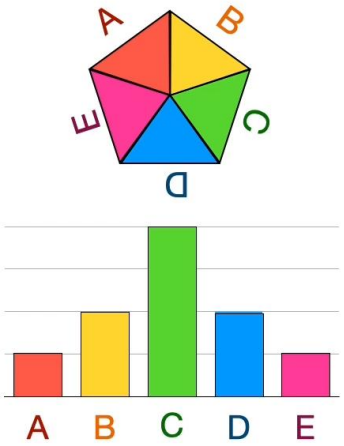
Die 1 (original)



Die 2



Die 3



Cross Entropy

- Die 1이 Sequence를 가질 확률

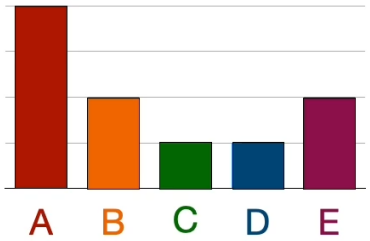
A	A	A	A	B	B	C	D	E	E

$$\log(0.4 \cdot 0.4 \cdot 0.4 \cdot 0.4 \cdot 0.2 \cdot 0.2 \cdot 0.1 \cdot 0.1 \cdot 0.2 \cdot 0.2) = -14.71$$

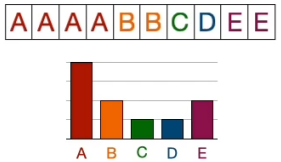
$$\log(0.4)+\log(0.4)+\log(0.4)+\log(0.4)+\log(0.2)+\log(0.2)+\log(0.1)+\log(0.1)+\log(0.2)+\log(0.2)$$

$$-\frac{4\log(0.4)+2\log(0.2)+1\log(0.1)+1\log(0.1)+2\log(0.2)}{10} = 1.471$$

Die 1

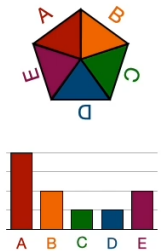


$p(A) = 0.4$
$p(B) = 0.2$
$p(C) = 0.1$
$p(D) = 0.1$
$p(E) = 0.2$



P P

$$-0.4\log(0.4)$$
$$-0.2\log(0.2)$$
$$-0.1\log(0.1)$$
$$-0.1\log(0.1)$$
$$-0.2\log(0.2)$$



$$-\sum_i p_i \log(p_i) = 1.471$$

Cross Entropy

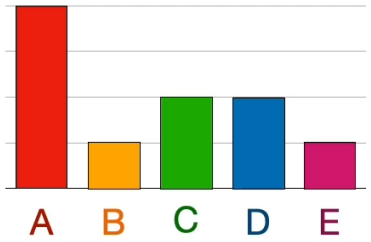
- Die2가 Sequence를 가질 확률

A	A	A	A	B	B	C	D	E	E
0.4	0.4	0.4	0.4	0.1	0.1	0.2	0.2	0.1	0.1

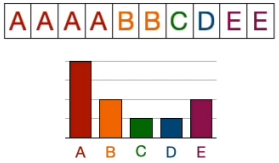
$= 0.0000001024$

$$-\frac{4\log(0.4) + 2\log(0.1) + 1\log(0.2) + 1\log(0.2) + 2\log(0.1)}{10} = 1.609$$

Die 2

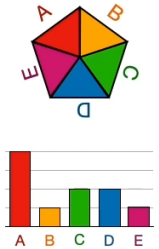


$q(A) = 0.4$
$q(B) = 0.1$
$q(C) = 0.2$
$q(D) = 0.2$
$q(E) = 0.1$



P Q

$$-0.4\log(0.4)$$
$$-0.2\log(0.1)$$
$$-0.1\log(0.2)$$
$$-0.1\log(0.2)$$
$$-0.2\log(0.1)$$



Cross entropy $H(P\,|\,Q) = -\sum_i p_i \log(q_i) = 1.609$

Cross Entropy

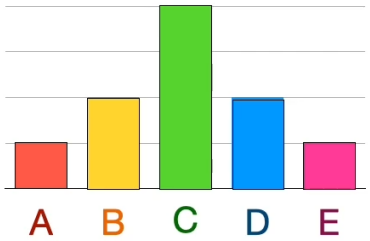
- Die3가 Sequence를 가질 확률

A	A	A	A	B	B	C	D	E	E
0.1	0.1	0.1	0.1	0.2	0.2	0.4	0.2	0.1	0.1

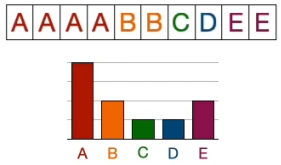
$= 0.0000000032$

$$-\frac{4\log(0.1) + 2\log(0.2) + 1\log(0.4) + 1\log(0.2) + 2\log(0.1)}{10} = 1.956$$

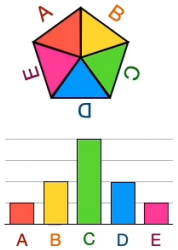
Die 3



$r(A) = 0.1$
$r(B) = 0.2$
$r(C) = 0.4$
$r(D) = 0.2$
$r(E) = 0.1$



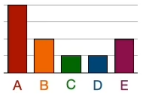
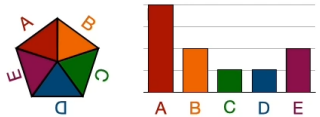
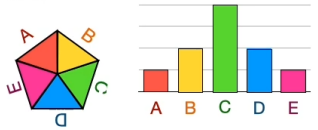
P	R
-0.4	$\log(0.1)$
-0.2	$\log(0.2)$
-0.1	$\log(0.4)$
-0.1	$\log(0.2)$
-0.2	$\log(0.1)$



Cross entropy $H(P\,|\,R) = -\sum_i p_i \log(r_i) = 1.956$

KL-Divergence

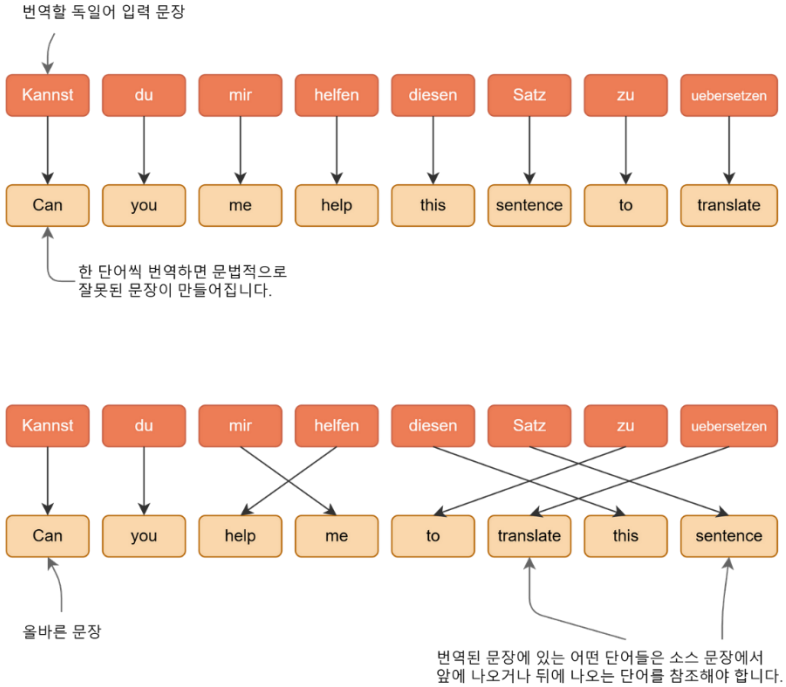
- Cross Entropy에서 실제 데이터 자체의 Entropy를 뺀 값 (모델이 틀린 만큼의 벌점)

Sequence	Probability	Cross-entropy	KL-divergence
A A A A B B C D E E 			$D_{KL}(P\|Q)$ = Cross Entropy(P, Q) - Entropy(P)
Die 1 	$0.4^4 \cdot 0.1^2 \cdot 0.2^1 \cdot 0.2^1 \cdot 0.1^2$  0.0000004096 	$-0.4\log(0.4) - 0.2\log(0.2) - 0.1\log(0.1)$ $-0.1\log(0.1) - 0.2\log(0.2)$ $H(P) = 1.471$ (entropy)	$\mathbb{D}(P\|P) = 1.471 - 1.471$ 0
Die 2 	$0.4^4 \cdot 0.1^1 \cdot 0.2^2 \cdot 0.2^2 \cdot 0.1^1$ 0.0000001024	$-0.4\log(0.4) - 0.2\log(0.1) - 0.1\log(0.2)$ $-0.1\log(0.2) - 0.2\log(0.1)$ $H(P\|Q) = 1.609$	$\mathbb{D}(Q\|P) = 1.609 - 1.471$ 0.138
Die 3 	$0.4^1 \cdot 0.1^2 \cdot 0.2^4 \cdot 0.2^2 \cdot 0.1^1$ 0.0000000032 	$-0.4\log(0.1) - 0.2\log(0.2) - 0.1\log(0.4)$ $-0.1\log(0.2) - 0.2\log(0.1)$ $H(P\|R) = 1.956$	$\mathbb{D}(R\|P) = 1.956 - 1.471$ 0.485

언어 모델의 발전

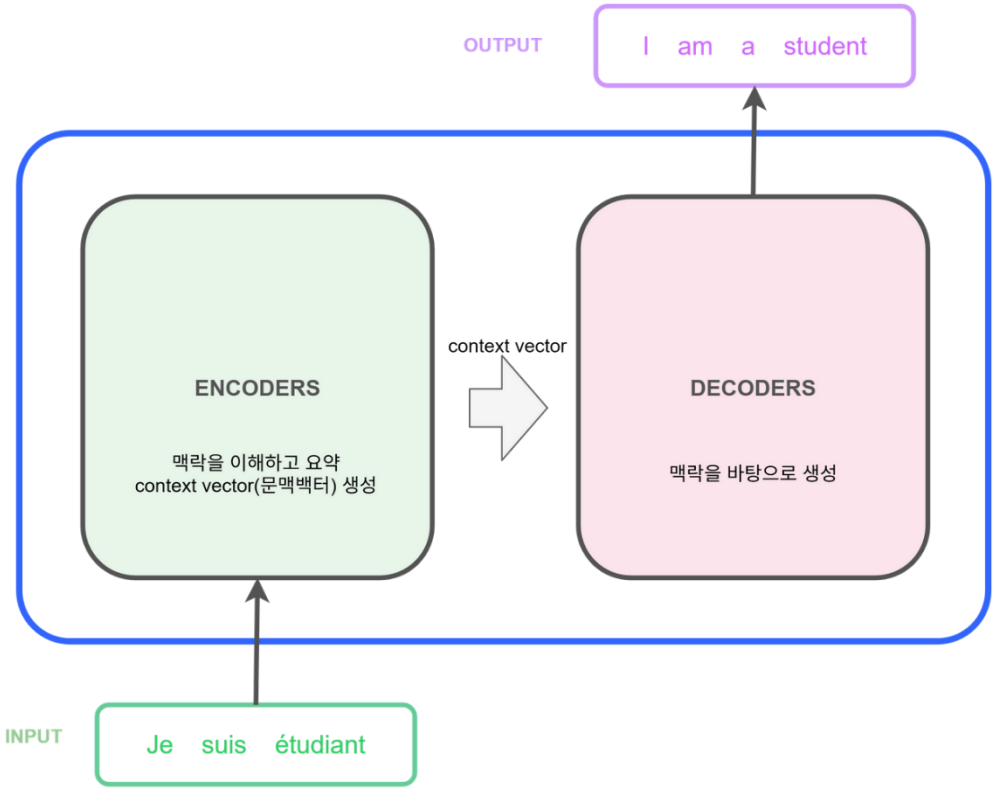
규칙기반 번역

- 형태소 분석-> 구문 분석 -> 변환 -> 규칙으로 어순 배열



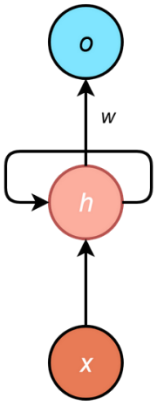
언어모델: Encoders, Decoders

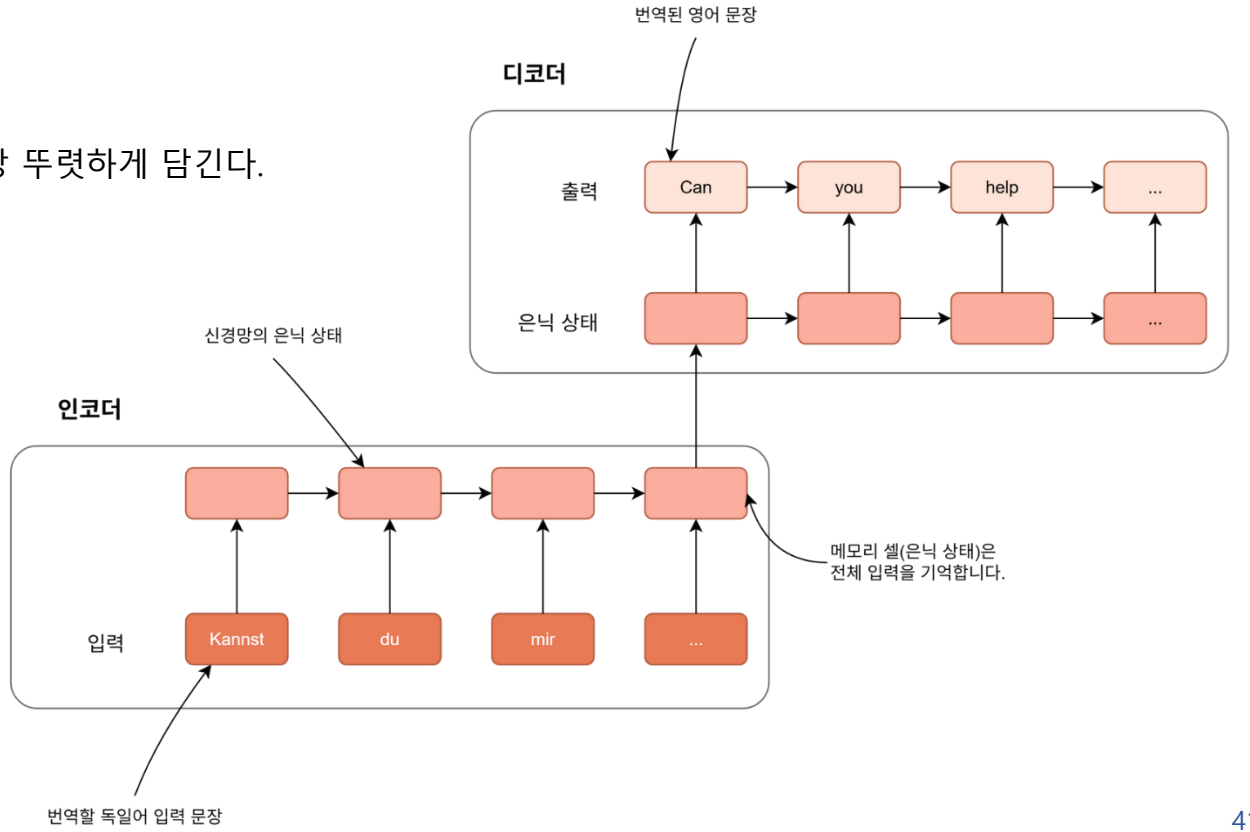
- 입력을 이해하고, 이해를 바탕으로 생성함



언어모델:RNN

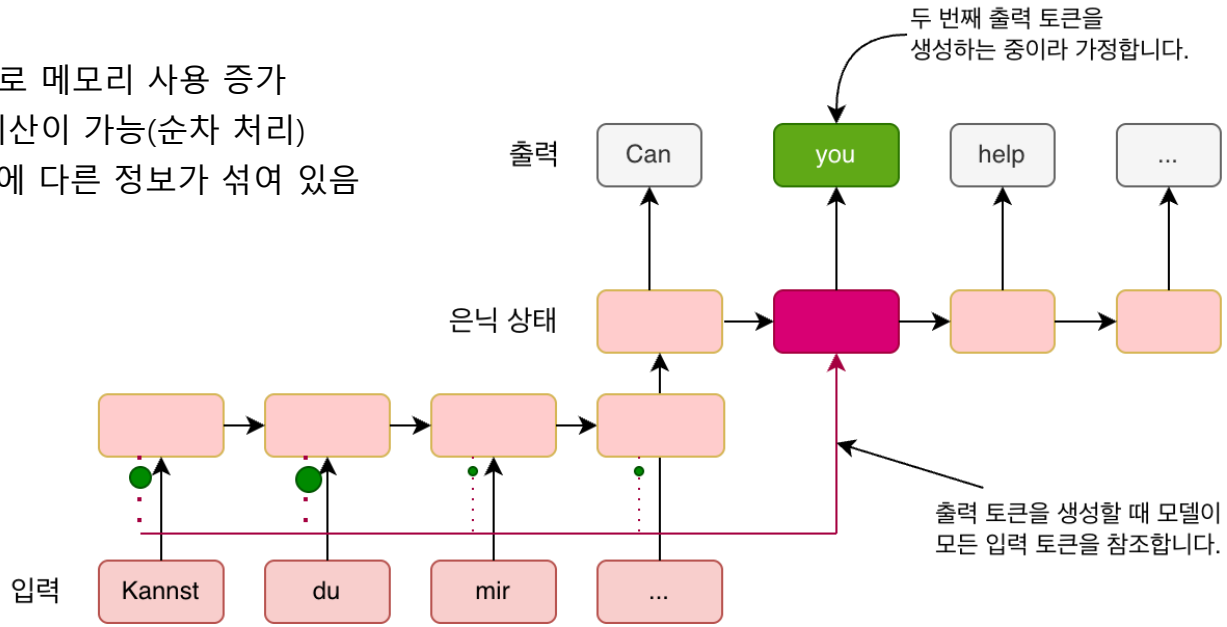
- "문맥(Context)"을 기억하는 메모리의 탄생, 가변 길이 처리
- 단점
  - 길수록 잊혀진다.
  - 가장 마지막 정보가 가장 뚜렷하게 담긴다.





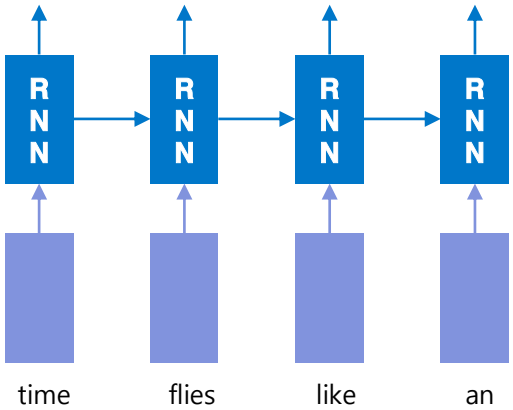
언어모델: RNN Attention

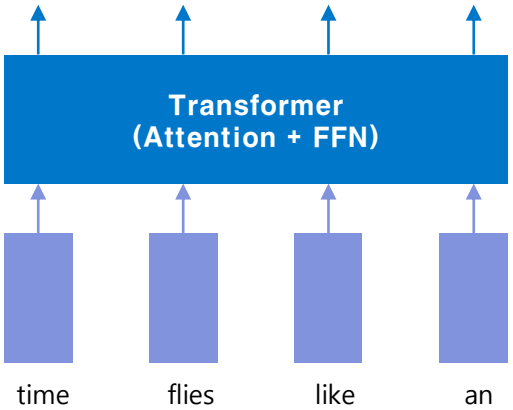
- 입력 단어들과의 Attention 참고해서 성능 향상, 어떤 단어를 주목할지를 학습
- 단점
  - 이전 은닉 상태의 저장으로 메모리 사용 증가
  - 이전 계산을 해야 다음 계산이 가능(순차 처리)
  - 참고하고자 하는 Hidden에 다른 정보가 섞여 있음



언어모델: Transformer

- 트랜스포머의 핵심은 어텐션(Attention)과 병렬화 가능한 아키텍처

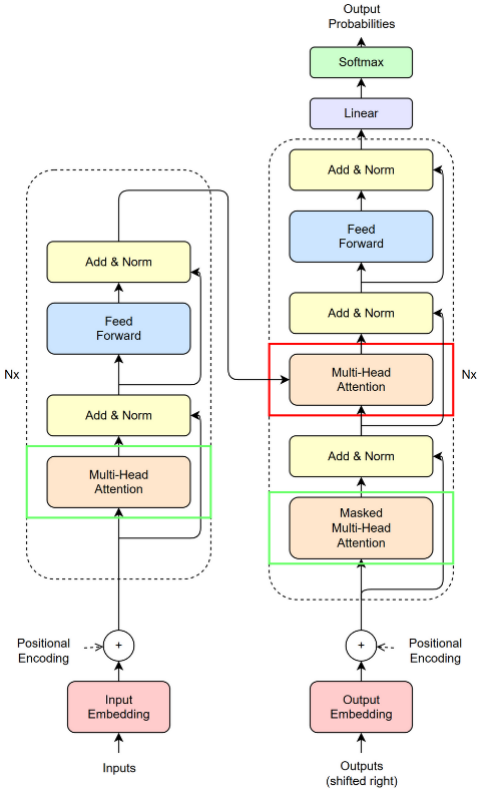


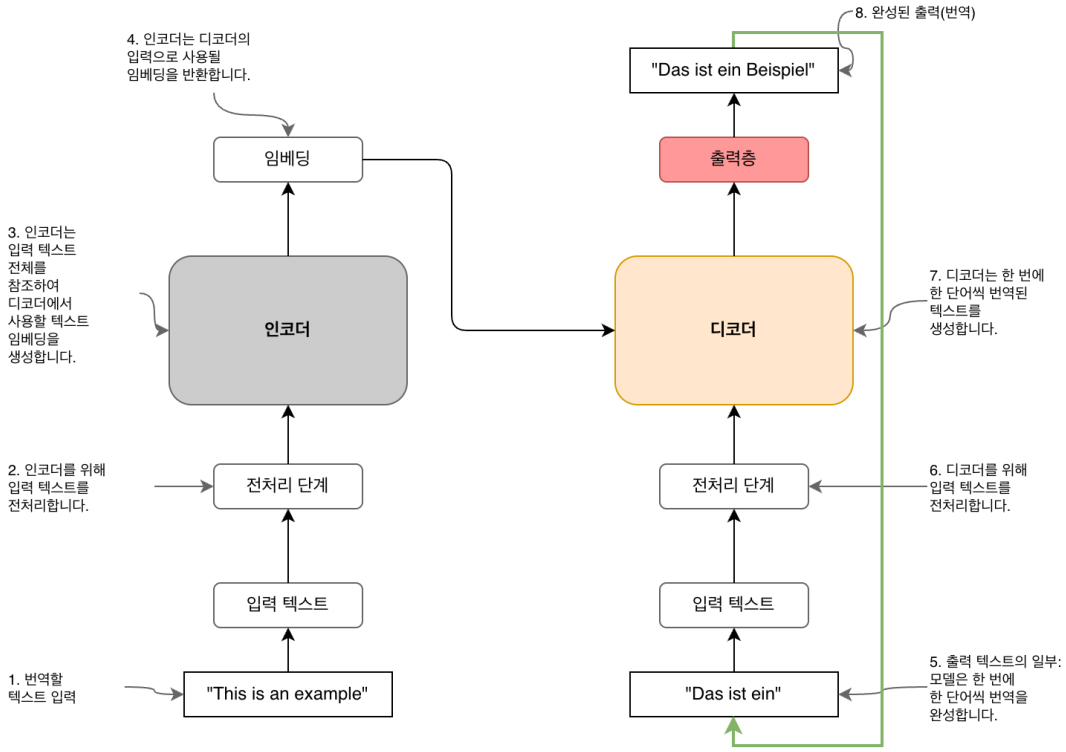


RNN is hard to parallelize and not time-efficient !

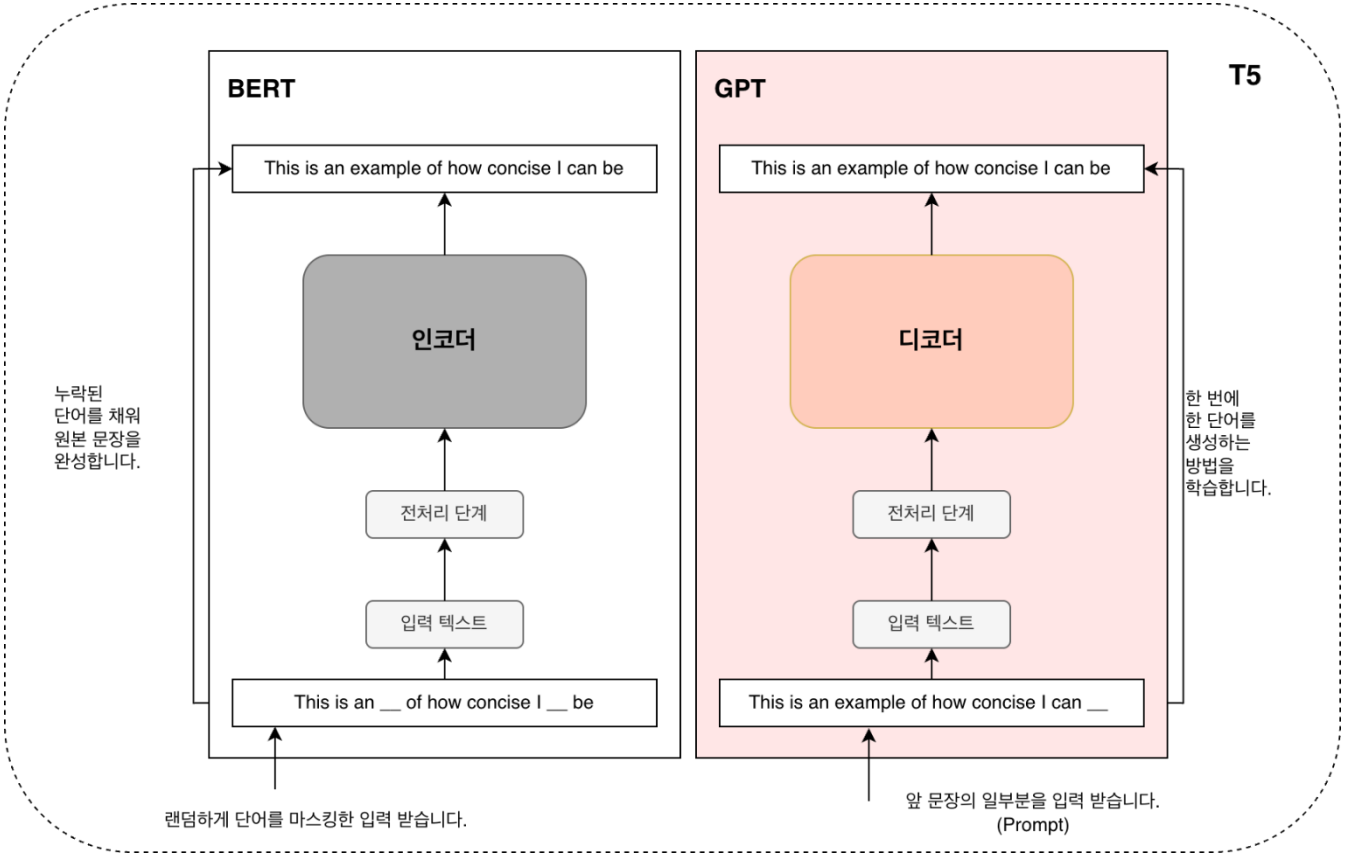
언어모델: Transformer

- Attention과 FFN으로 구성





언어모델:BERT, GPT, T5



언어모델: BERT, GPT

- Pre-training: Self-supervised Learning
  - BERT,T5: Masked Language Model (MLM), 양방향 학습
  - GPT: Causal Language Model (CLM) - Next Word Prediction

BERT, T5

- **빈칸 채우기 문제**
- **양쪽 문맥 모두 참조 가능**

인재개발원은 _____ 에 있습니다

나는 이번 학기에 _____ 강의를 수강합니다

BERT는 트랜스포머의 _____ 구조만을 사용합니다

조성진은 한국이 낳은 훌륭한 _____ 입니다

피보나치 수열은 1, 1, 2, 3, 5, 8, 13, 21, _____ 와 같다

GPT

- **다음 단어 맞추기 문제**
- **앞쪽 문맥만 참조 가능**

인재개발원은 _____

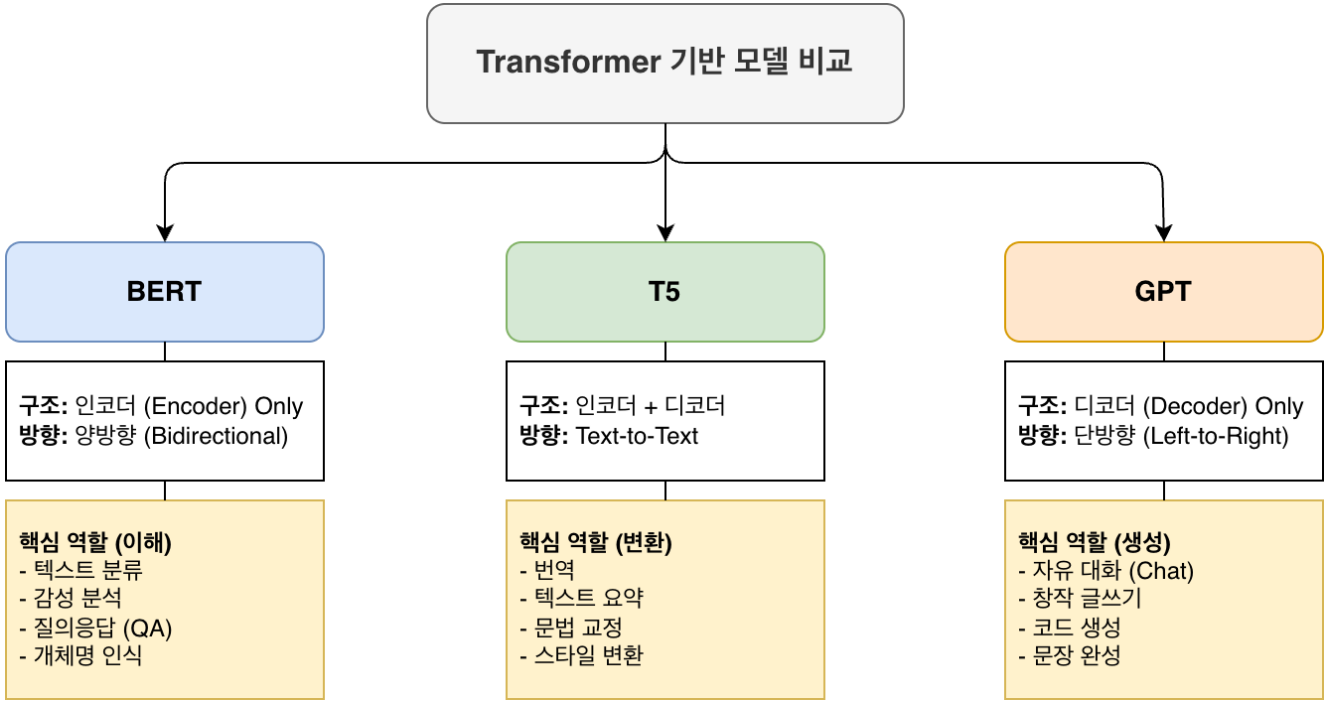
나는 이번 학기에 딥러닝 강의를 _____

GPT는 트랜스포머의 _____

조성진은 한국이 낳은 훌륭한 _____

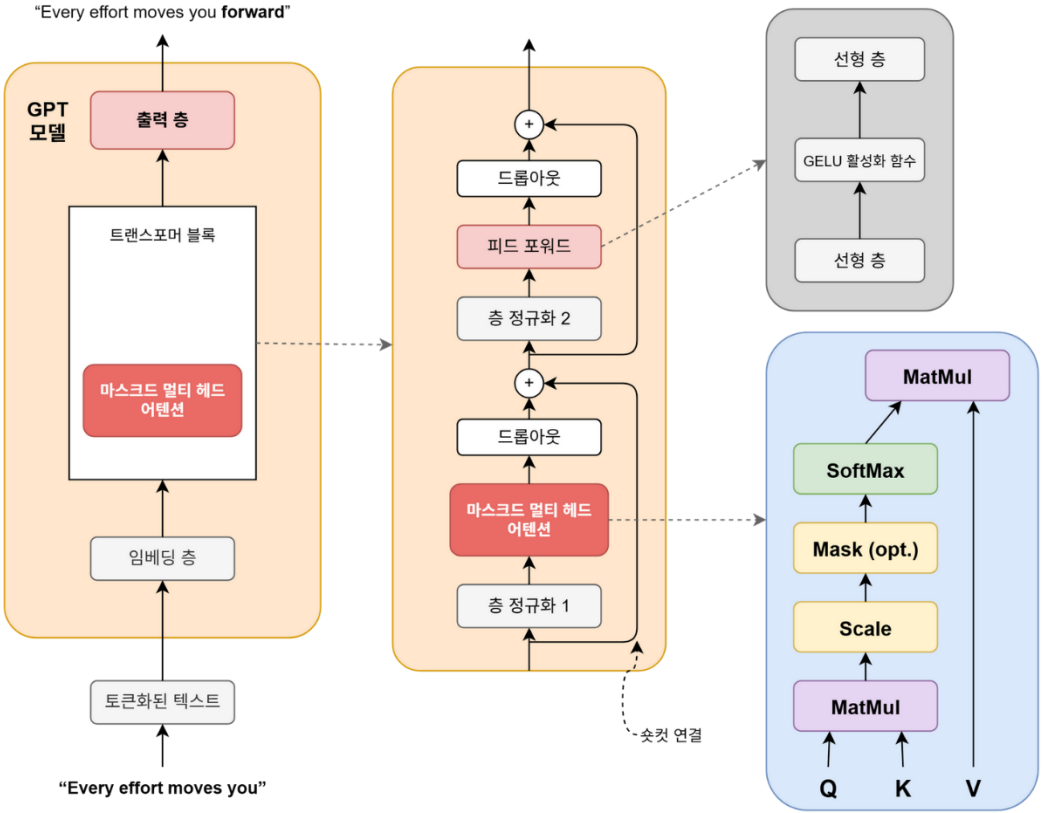
피보나치 수열은 1, 1, 2, 3, 5, 8, 13, 21, _____

언어 모델 비교


Transformer 기반 모델 비교
BERT
T5
GPT
구조: 인코더 (Encoder) Only
방향: 양방향 (Bidirectional)
구조: 인코더 + 디코더
방향: Text-to-Text
구조: 디코더 (Decoder) Only
방향: 단방향 (Left-to-Right)
핵심 역할 (이해)
- 텍스트 분류
- 감성 분석
- 질의응답 (QA)
- 개체명 인식
핵심 역할 (변환)
- 번역
- 텍스트 요약
- 문법 교정
- 스타일 변환
핵심 역할 (생성)
- 자유 대화 (Chat)
- 창작 글쓰기
- 코드 생성
- 문장 완성

언어 모델: GPT

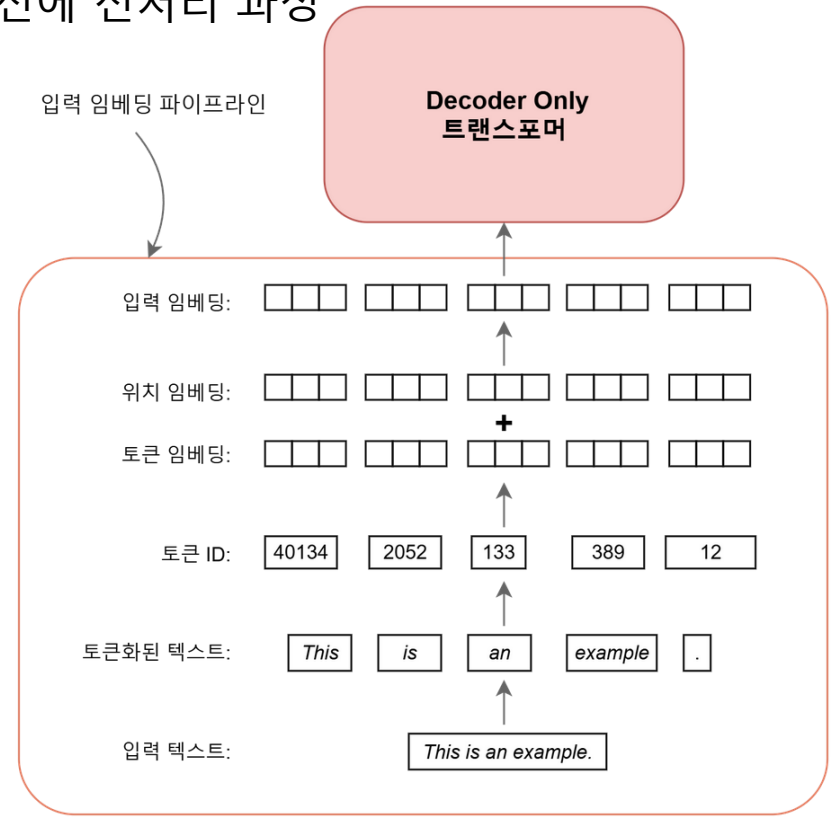
- Decoder만 사용



데이터 입력

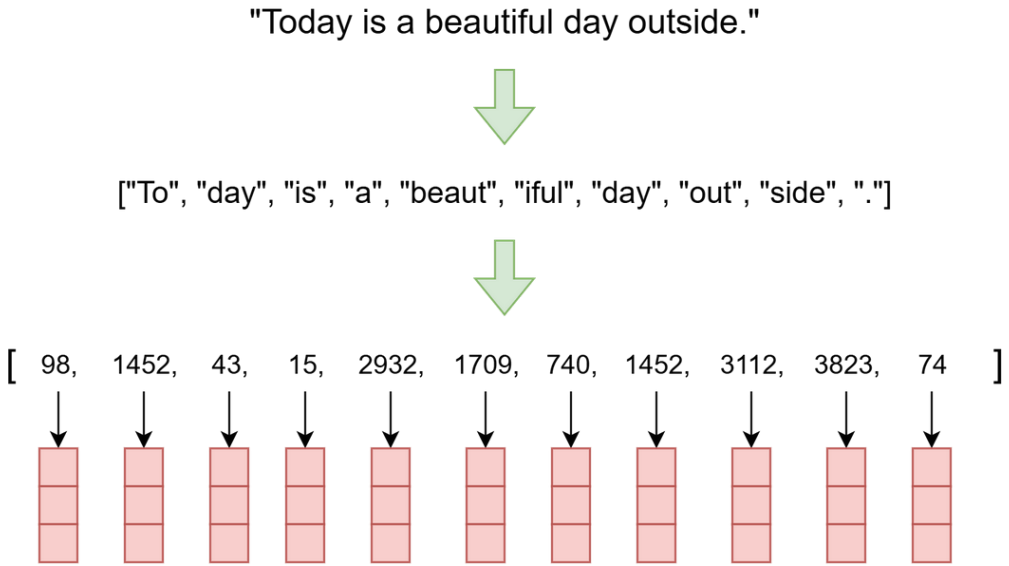
데이터 입력: 전처리

- 모델에 입력하기 전에 전처리 과정



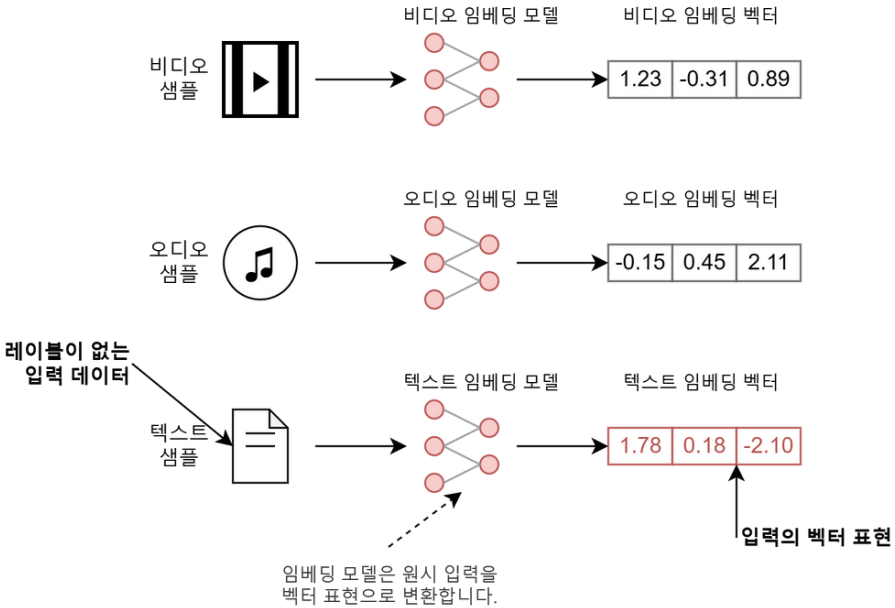
데이터 입력: Tokenizer

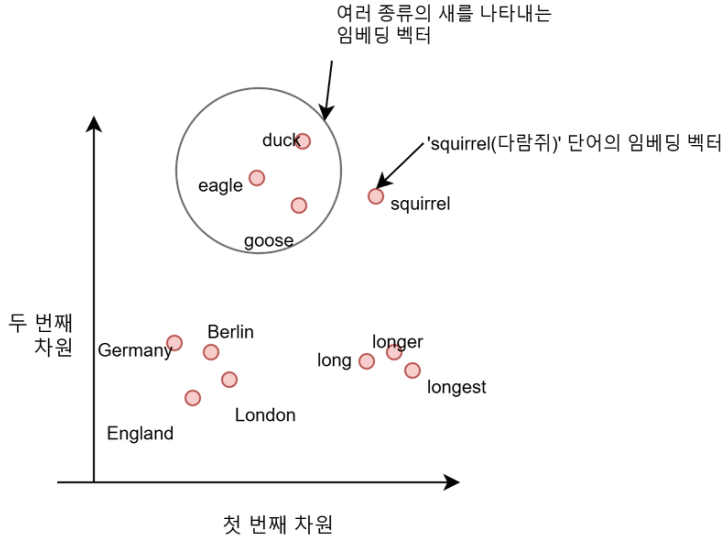
- 입력 문장을 의미 단위로 잘라서 고유 번호를 부여
  - 자주 쓰이는 단어는 가능하면 길게: 데이터 압축을 높일 수 있음
  - 가능하면 의미 단위로 자름



데이터 입력: embedding

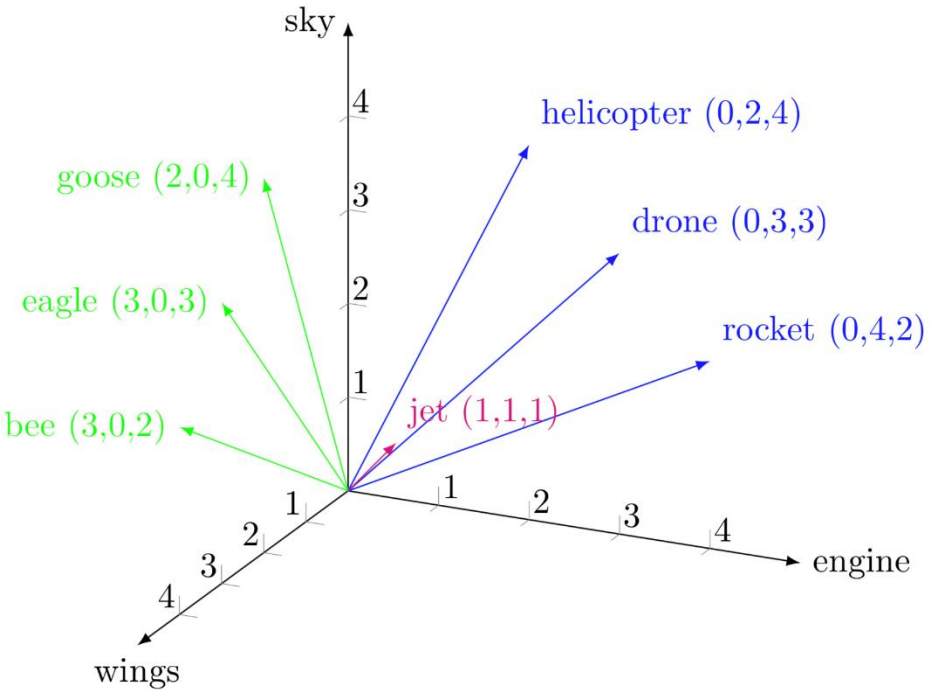
- 어떤 데이터를 벡터로 표현한 것





데이터 입력: embedding

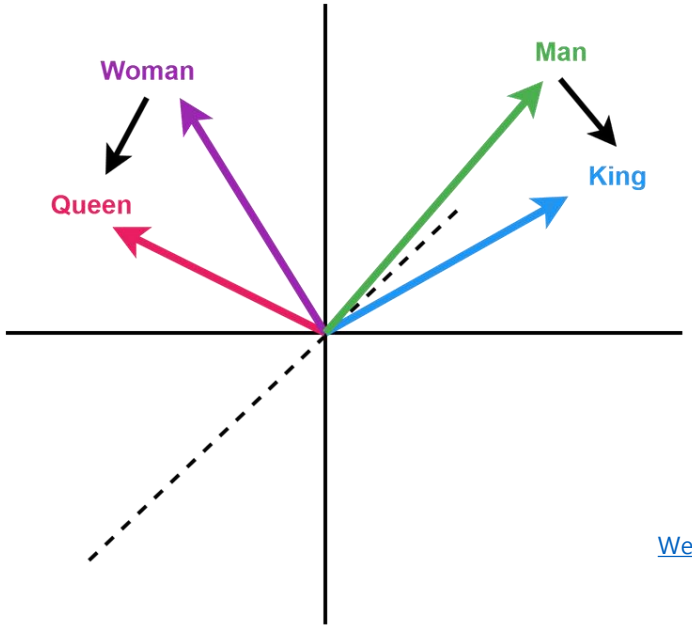
- 벡터 공간의 점으로 맵핑



트랜스포머, ChatGPT가 트랜스포머로 만들어졌죠. - DL5

데이터입력:백터 공간

- 비슷한 의미를 가진 단어는 비슷한 공간에 위치
  - 단어간 의미를 계산할 수 있음
  - Woman – Queen ≈ Man - King



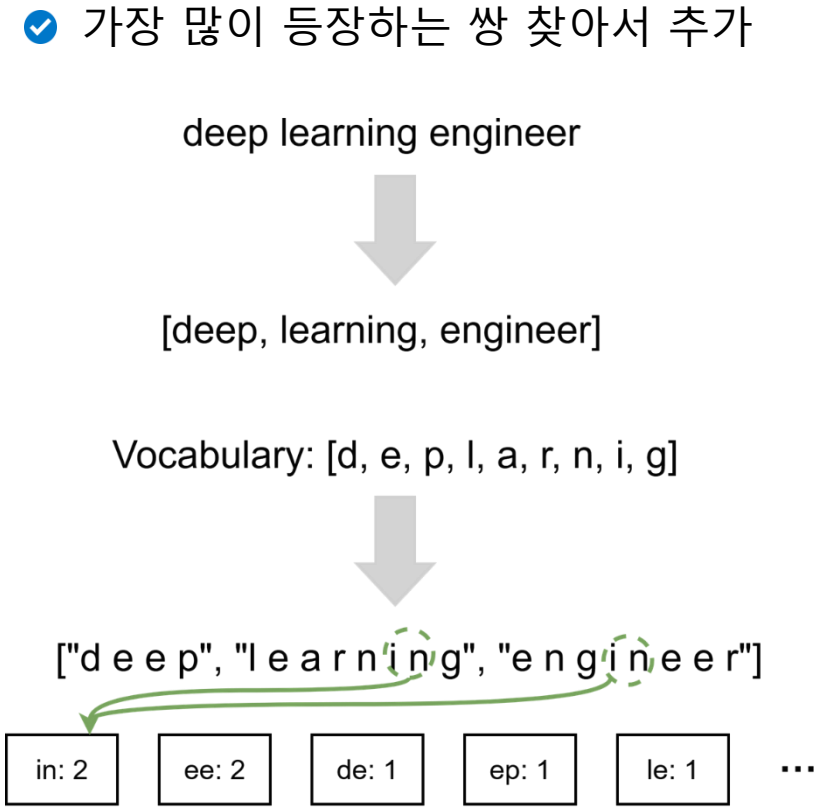
WebVectors: Semantic Calculator

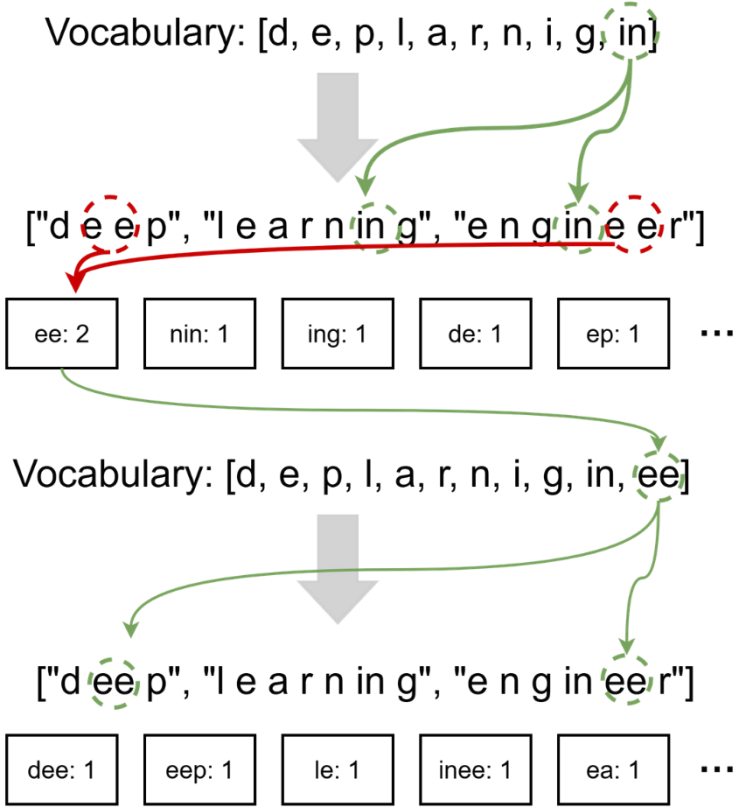
Chapter_1_Exercise_Vector Space.ipynb

WebVectors: Semantic Calculator

데이터입력: BPE Tokenizer

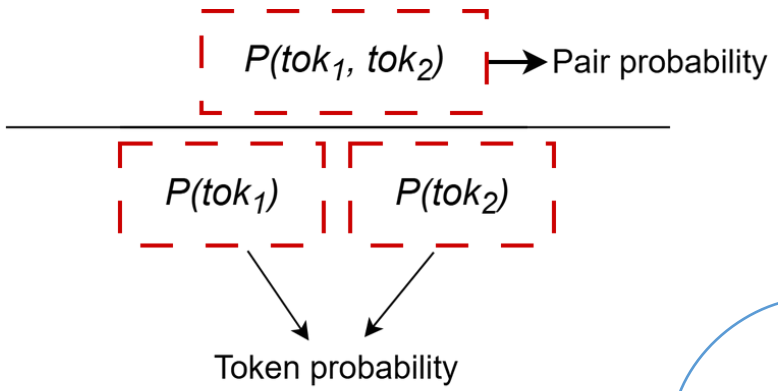
- 가장 많이 등장하는 쌍 찾아서 추가



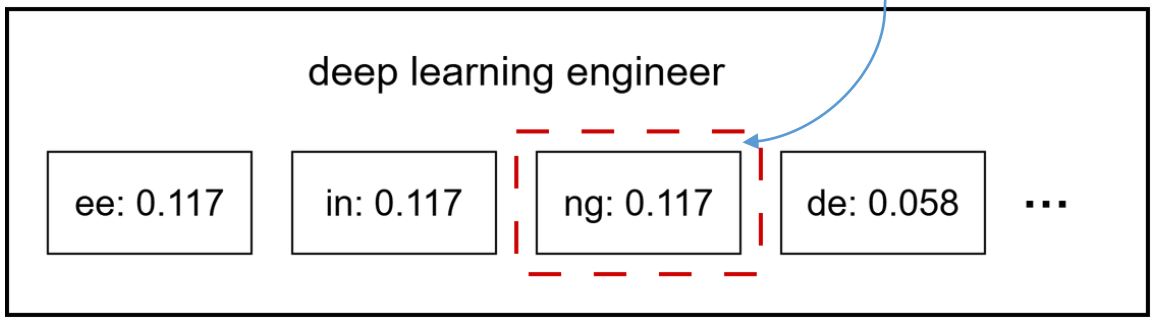


데이터입력: Wordpiece Tokenizer

- 각각 나올 확률보다 쌍으로 나올 확률이 높으면 추가



$$\frac{(ng\text{가 나올 확률})}{n\text{이 나올 확률})(g\text{가 나올 확률})}$$



데이터입력: Unigram Tokenizer

- 최선의 조합을 찾고, 쓸모없는 것은 버림

```
("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

("h", 15) ("u", 36) ("g", 20) ("hu", 15) ("ug", 20) ("p", 17) ("pu", 17) ("n", 16)
("un", 16) ("b", 4) ("bu", 4) ("s", 5) ("hug", 15) ("gs", 5) ("ugs", 5)
```

$P(["p", "u", "g"]) = P("p") \times P("u") \times P("g") = {}^{5}/_{210} \times {}^{36}/_{210} \times {}^{20}/_{210} = 0.000389$

$P(["pu", "g"]) = P("pu") \times P("g") = {}^{5}/_{210} \times {}^{20}/_{210} = 0.0022676$

```
"hug": ["hug"] (score 0.071428)
"pug": ["pu", "g"] (score 0.007710)
"pun": ["pu", "n"] (score 0.006168)
"bun": ["bu", "n"] (score 0.001451)
"hugs": ["hug", "s"] (score 0.001701)
```

Best 조합

hug는 유지

```
"hug": ["hug"] (score 0.071428)
"hugs": ["hug", "s"] (score 0.001701)
```

```
"hug": ["hu", "g"] (score 0.006802)
"hugs": ["hu", "gs"] (score 0.001701)
```

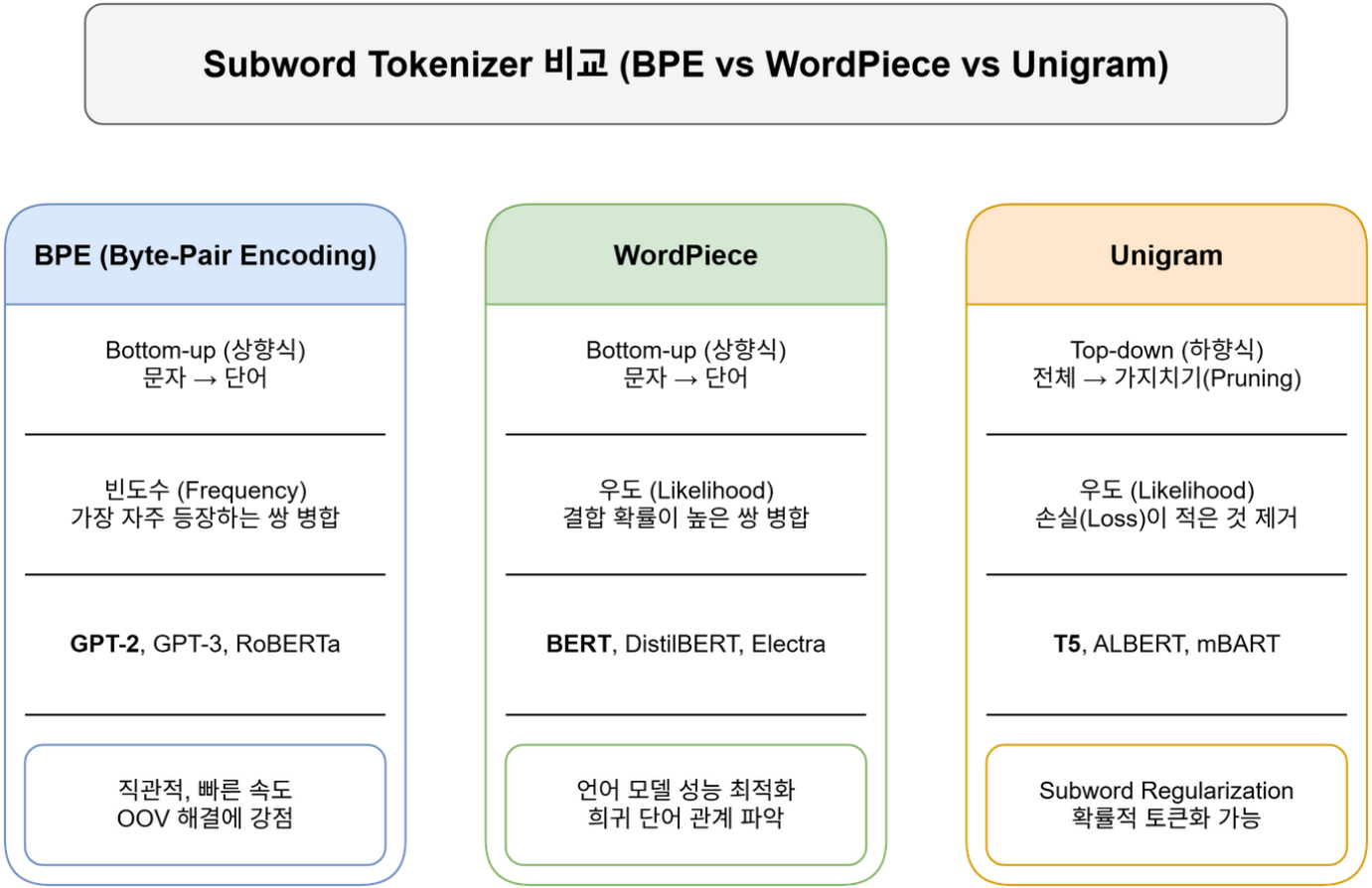
pu는 삭제

```
"pug": ["pu", "g"] (score 0.0022676)
"pun": ["pu", "n"] (score 0.0061678)
```

```
"pug": ["p", "ug"] (score 0.0022676)
"pun": ["p", "un"] (score 0.0061678)
```

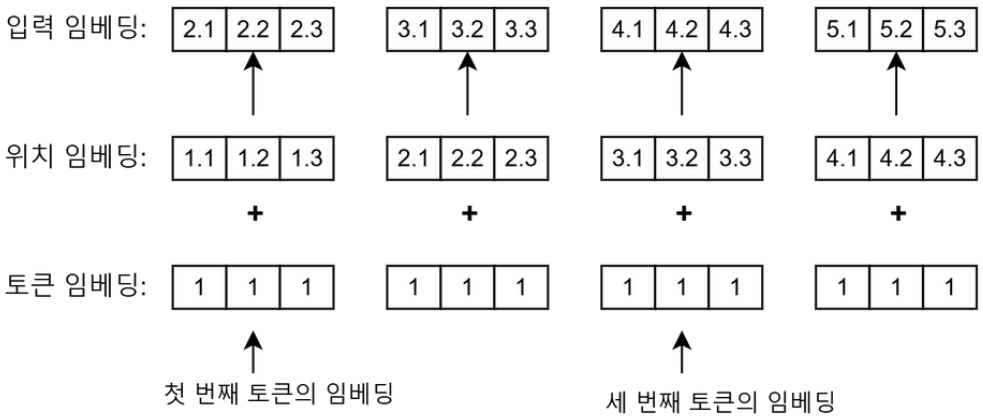
- 확률에 따라 토큰화 결과를 매번 조금씩 다르게 샘플링
  playing → play + ing 또는 pl + ay + ing)
- 통계적 특성을 더 잘 반영하는 효율적인 어휘 사전을 구축
- 반면에 속도는 느림

데이터 입력 – Tokenizer



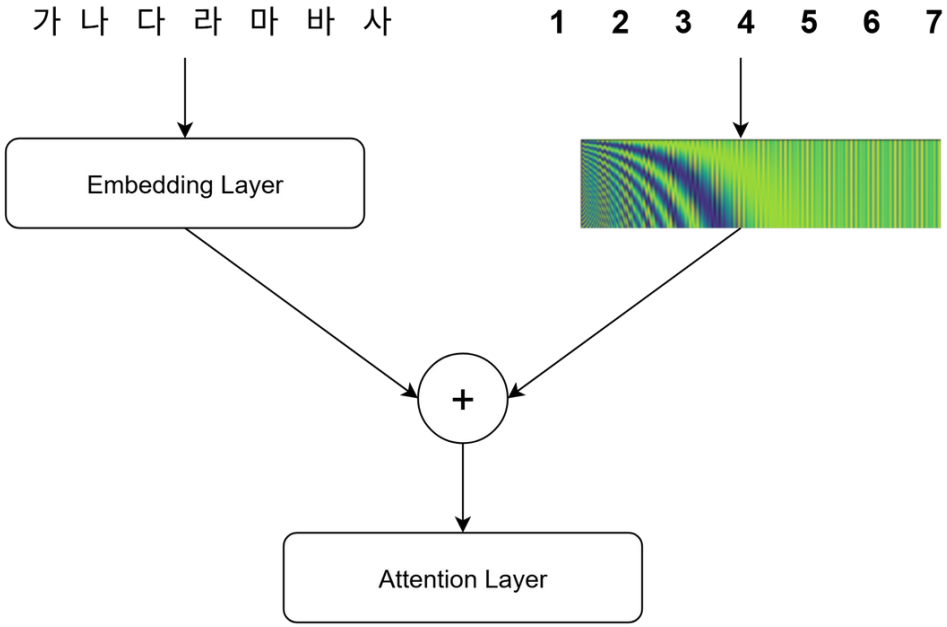
위치 인코딩

- 위치 인코딩 필요
  - Token에는 단어의 위치 정보가 없음
  - Tom ate the meat ≠ The meat ate Tom.
- 조건
  - 시퀀스 길이에 관계없이 동일한 위치
  - 의미적 유사성보다 위치적 유사성이 더 강조되지 않도록 크기가 제한



절대 위치 인코딩

- 단어의 위치를 절대값으로 표기

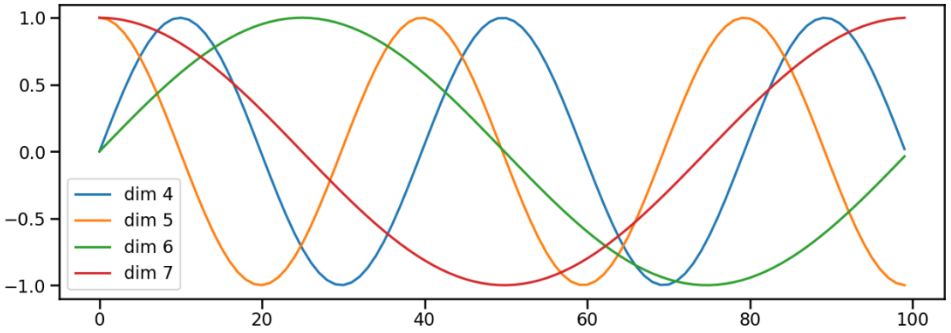


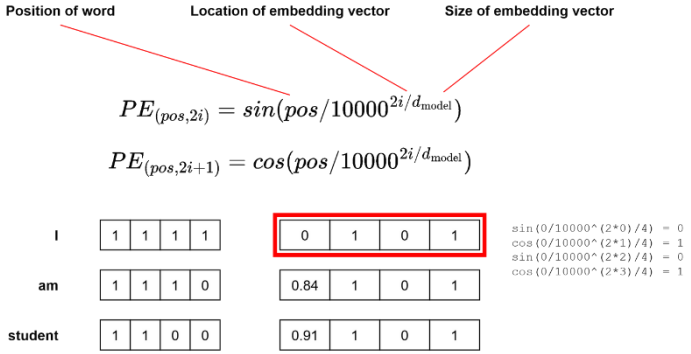
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

Sinusoidal Positional Encoding

- 절대 위치: 삼각함수
  - 낮은 차원: 주기가 짧아서 빠르게 변함(세밀한 위치 구분)
  - 높은 차원: 주기가 매우 길어서 문장이 길어져도 반복되지 않음
  - 결과: 수만 단어 길이의 문장에서도 유일한 위치 지문을 만들 수 있음

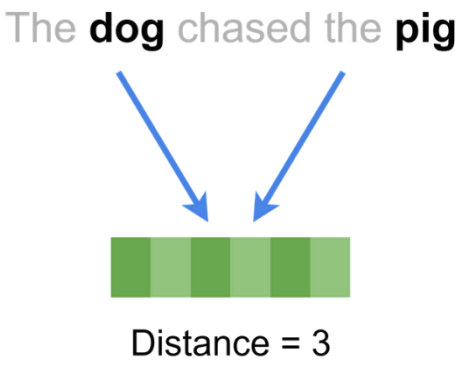


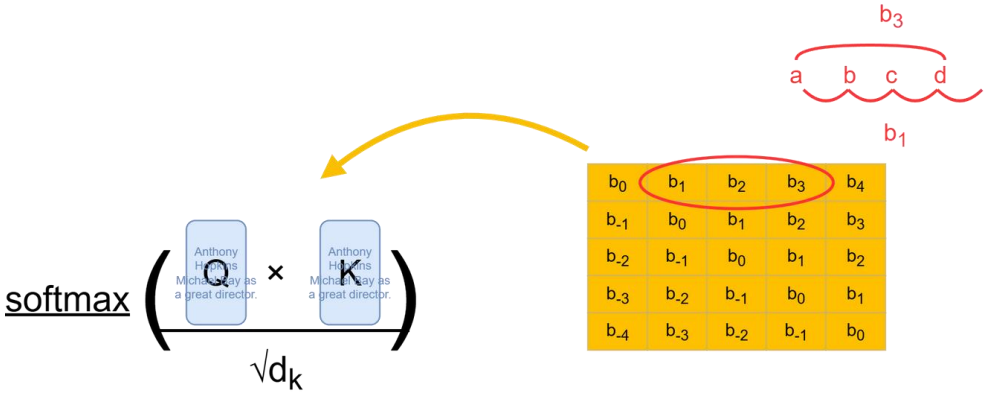


자연어 처리 트랜스포머 1강(Embedding, Positional Encoding)
Harvard_Transformer.ipynb - Colab

상대 위치 인코딩

- 단어와 단어 사이의 거리를 임베딩
  - 위치 편향을 bias로 더함
  - 위치 임베딩의 일관성 유지
- 단점
  - 속도가 느림: 위치를 계산해서 더해야 함, 매 스텝마다 값이 변해서 캐싱이 어려움

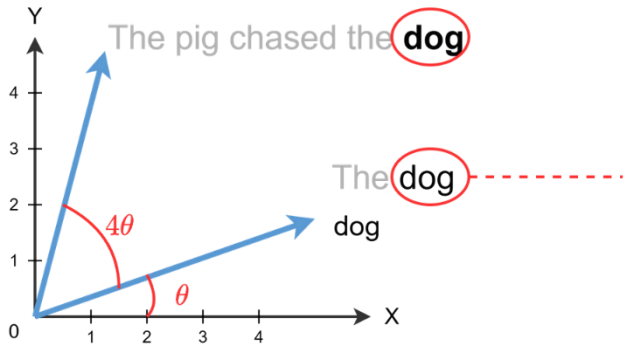


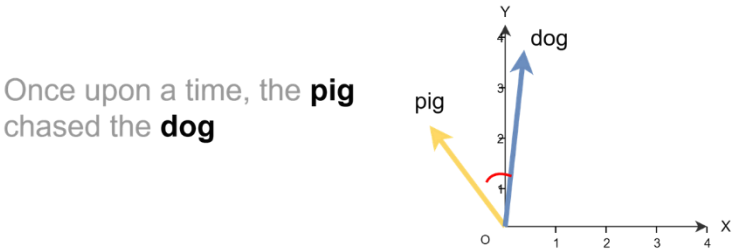


ROPE

ROPE(Rotary Positional Embeddings)

- 위치(m)와 각도(θ)에 비례하여 벡터를 회전(rotation)시키는 방식
- 상대적인 거리가 보존됨
- 효율적인 캐싱(KV cache)가 가능함
- 사인함수 임베딩보다 학습 속도가 빠름





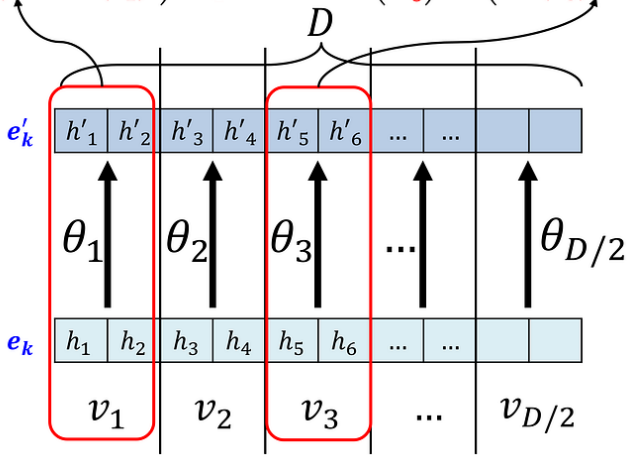
Rotary Positional Embeddings: Combining Absolute and Relative - YouTube

Rotary Positional Embeddings Explained | Transformer

ROPE

- 2개씩 짝지어서 각도를 다르게 회전함

$$\begin{pmatrix} h'_1 \\ h'_2 \end{pmatrix} = \begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \qquad \begin{pmatrix} h'_5 \\ h'_6 \end{pmatrix} = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) \end{pmatrix} \begin{pmatrix} h_5 \\ h_6 \end{pmatrix}$$



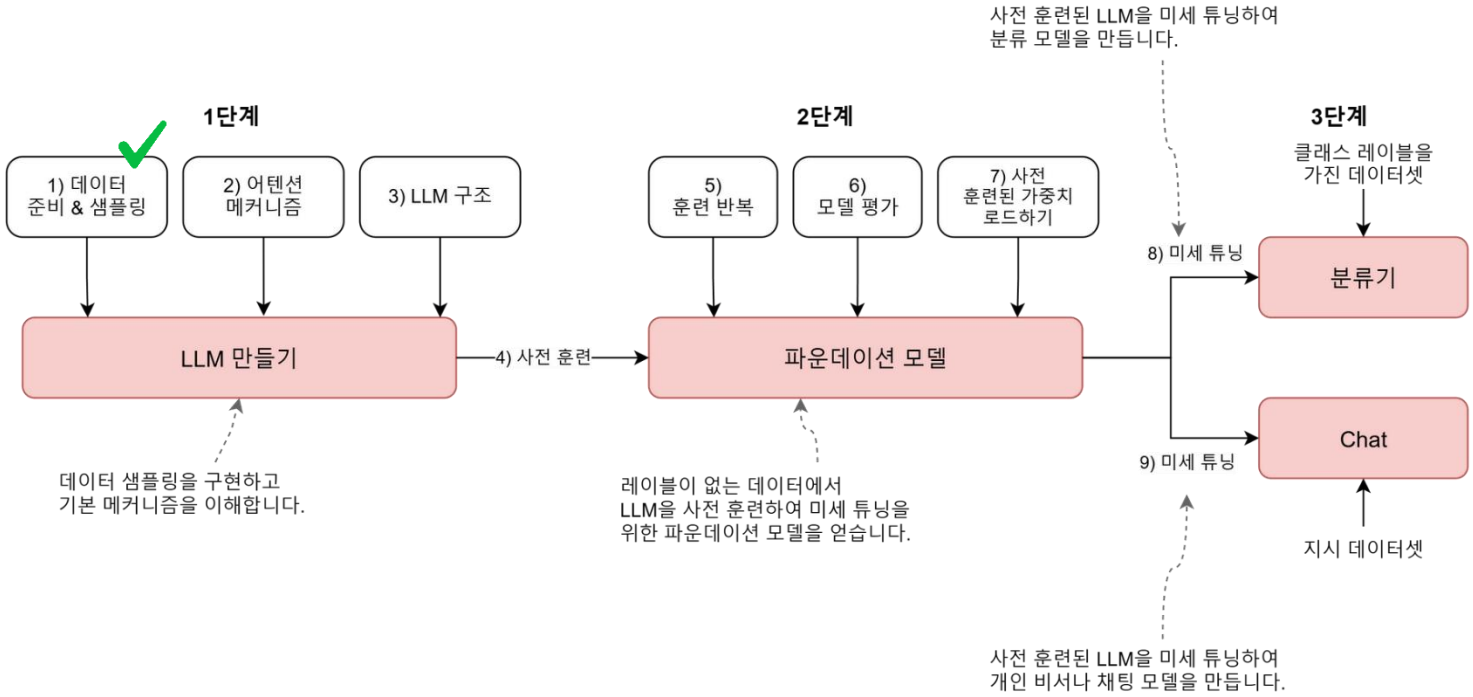
Rotary Positional Embeddings: Combining Absolute and Relative - YouTube

Rotary Positional Embeddings Explained | Transformer

학습

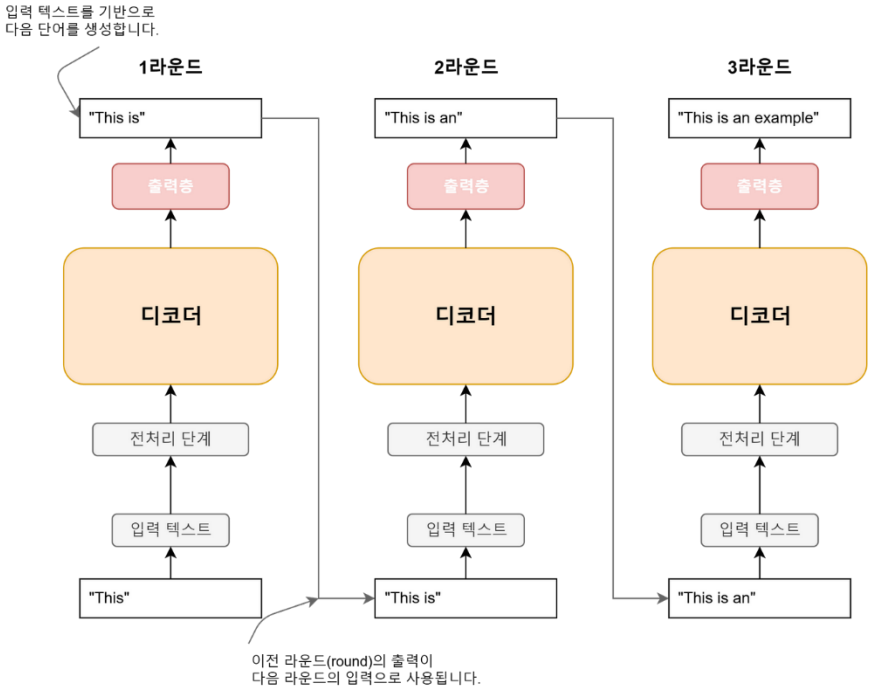
모델 학습

- 파운데이션 모델 -> Chat 모델



Text 생성

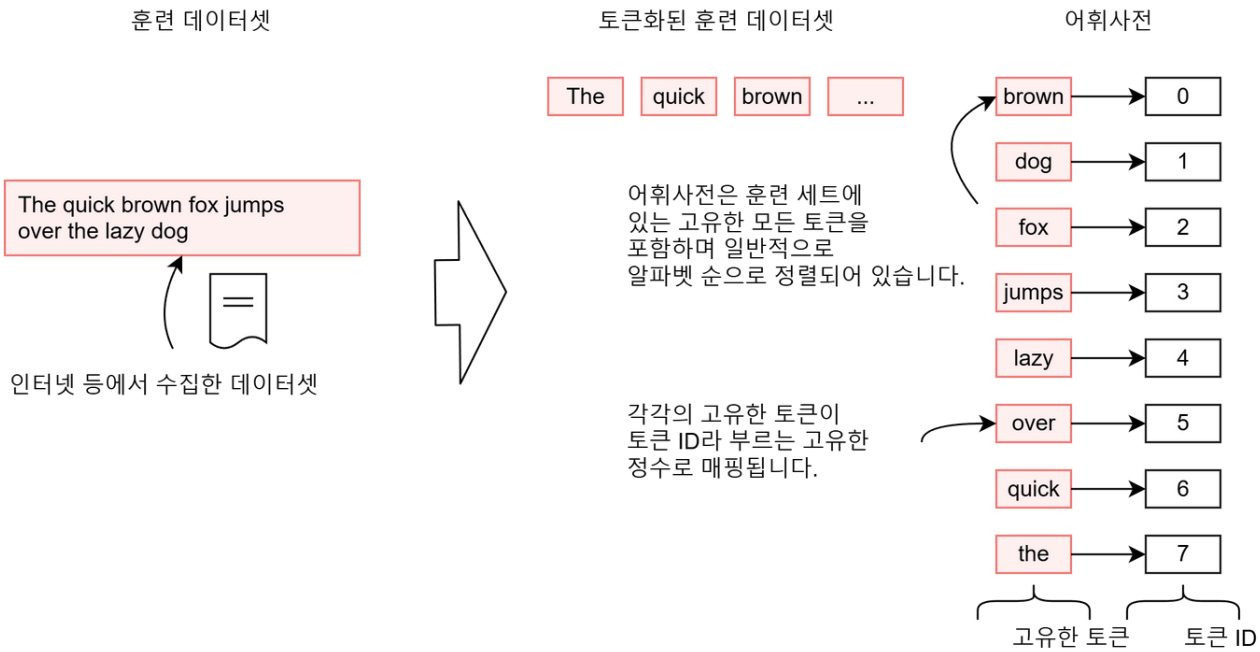
- 다음 단어를 예측: Autoregressive



The model is simply trained to predict the next **word**

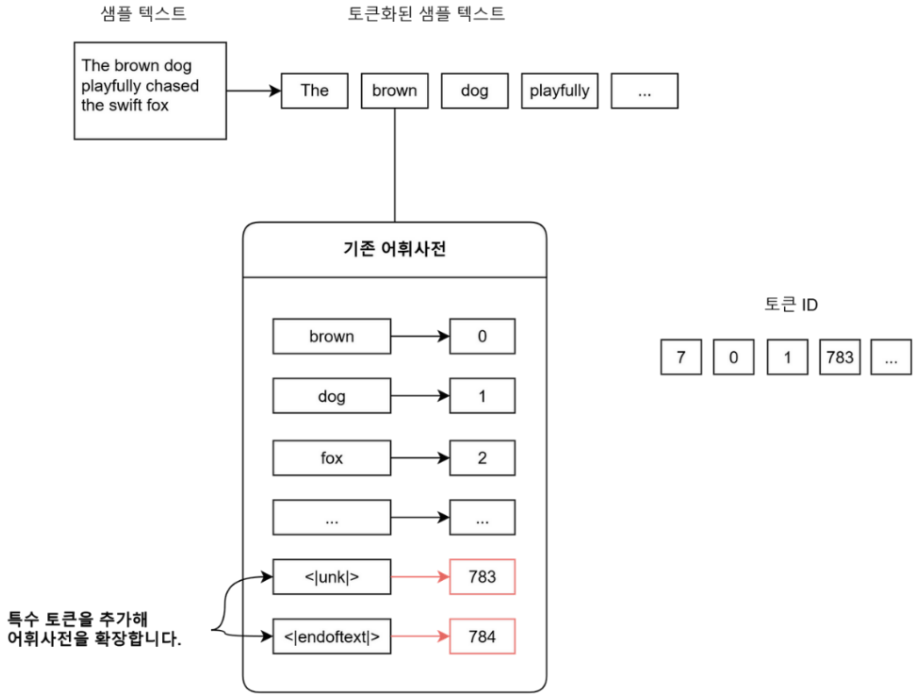
Token화

- 데이터셋을 Token으로 변경



Special Token

- 모델을 제어하고 입력·출력의 구조를 정의하는 제어 신호
  - 학습 대상에 포함(단 Padding Token은 제외)



Special Token

- 모델마다 조금씩 다를 수 있음

1. 문장의 시작과 끝
Sequence Control

- **<BOS>**
  문장의 시작
- **<EOS>**
  문장의 끝 (생성 중지)
- **<PAD>**
  길이 맞추기 (패딩)

2. 대화 구조 제어
Chat/Instruction

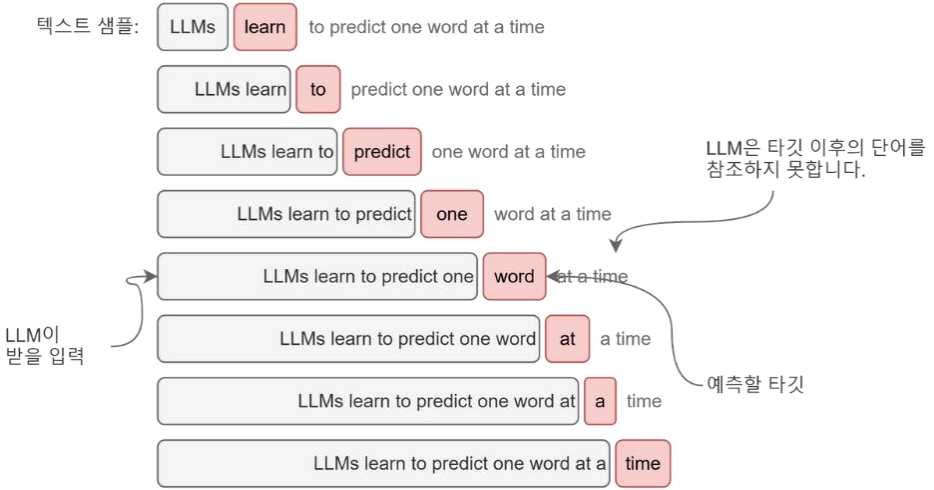
- **<|user|>**
  사용자 질문 시작
- **<|assistant|>**
  AI 답변 시작
- **<|system|>**
  시스템 프롬프트 (페르소나)

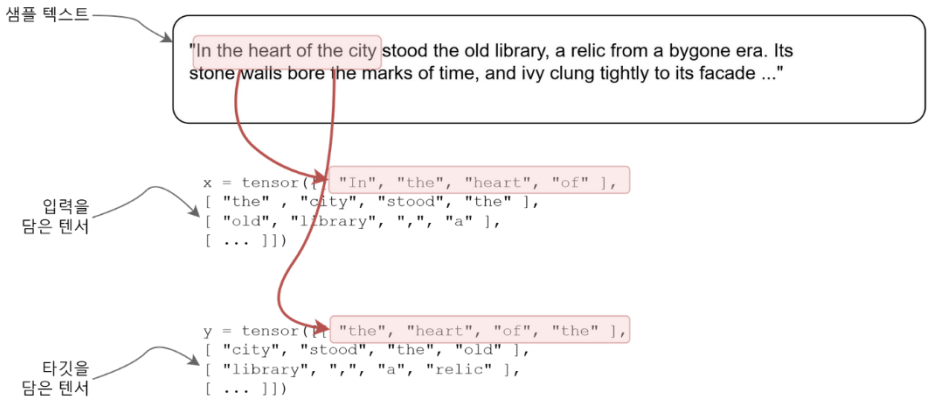
3. 기능 수행 및 사고
Task-Specific

- **<|thought|>**
  사고 과정 (CoT)
- **<|file_separator|>**
  파일 경계 구분
- **<|tool_call|>**
  외부 도구 호출

데이터셋

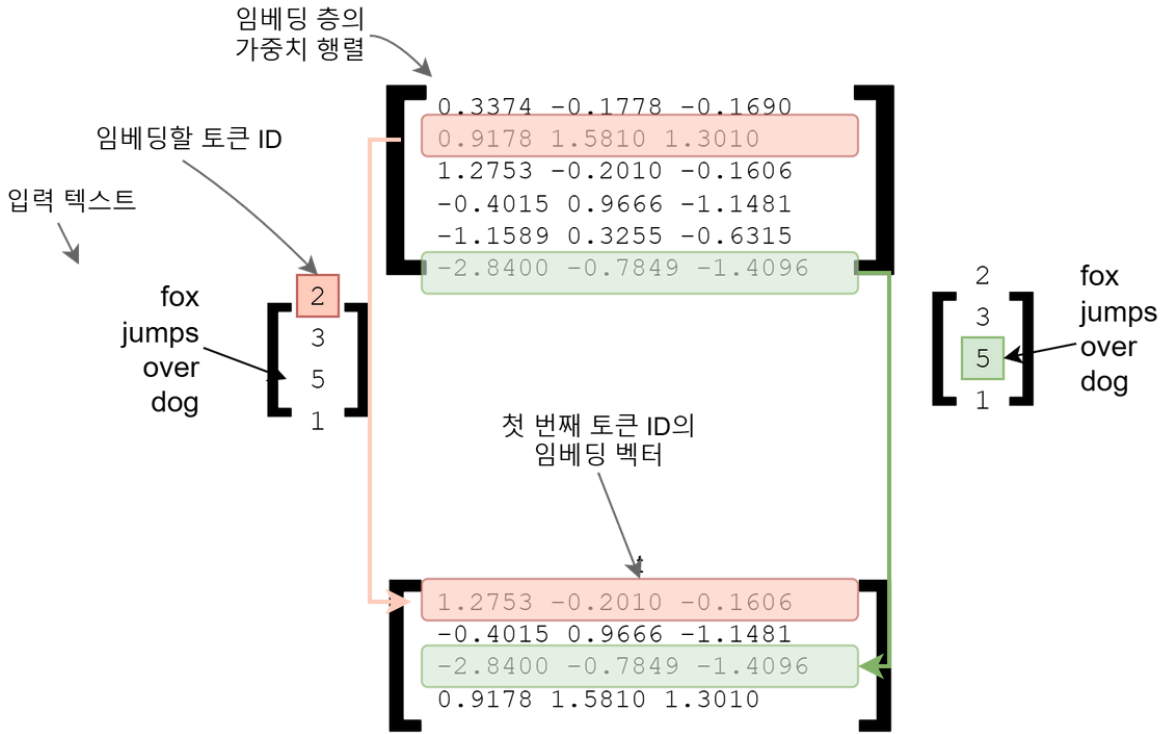
- Self-Supervised : Label 없이 생성





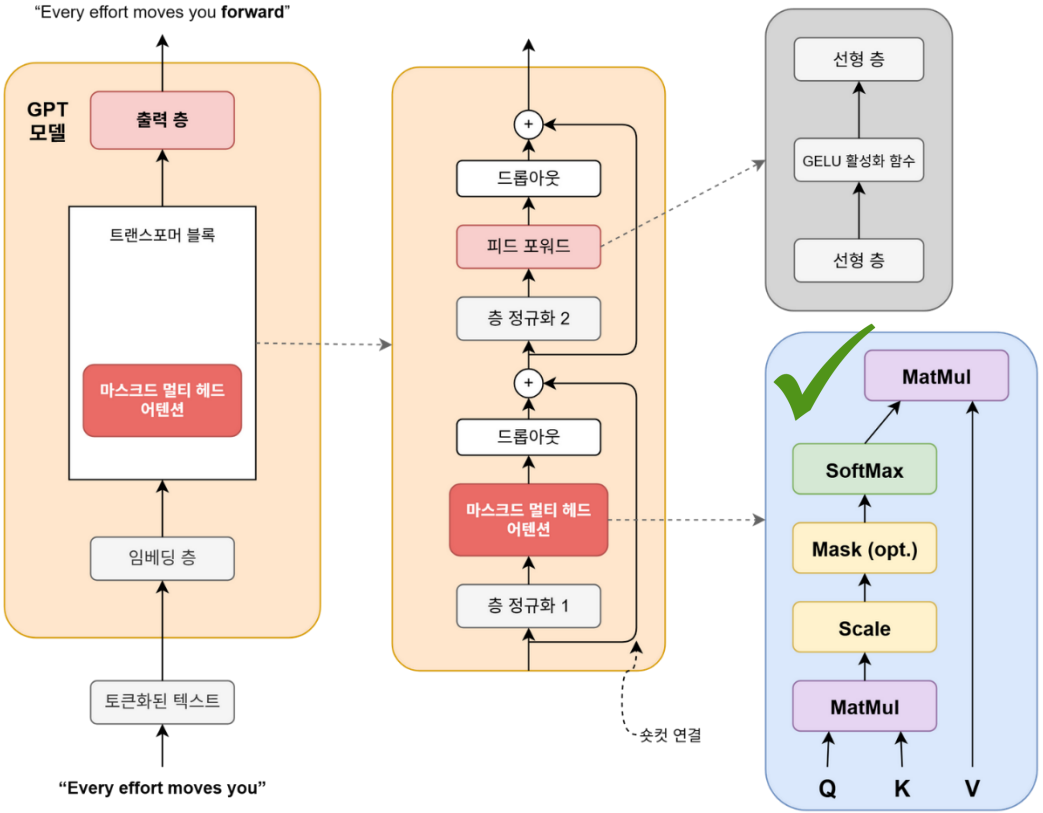
Token 임베딩

- Token id를 연속 벡터 공간의 점으로 맵핑

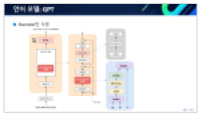


Chapter_2_Exercise_Dataset.ipynb

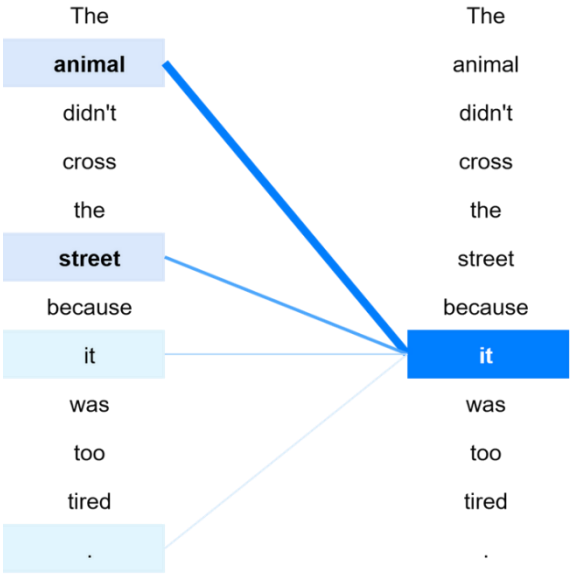
GPT: Atttention



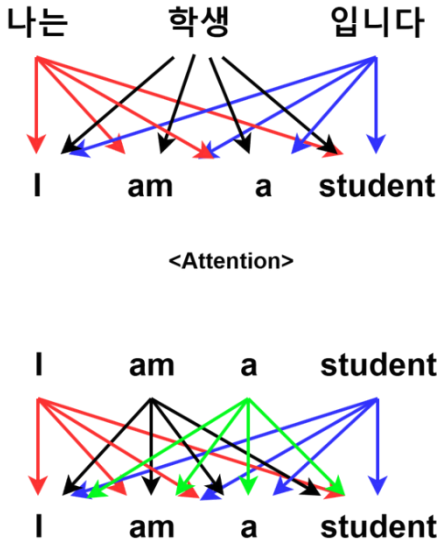
Attention



- 문맥을 파악
  - 연관되어 있으면 값이 크게 나타남



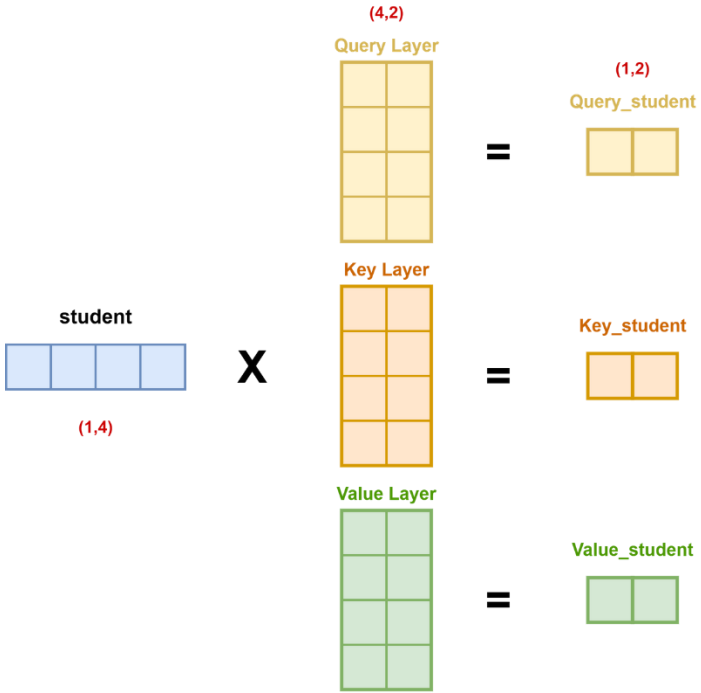
글의 문맥 이해 (Context)



<Self-Attention>

Attention

- Embeding에서 Weight를 곱해서 Q,K,V를 구함

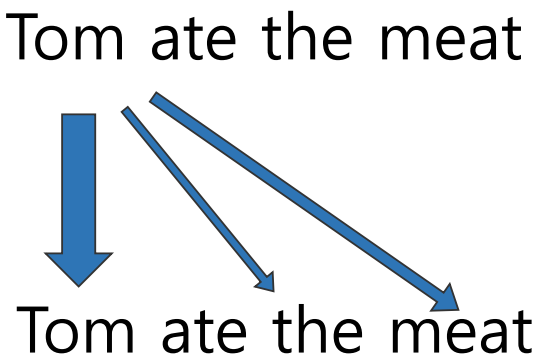


Q,K,V로 분리하지 않으면
- 질문을 던지는 주체(Query)
- 질문에 답하는 대상(Key)
- 전달할 정보(Value)

구분이 되지 않음

Attention

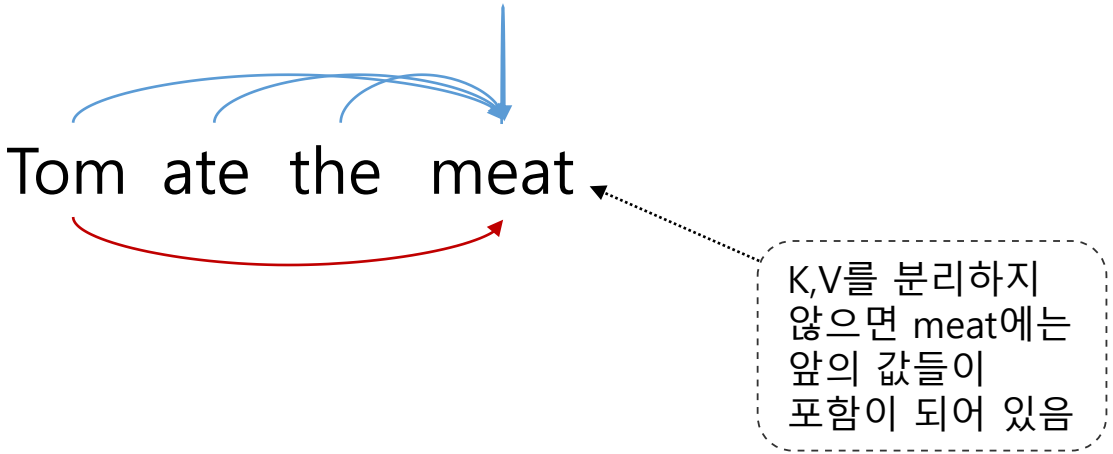
- Q,K를 분리하지 않으면 Attention이 제대로 되지 않음
- Q,K를 분리하지 않으면
  - 항상 자기 자신을 우선 주목함
  - 순서를 변경해도 동일한 내적이 나옴



Tom@meat = meat@Tom

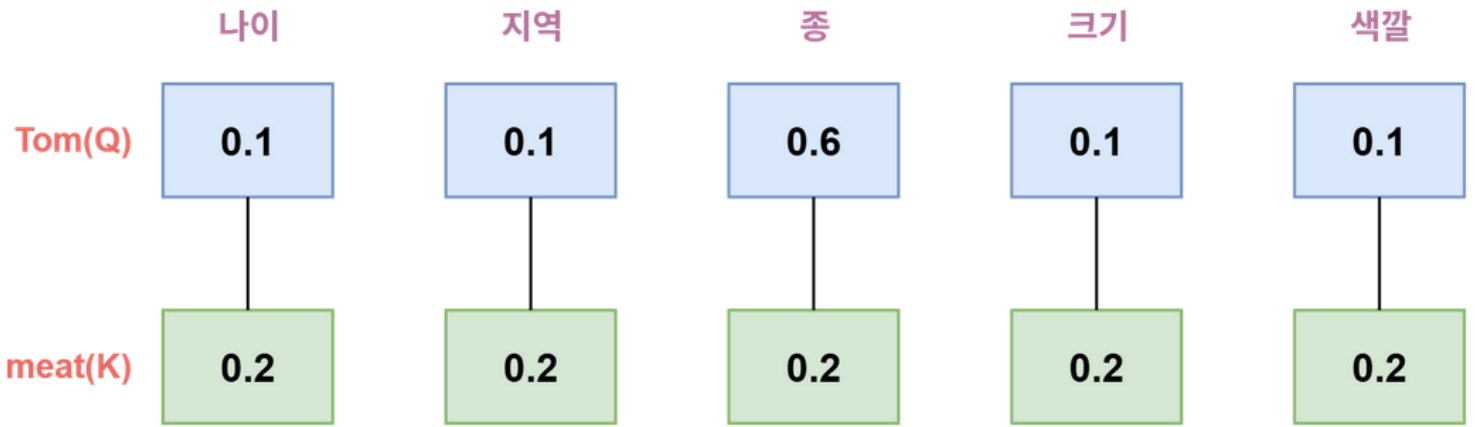
Attention

- K,V를 분리하지 않으면 Key에 다른 값이 섞일 수 있음
  - 문맥 벡터와 key가 분리해야 함



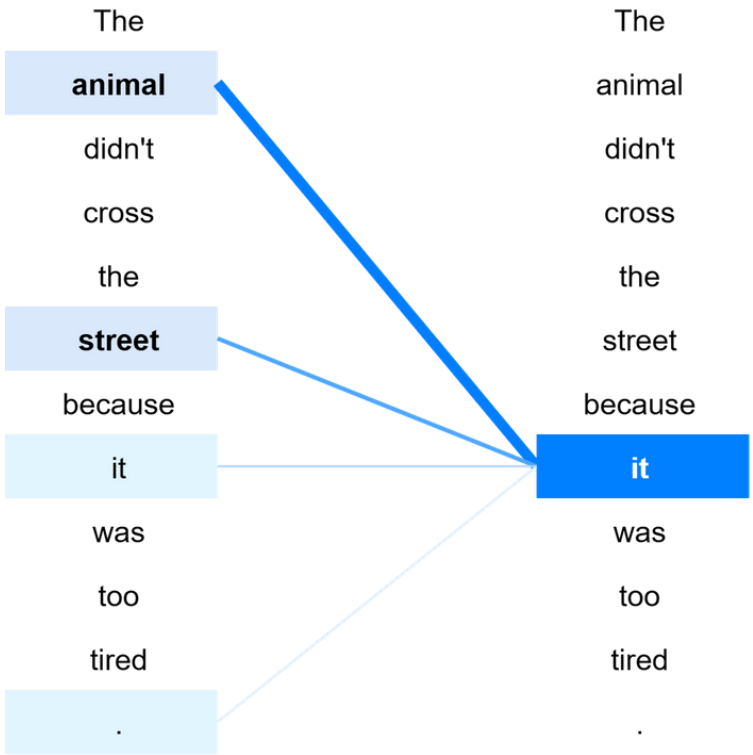
Attention

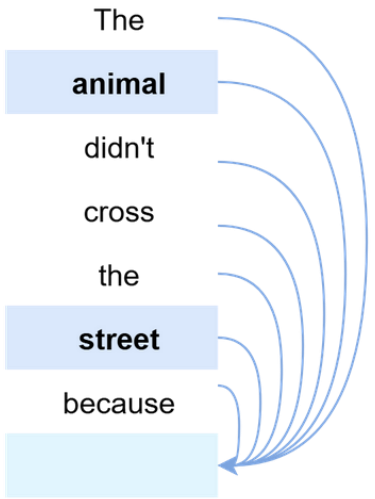
- Query는 값을 이용하여 질문을 한다.



Attention

- 다음에 올 단어를 예측하는 것을 학습함으로써 Attention Score가 의미를 가짐

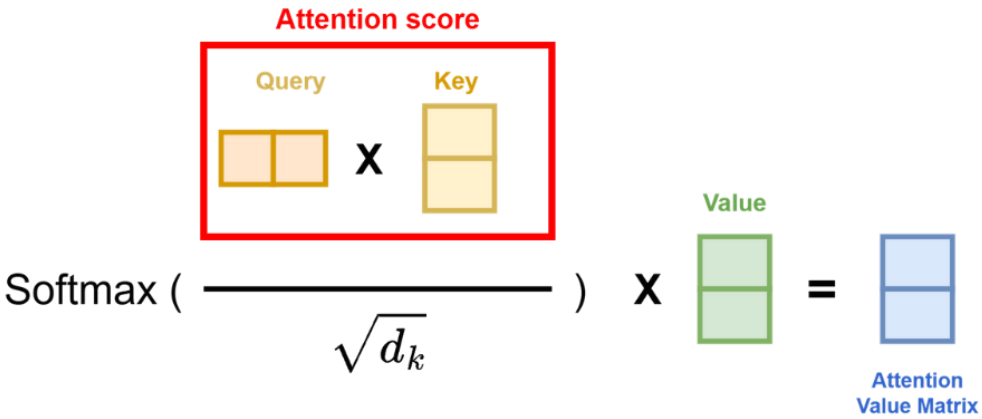




Attention Score

- 연관되어 있으면 값이 크게 나타남

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Attention Map

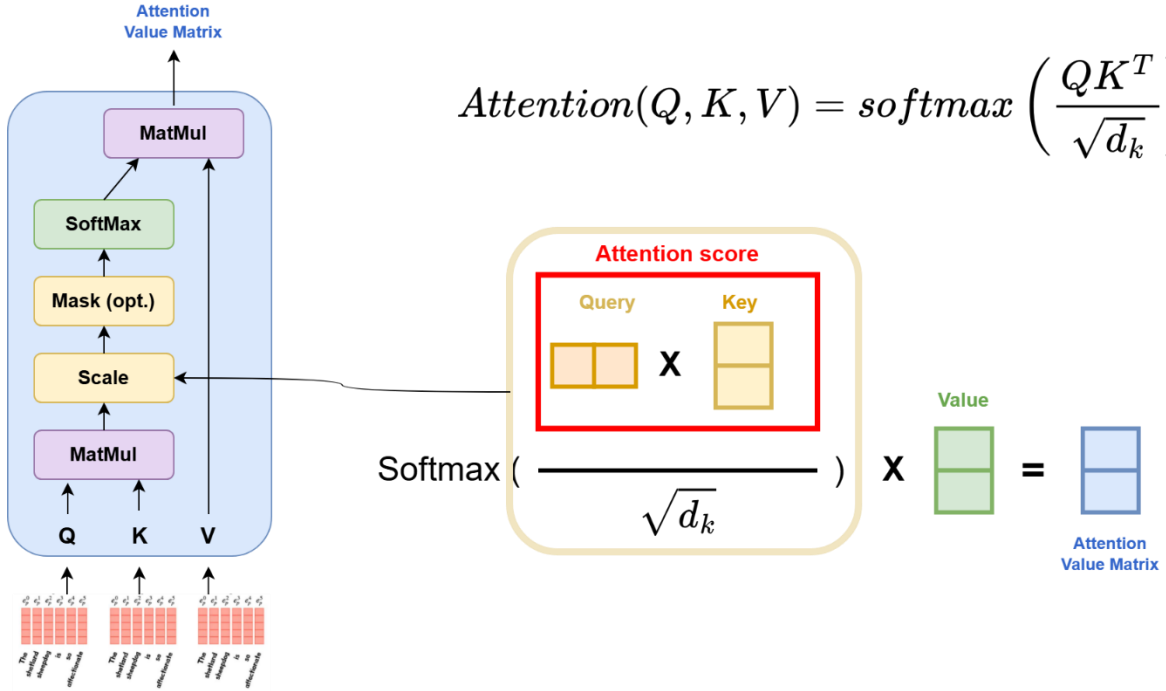
- Attention Score를 시각화 한 행렬, 어떤 단어와 어떤 단어 연관되어 있는지 시각화

	I	am	a	student
I	1.0	0.5	0.3	1.0
am	0.5	1.0	0.4	0.2
a	0.3	0.4	1.0	0.1
student	1.0	0.2	0.1	1.0

I가 Student와 밀접한 연관이 있다는 것을 보여줌

Scale

- d_k 로 나누어 분산을 다시 1로 맞춤
  - Attention 연산을 하면 차원 수(d_k)가 커질수록 결과값(Dot-product)이 매우 커지는 경향

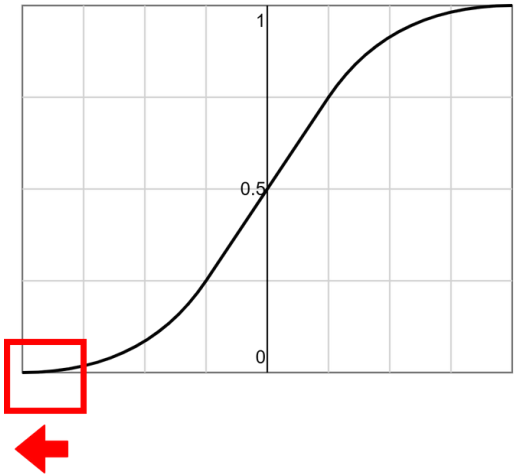


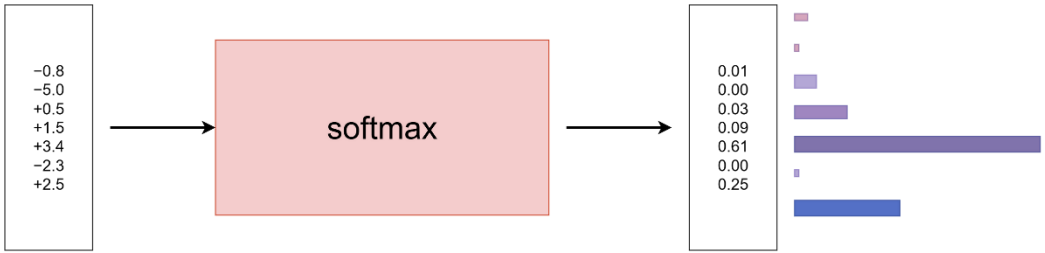
$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Softmax

- 어떤 숫자의 나열을 확률분포로 변환: 지수 함수를 사용

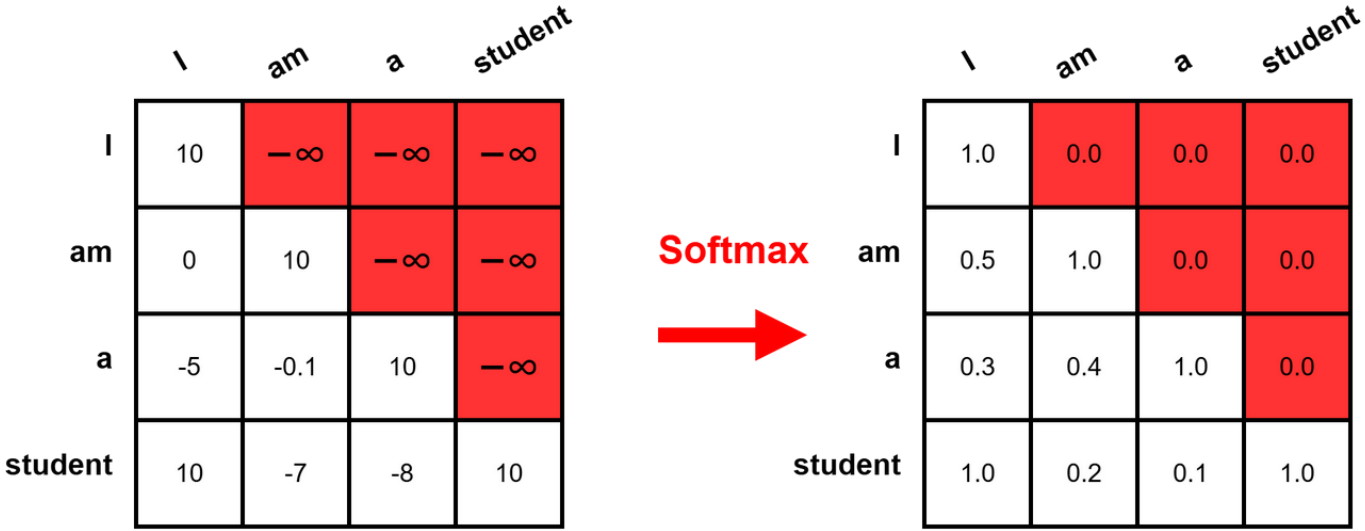
$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_i}}$$





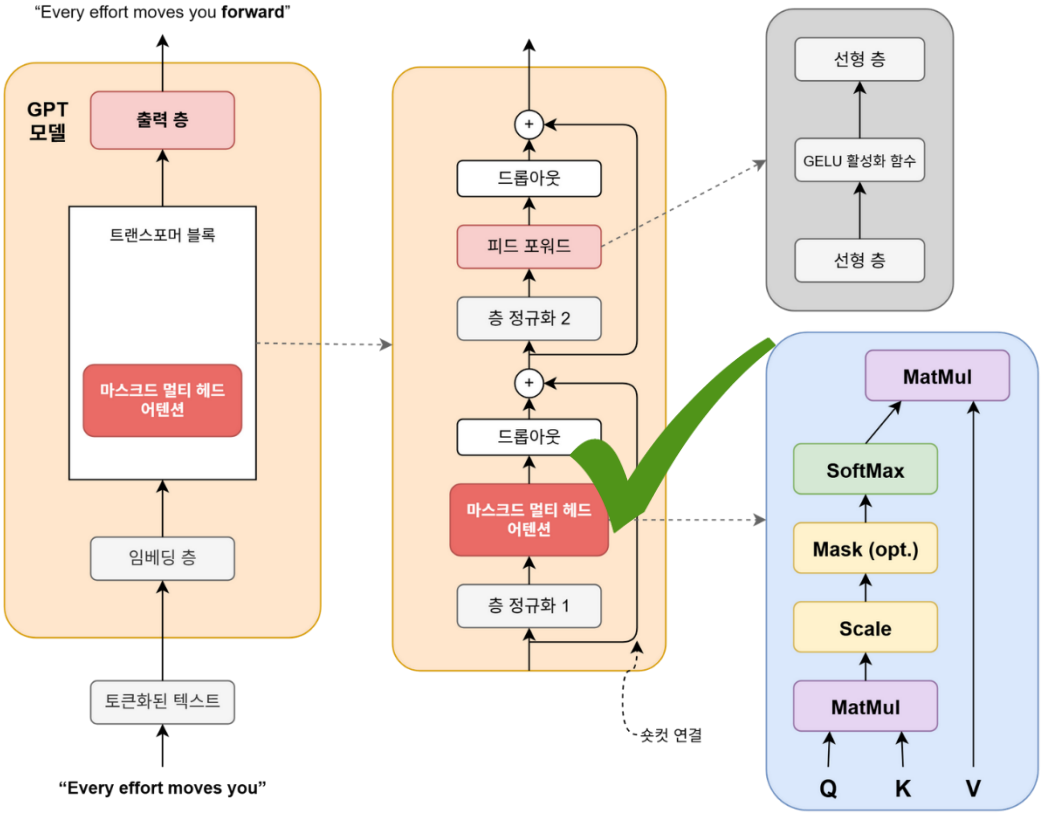
Masking

- 다음 단어가 학습되지 않도록 가림, $-\infty$로 치환 후 softmax



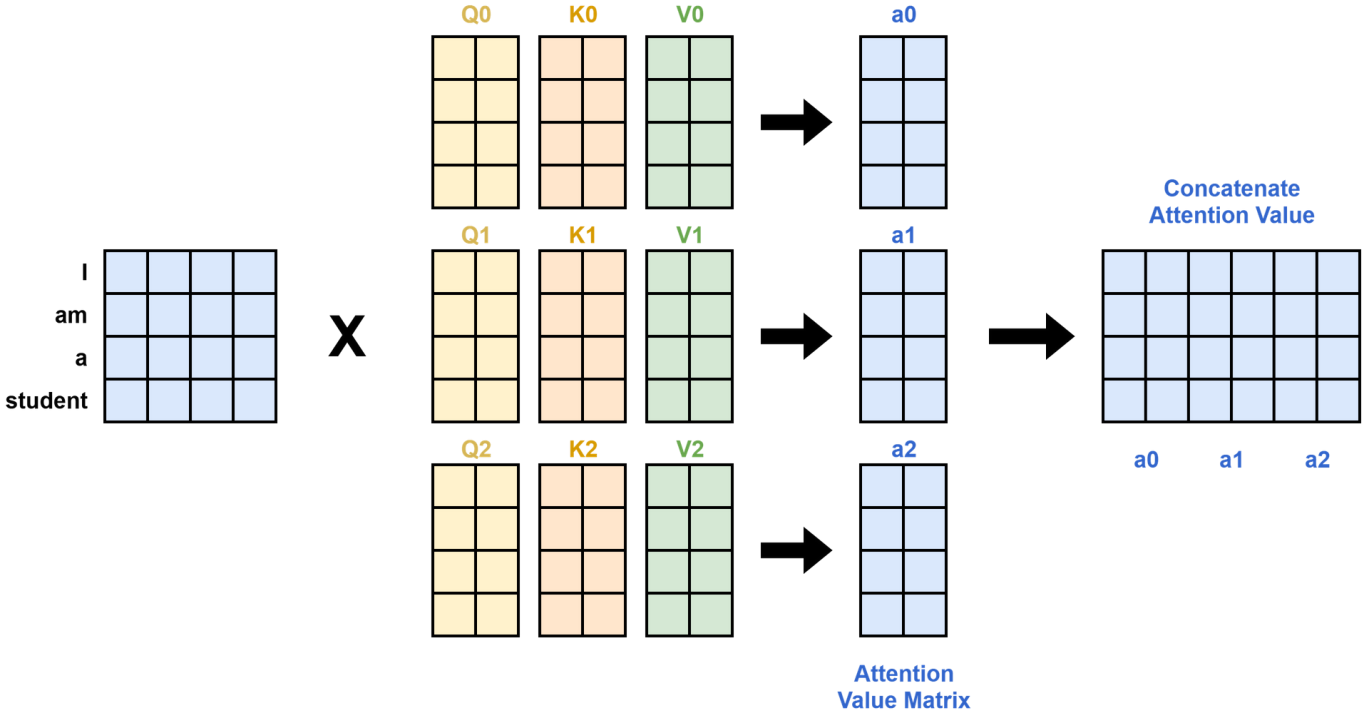
GPT: layer들

- Decoder만 사용



Multi-head Attention

- 각각의 Head는 다른 관점에서 문맥을 파악함

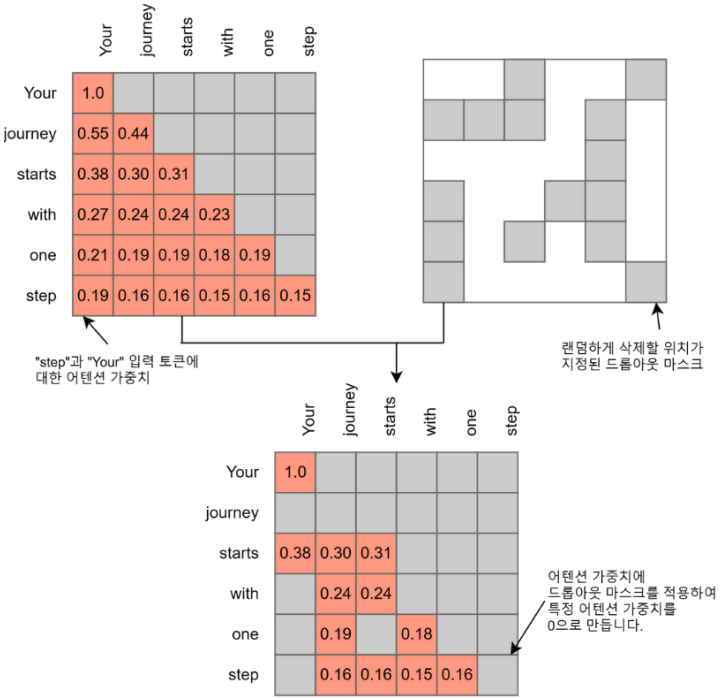


Chapter_3_Excercise_Attention.ipynb

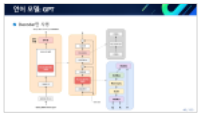
Dropout

Overfitting 방지

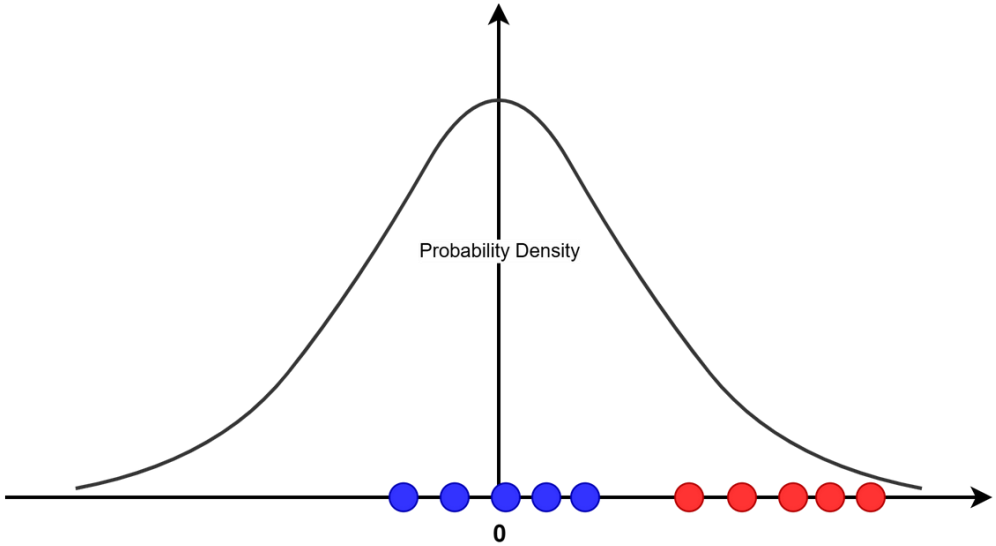
- Softmax 이후에 랜덤으로 0으로 마스킹함, 이번에 학습할 때는 여기만 집중해
- 학습 속도가 느리고, 데이터가 많아서 학습에는 잘 사용하지 않음



정규화

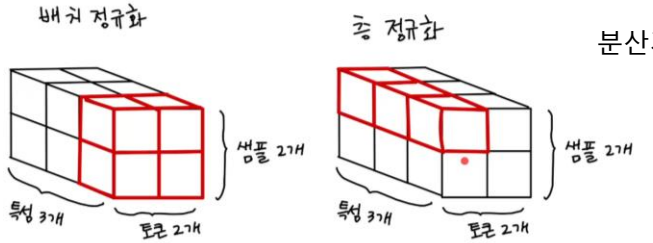


- 평균이 0이고 표준편차가 1이 되도록 변환
  - 학습을 안정화 시킴
  - 과도한 정규화는 학습 내용을 사라지게 할 수 있음



정규화: Layer Normalization

- LLM에서는 주로 Layer Normalization을 사용함
  - 학습된 내용을 보존해서 필요한 범위만 정규화 하도록 감마(γ)와 베타(β)는 학습
  - 학습하는 이유는 정규화가 항상 최선은 아니기 때문

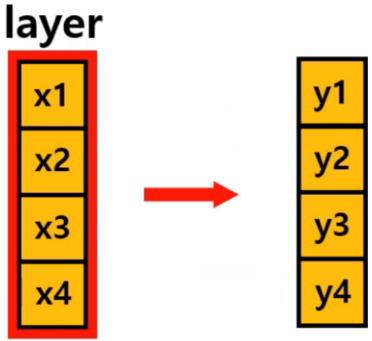


평균계산 $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$

분산계산 $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$

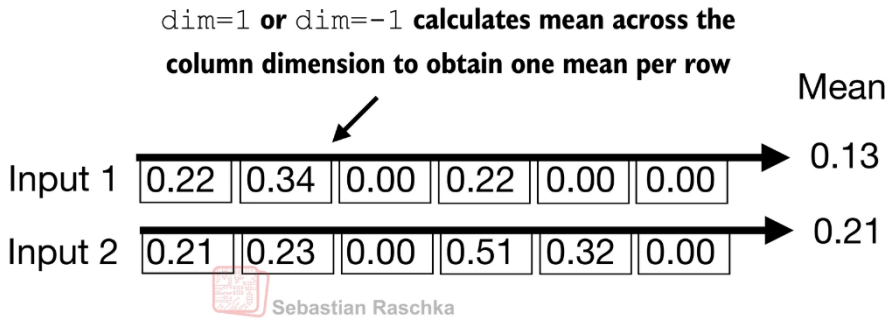
정규화 $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$

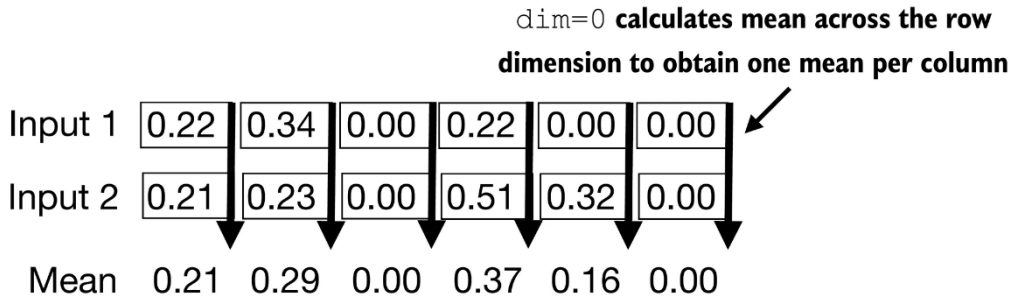
스케일 및 시프트 $y_i = \gamma\hat{x}_i + \beta$



정규화: Normalization

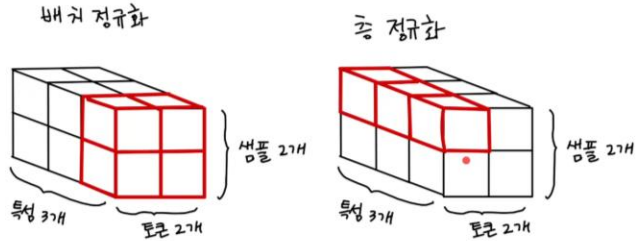
- 행렬에서 Layer, Batch Normalization





정규화: RMS Normalization

- 비용이 많이 드는 평균을 제외
  - 정규화의 핵심 효과는 평균을 0으로 맞추는 것보다 데이터의 크기(Scale)를 일정하게 유지하는 것에서 나옴



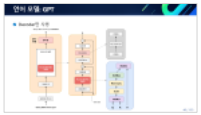
$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

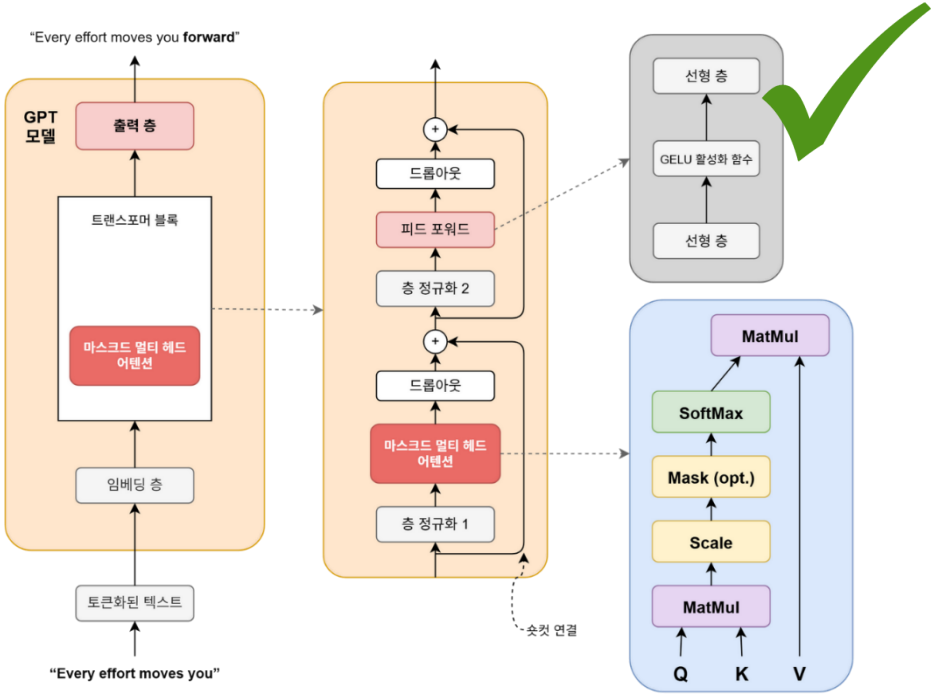
$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma\hat{x}_i + \beta$$

Feed Forward Network

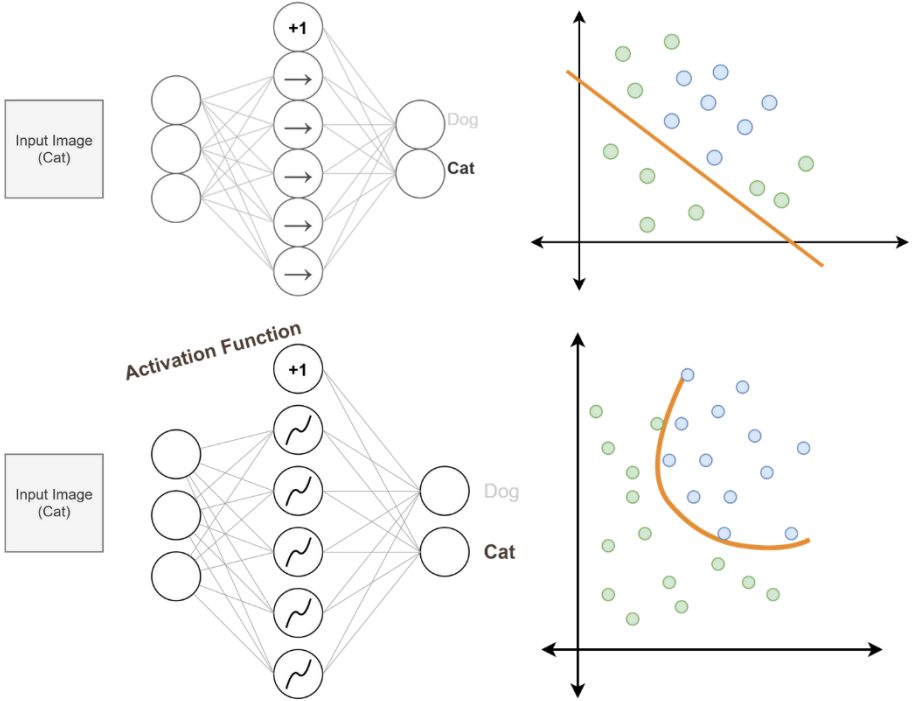


- 지식의 저장소
  - 수집된 정보를 해석하고 처리하여 저장된 지식과 연결



활성화함수

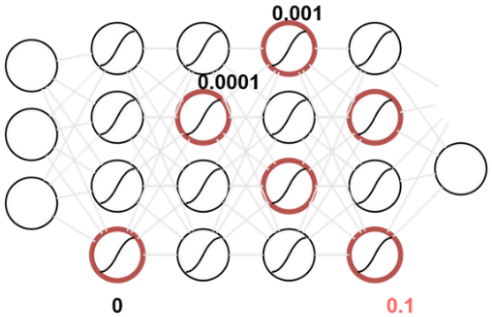
- 비선형성을 부여하여 복잡한 패턴을 학습이 가능하게 함



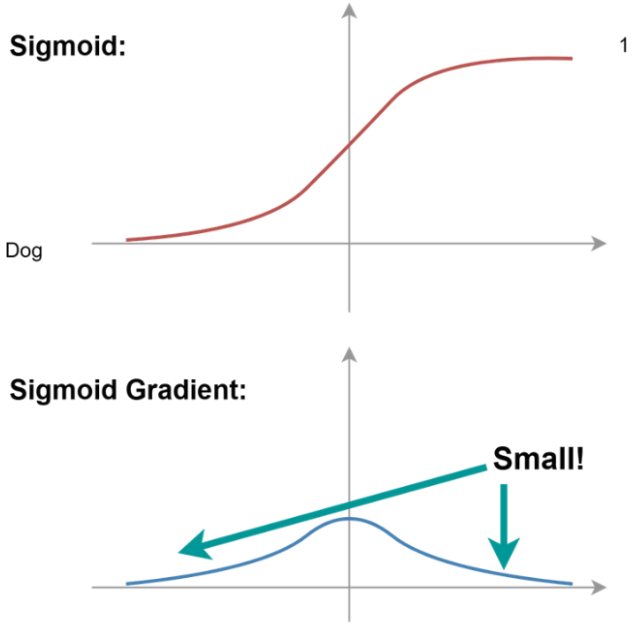
Activation Functions - EXPLAINED!

활성화 함수: Sigmoid

- 복잡한 패턴 학습 가능
- Layer가 많아질수록 Vanishing Gradient Problem 발생



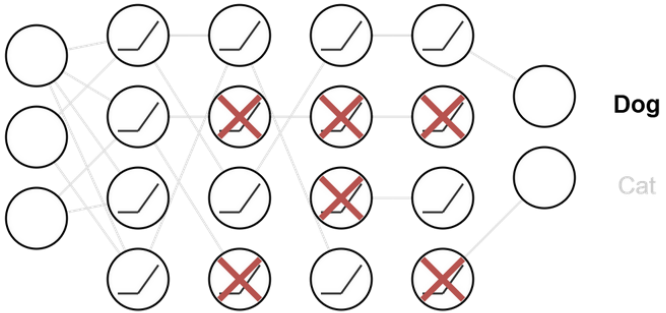
Vanishing Gradient Problem



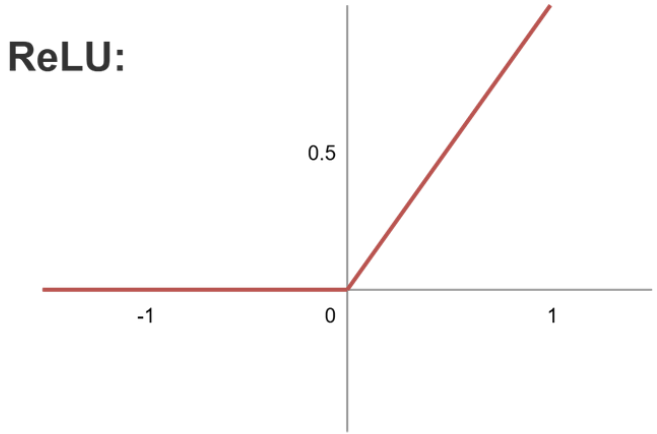
Activation Functions - EXPLAINED!

활성화 함수: ReLU

- Layer을 깊게 쌓을 수 있게 됨
- 비활성화되는 뉴런이 많아짐



Dying ReLU Problem



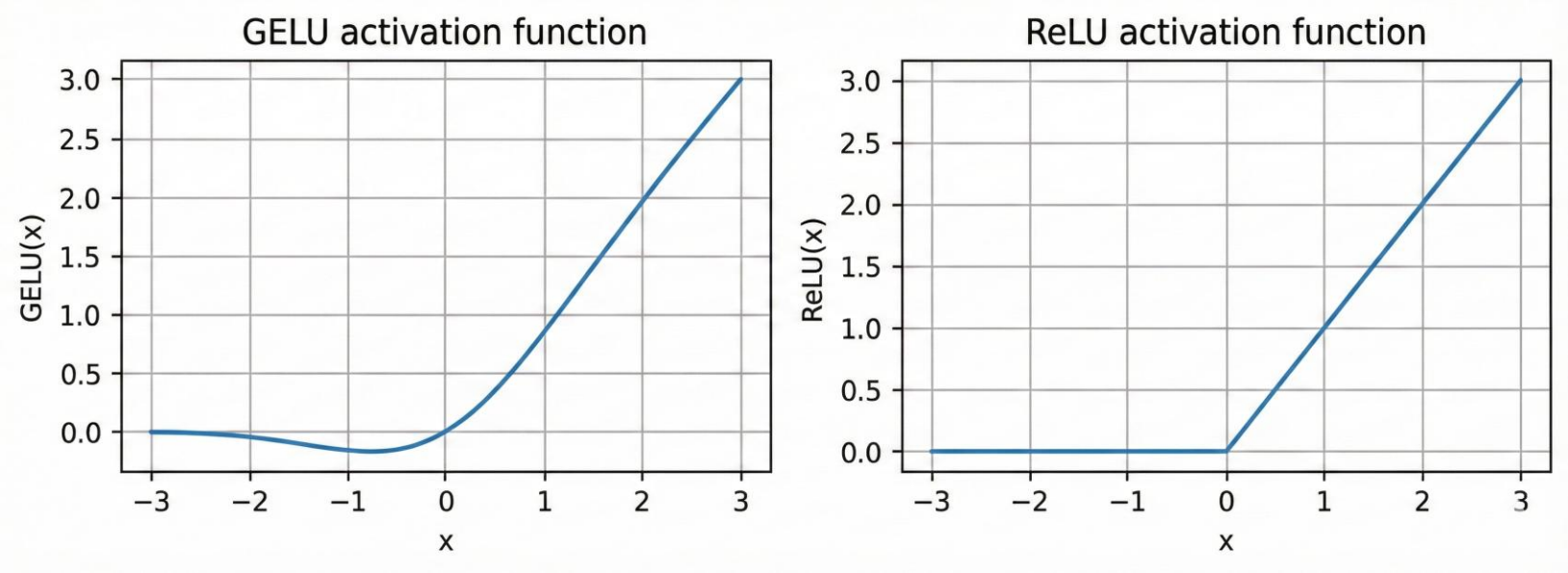
ReLU(Wx + b) = 0
if max(Wx + b, 0) = 0
or Wx + b < 0 or b < -Wx

This happens with very negative bias when W,x > 0

Activation Functions - EXPLAINED!

GELU

- 좀 더 안정화된 학습이 가능

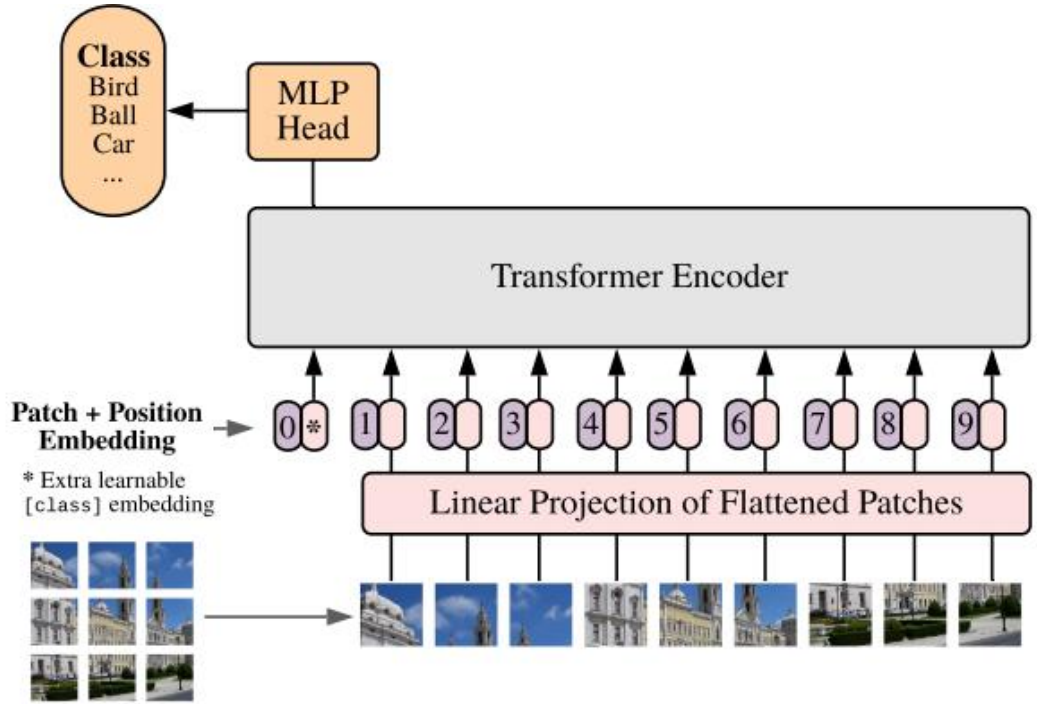


$$\mathrm{GELU}(x) \approx 0.5\,x\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}\left(x+0.044715\,x^3\right)\right)\right)$$

$$\mathrm{ReLU}(x) = \max(0, x)$$

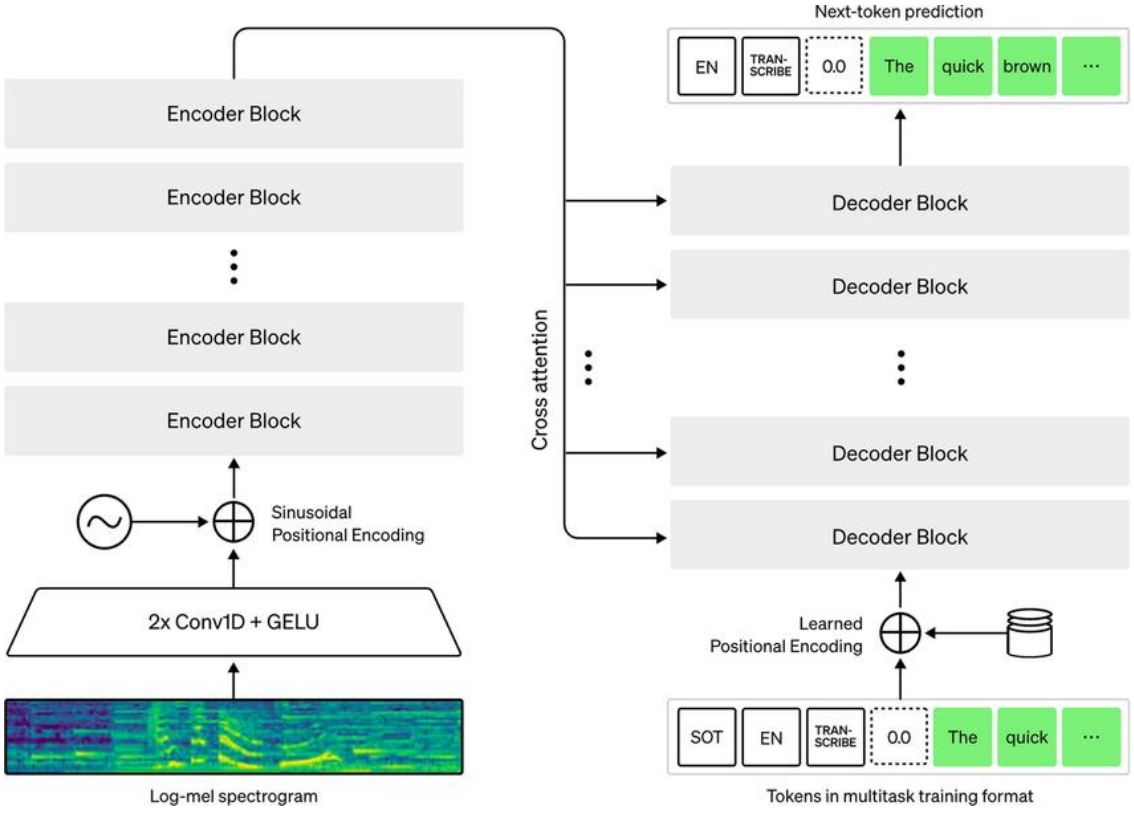
Vision Transformer

- 이미지 인식에 Transformer Encoder 사용



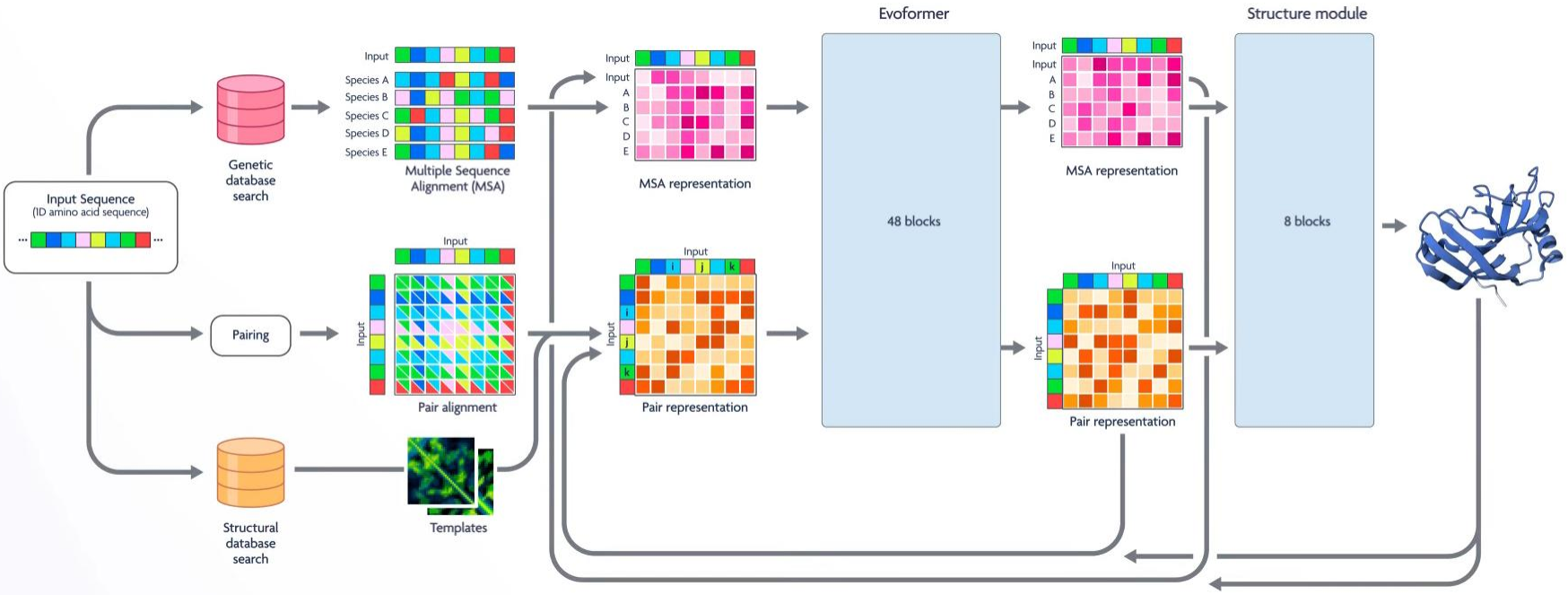
Whisper

- 음성 인식의 Encoder와 Decoder에 사용

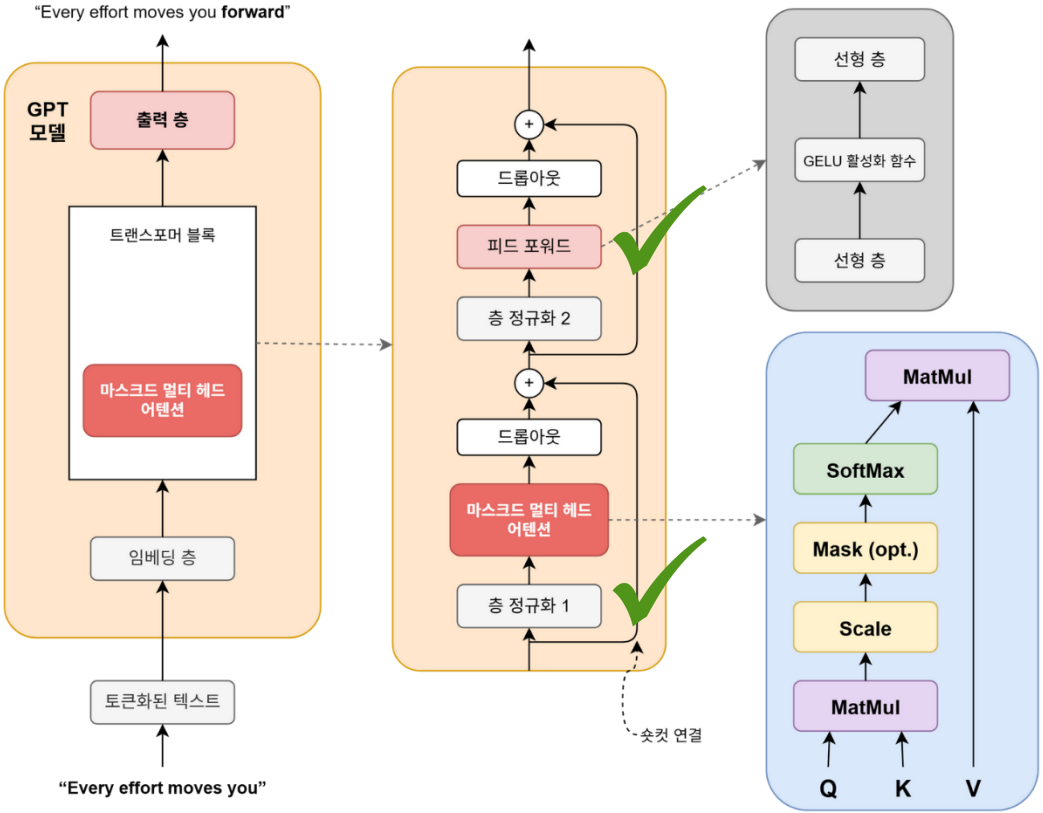


AlphaFold

- 아미노산과 아미노산의 관계를 파악하는데 Transformer 사용

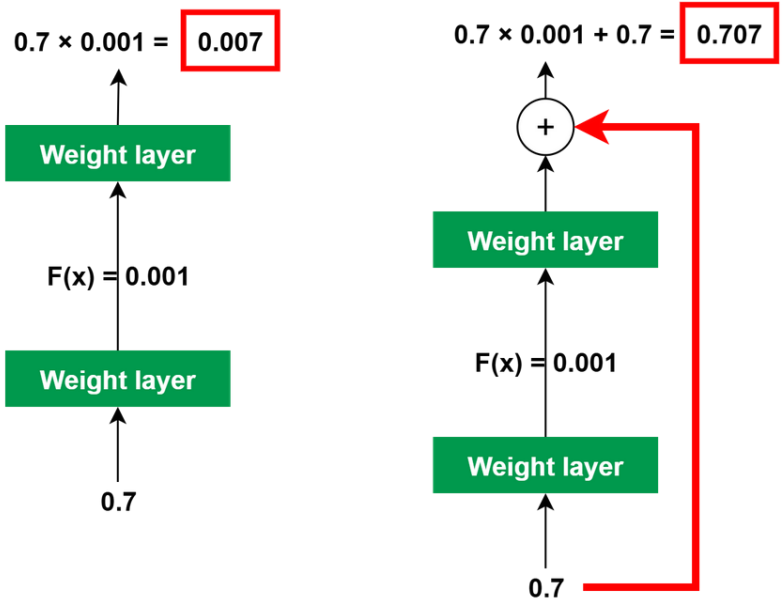


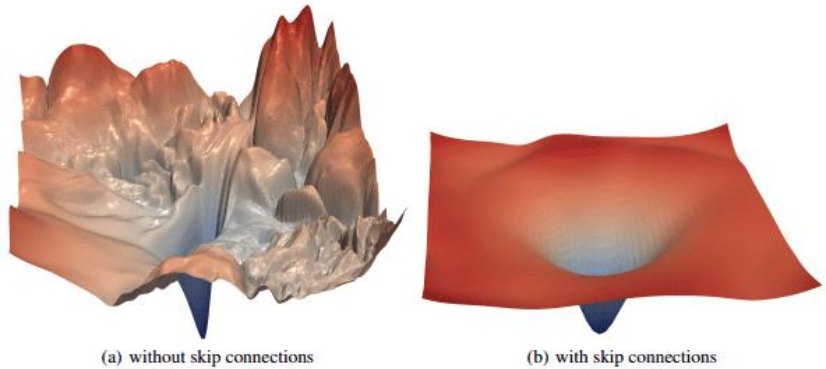
언어 모델: GPT


"Every effort moves you forward"
GPT 모델
출력 층
트랜스포머 블록
마스크드 멀티 헤드 어텐션
임베딩 층
토큰화된 텍스트
"Every effort moves you"
드롭아웃
피드 포워드
층 정규화 2
드롭아웃
마스크드 멀티 헤드 어텐션
층 정규화 1
숏컷 연결
선형 층
GELU 활성화 함수
선형 층
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

Residual connection

- 안정된 학습
  - 엄청 깊다면 값의 변화가 크지 않을 것이다. 변화량만 학습하면 되니까 학습이 쉽다





Intuitive Explanation of Skip Connections in Deep Learning | AI Summer

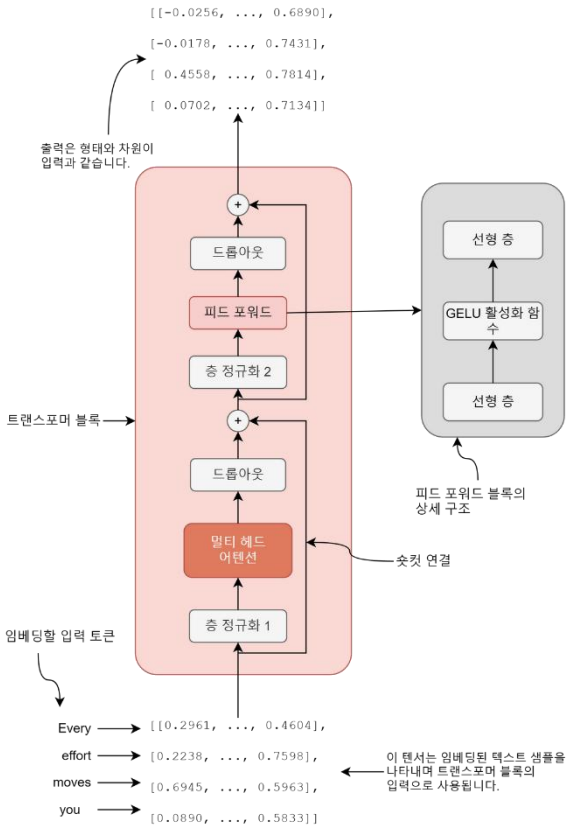
자연어 처리 트랜스포머 3강(Residual Connection, Layer Normalization)

[Legend 13] ResNet의 Skip-connection 제대로 이해하기!

GPT 모델 입출력

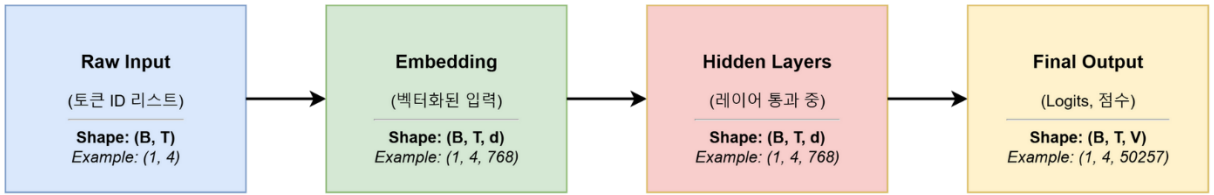
GPT 모델 입출력

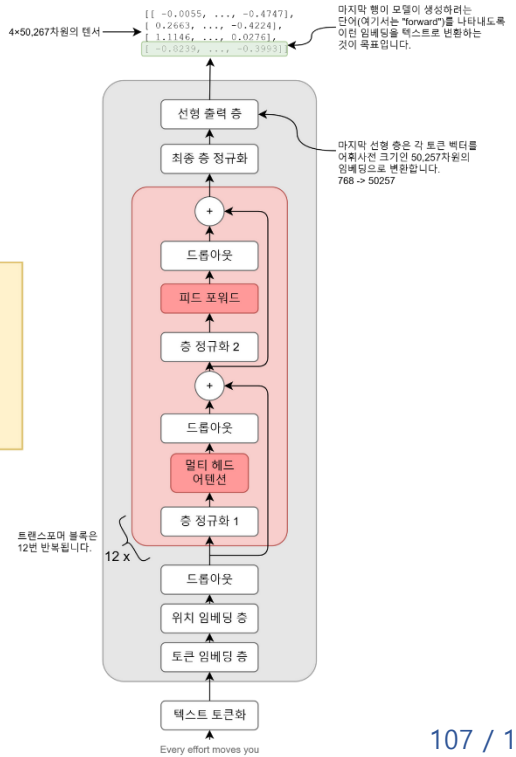
- Tramsformers 모듈의 입출력의 차원이 동일
  - 모듈을 계속 쌓기 위함
  - Residual connection 가능하도록 하기 위함



GPT 모델 입출력

- B(배치): 문장 1개 입력, T(길이): Token 4개, d (벡터 크기): 768, V (사전 크기): 50,257





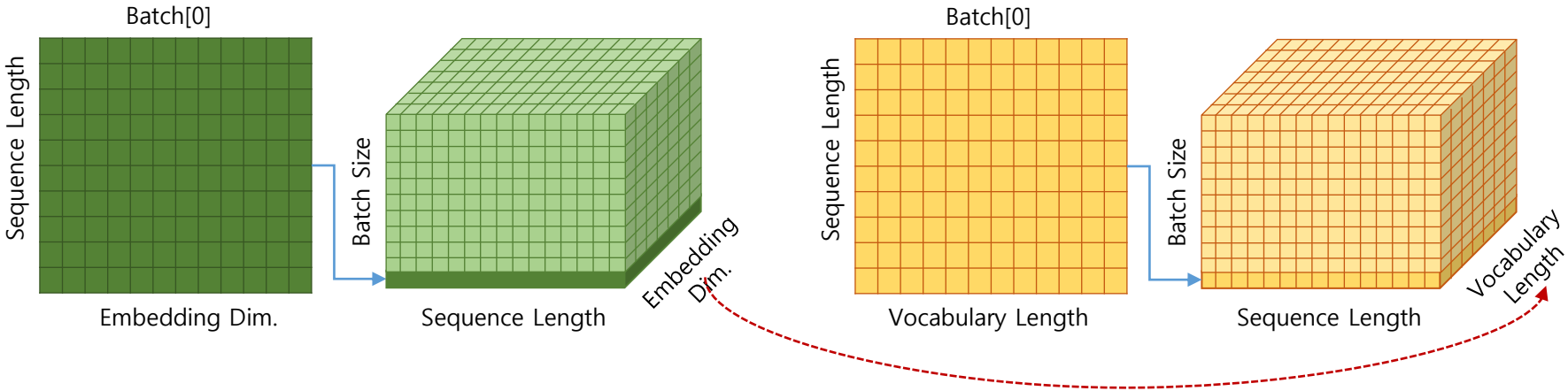
GPT 모델 입출력

Source Input Tensor

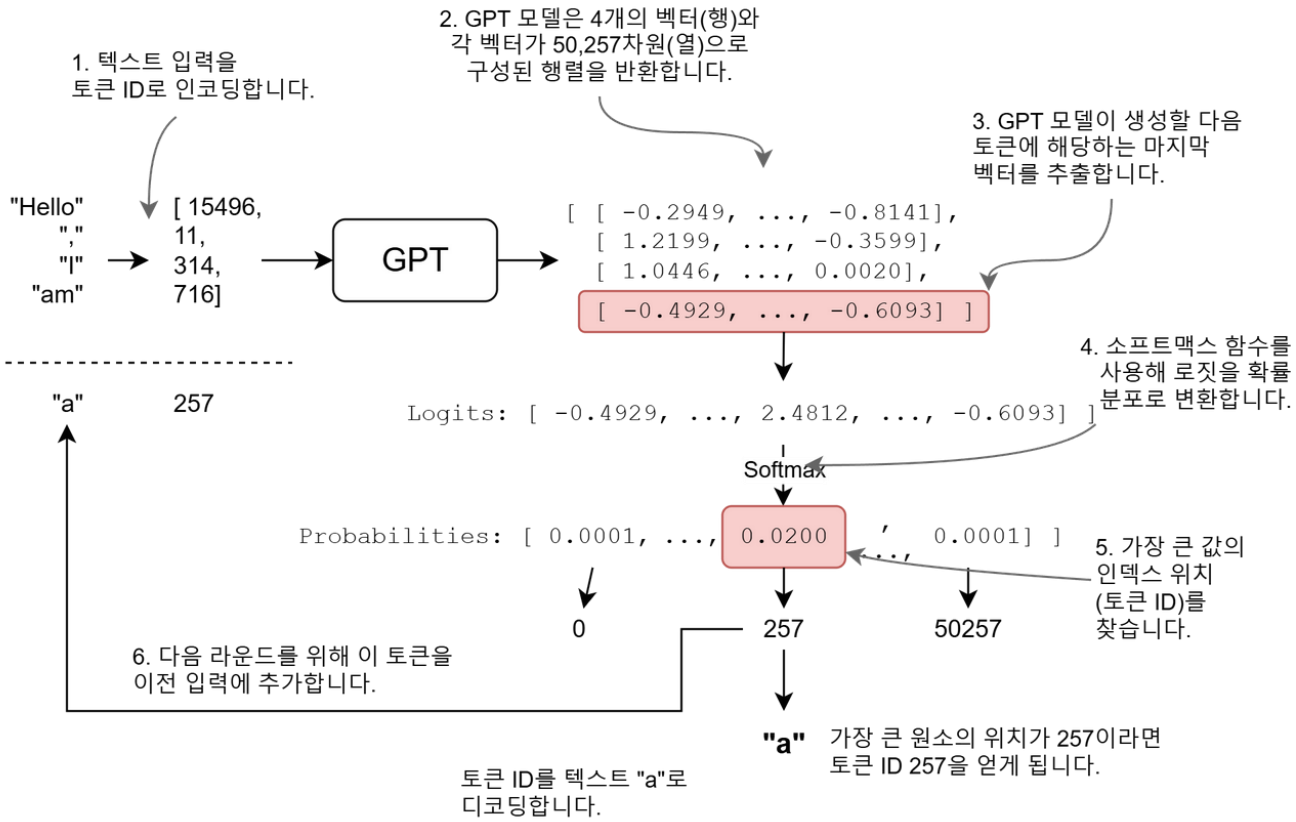
- Batch Size
- Sequence Length
- Embedding Dimension

Output Probabilities Tensor

- Batch Size
- Sequence Length
- Vocabulary Length.

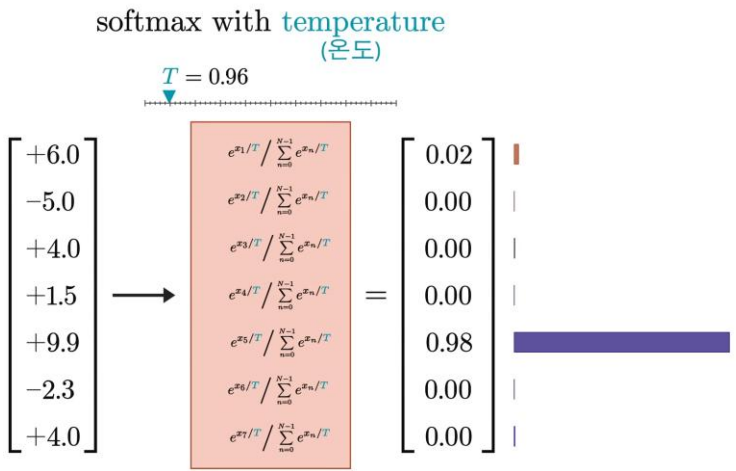


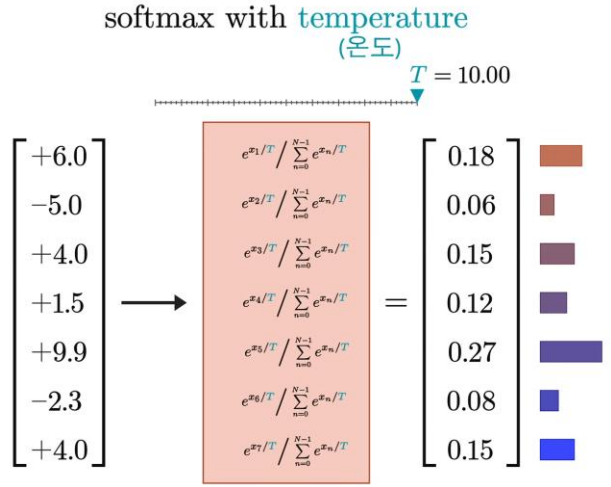
Output Token 생성 과정



Temperature(T)

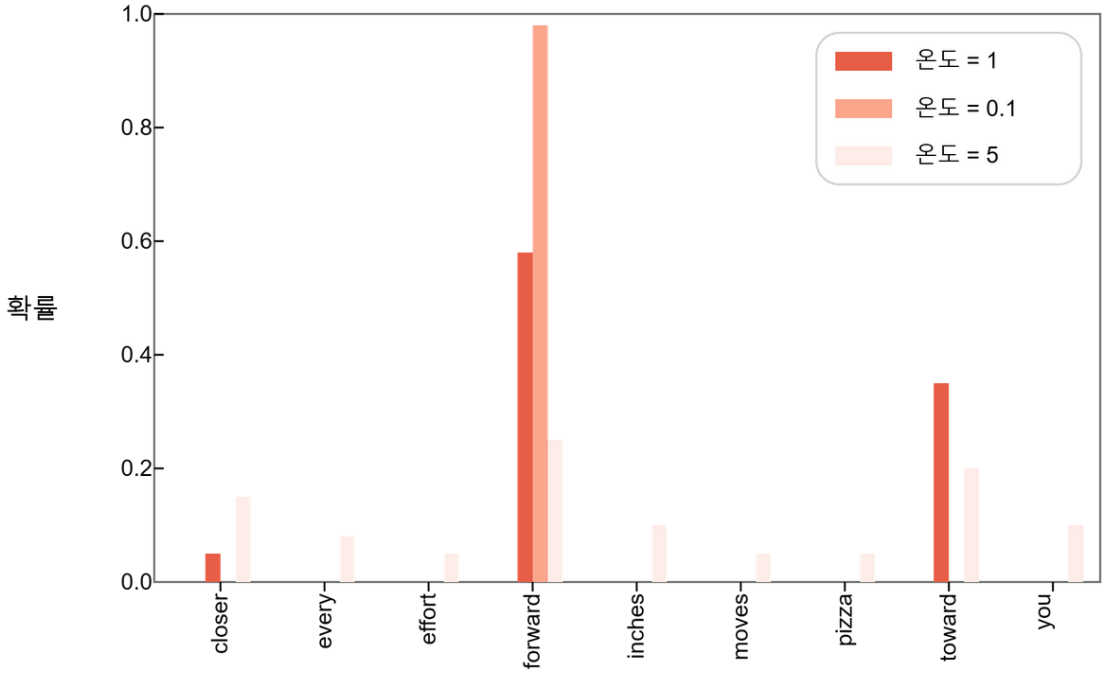
- T가 크면 더 작을 값들이 더 많은 비중을 차지하게 됨
  - Softmax에 T를 추가





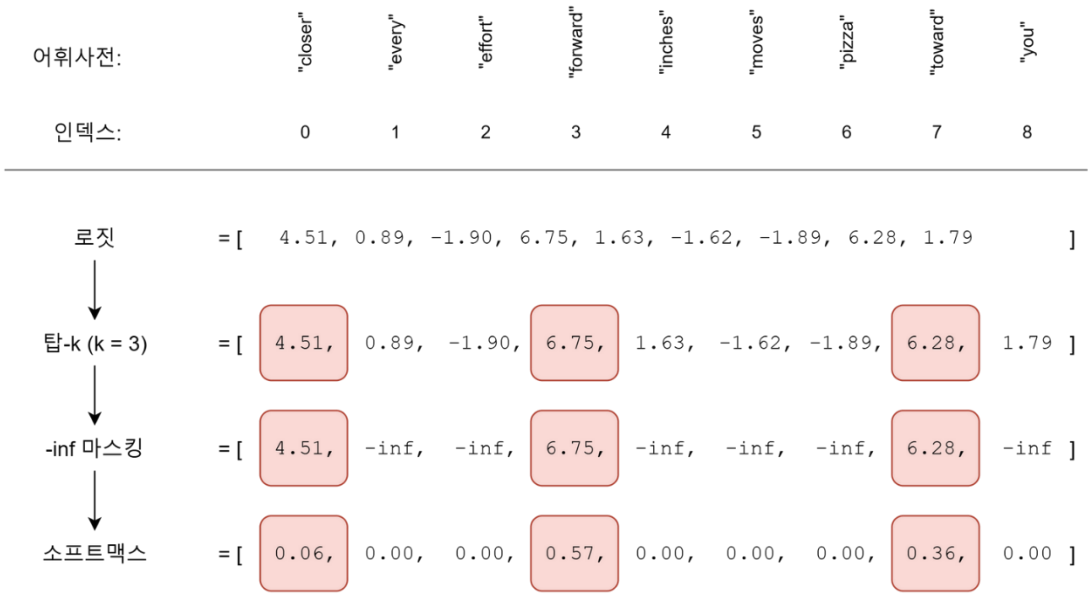
Temperature

- 온도 스케일링
  - 온도가 낮을수록 높은 확률이 확정적



Top-K 샘플링

- 샘플을 K만큼 고른 후 확률 기반으로 랜덤으로 선택



지식의 저장

Attention 의미

- 문맥의 의미를 더해서 새로운 문맥의 의미를 만듦

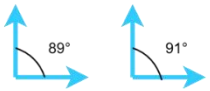
creature

A fluffy blue creature

FFN 지식의 저장

백터 직교

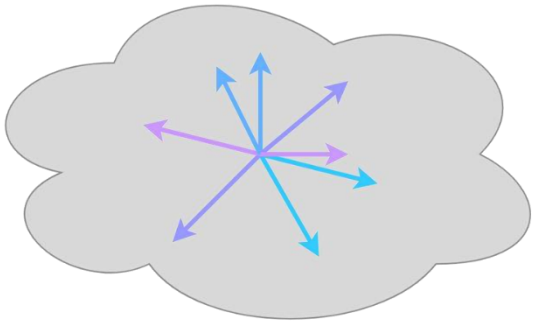
- N차원에서 서로 직교하는 벡터는 N개
- 89-91도까지 허용하면 고차원일수록 서로 직교하는 백터가 대다수임

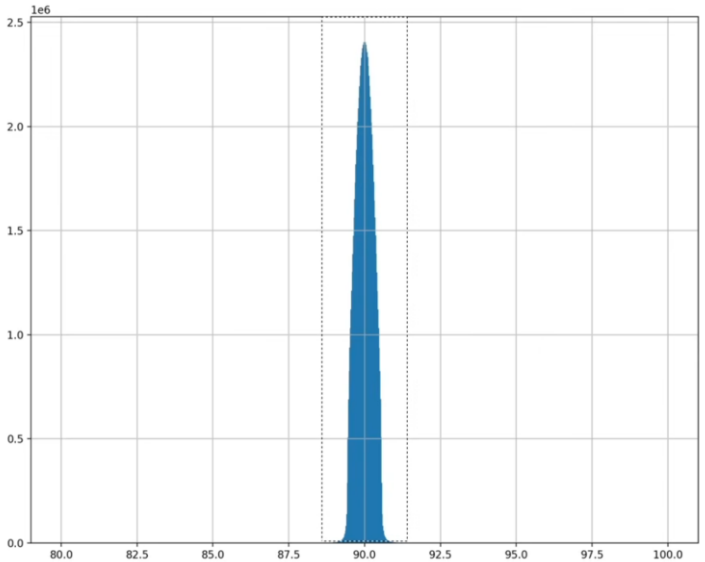


Choose multiple vectors,
each pair between 89° and 91° apart

Johnson–Lindenstrauss Lemma ⇨ Maximum # of vectors: ≈ exp(ε · N)

N-dimensional Space

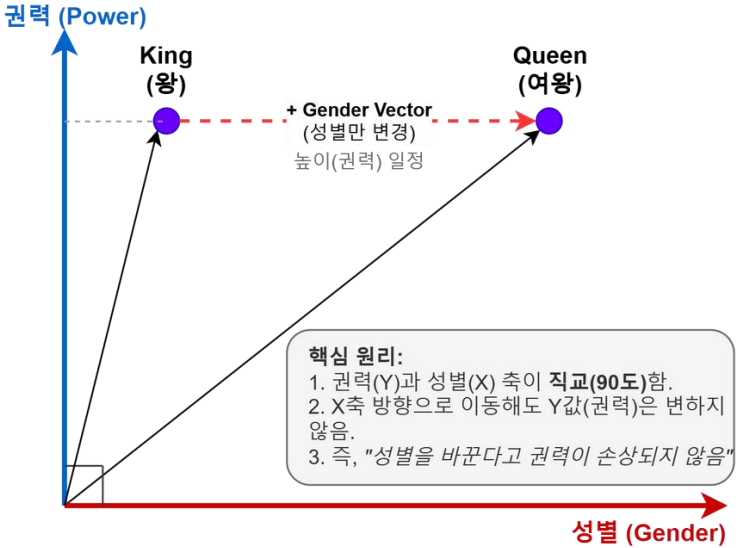


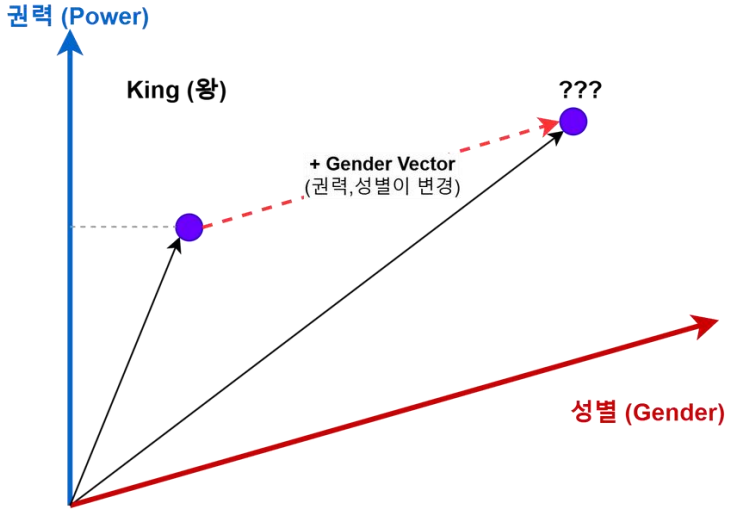


100만개 벡터를 서로 직교하는지 검사

FFN 지식의 저장

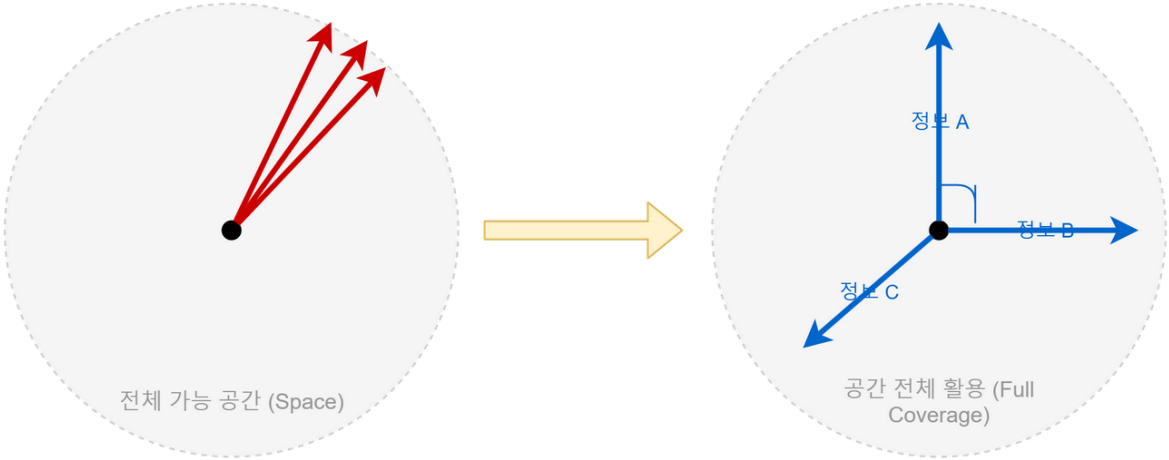
- 직교성(Orthogonality)과 특징 분리(Disentanglement)
  - 개념들이 서로 간섭하지 않도록 함





FFN 지식의 저장

정보 저장의 효율성



평행한 벡터 (Parallel)

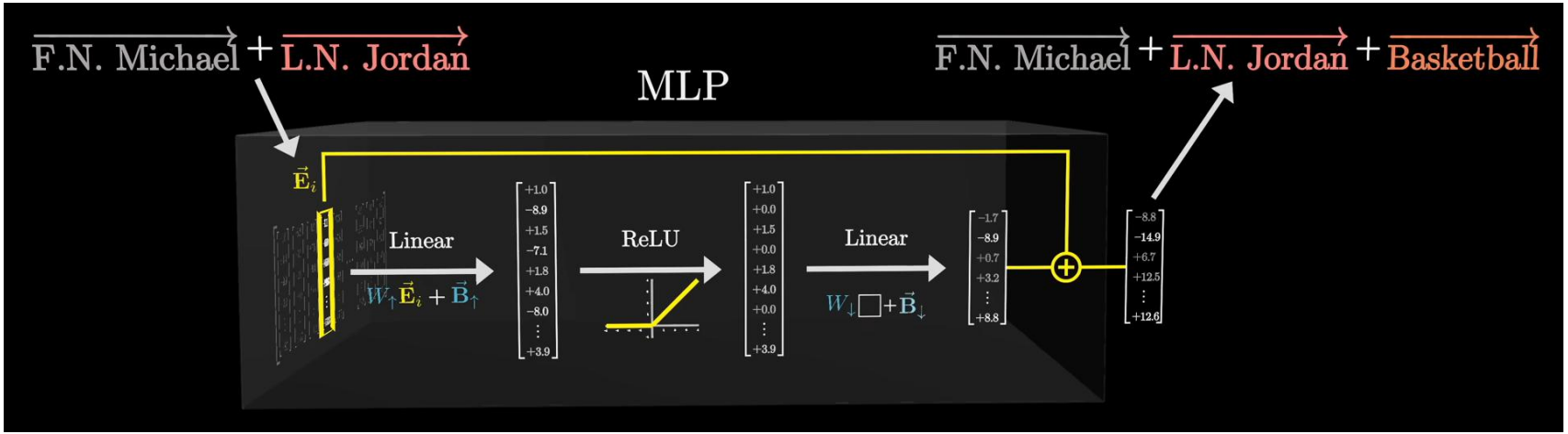
❌ **정보의 중복**
- 모두 같은 이야기만 함
- 나머지 공간은 텅 빔 (낭비)
- 유사도가 너무 높아 구분이 안 됨

직교하는 벡터 (Orthogonal)

✅ **표현력 극대화**
- 서로 완전히 다른 정보를 표현
- 공간을 골고루 사용하여 꽉 채움
- 내적값 0 = 간섭 없음 (독립적)

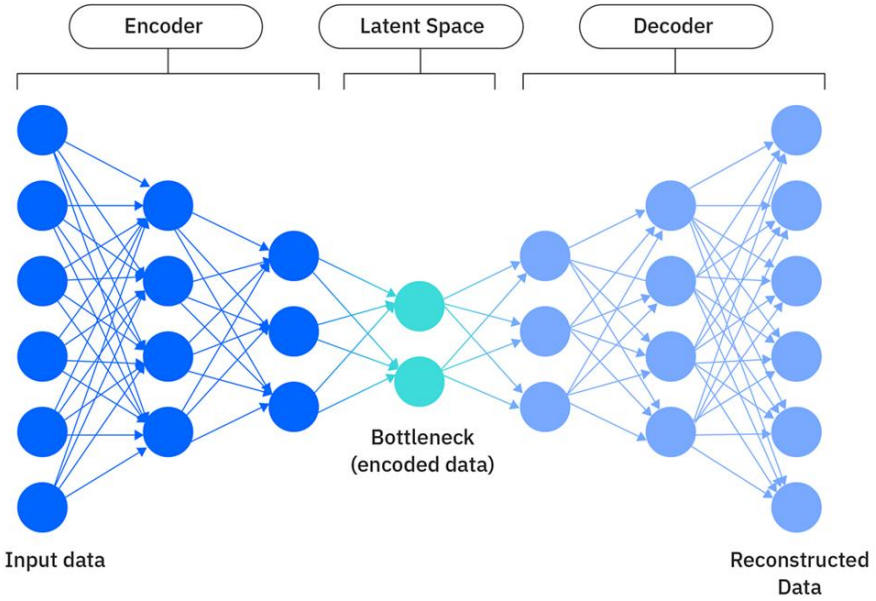
FFN 지식의 저장

- 수많은 Layer들이 정보를 저장하고 있으며, 다음 Layer로 가면서 새로운 정보를 만듦



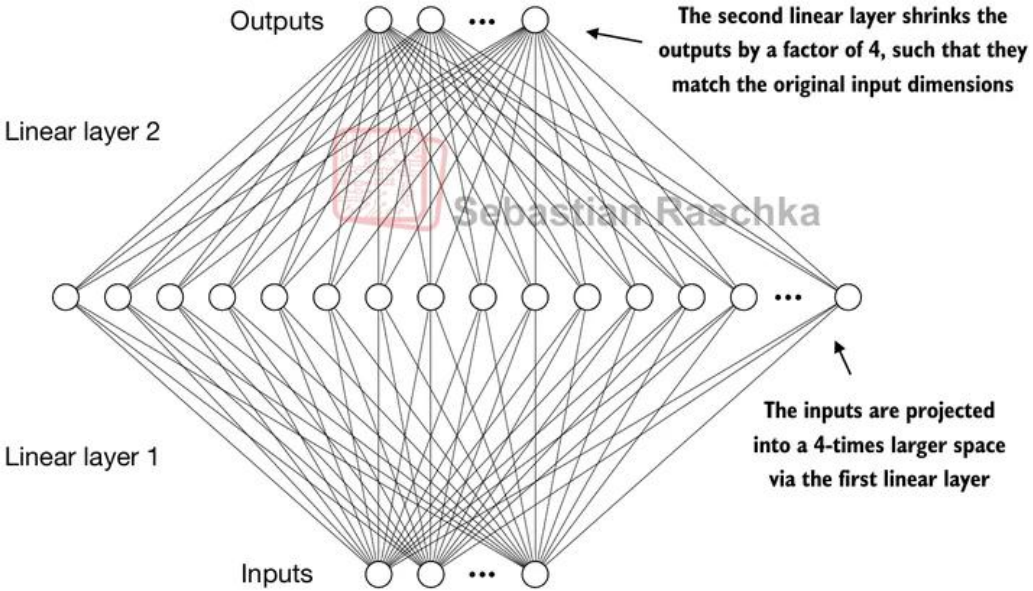
Latent Space

- 차원이 축소되어 정보가 압축되어서 저장되는 공간



Expansion Layer

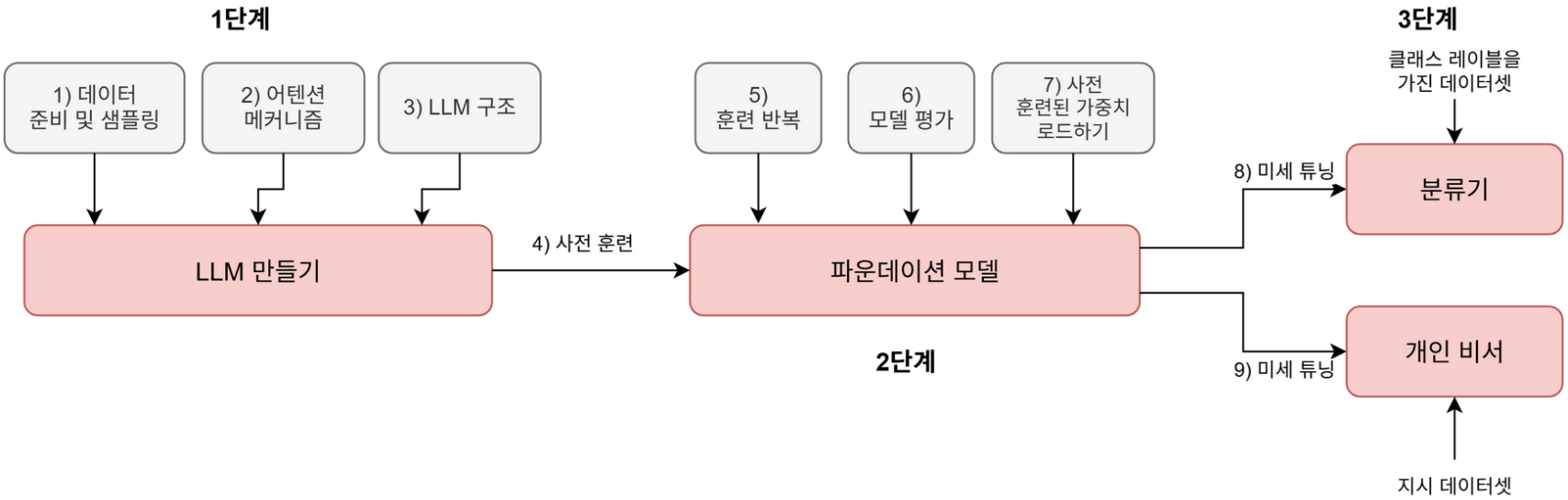
- 차원을 확대하여 특징을 세분화하여 저장, 활성화 함수에 의한 정보 손실 방지



Chapter_4_Excercise_GPT.ipynb

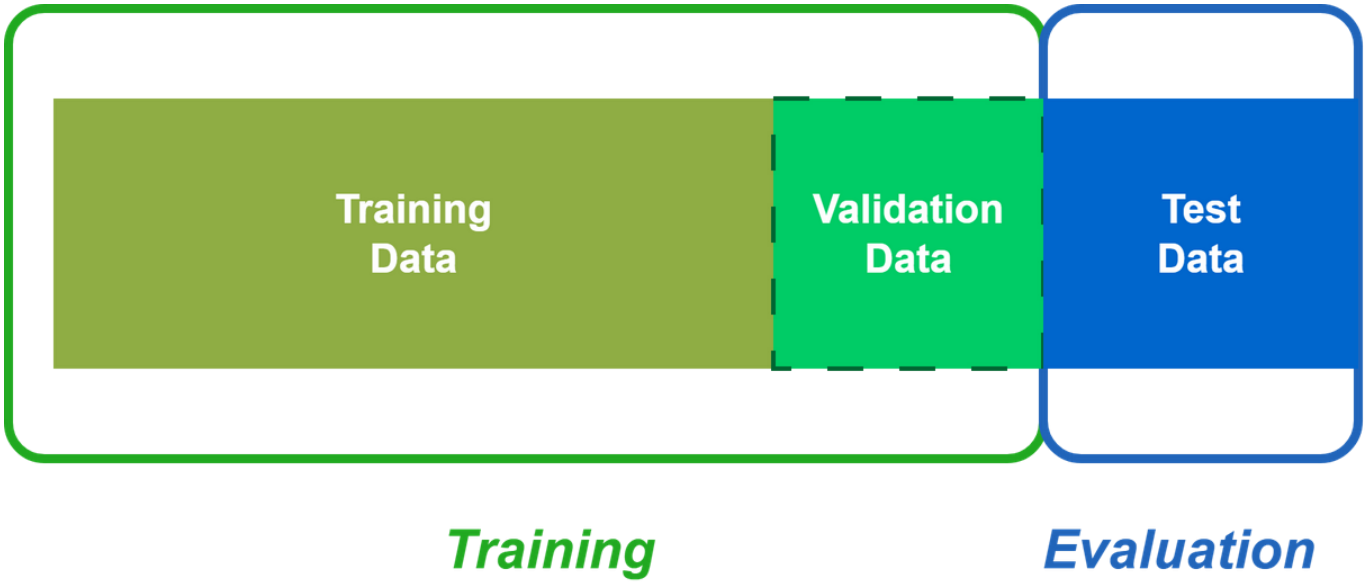
학습

학습과정


1단계
1) 데이터 준비 및 샘플링
2) 어텐션 메커니즘
3) LLM 구조
LLM 만들기
4) 사전 훈련
5) 훈련 반복
6) 모델 평가
7) 사전 훈련된 가중치 로드하기
파운데이션 모델
2단계
8) 미세 튜닝
9) 미세 튜닝
3단계
클래스 레이블을 가진 데이터셋
분류기
개인 비서
지시 데이터셋

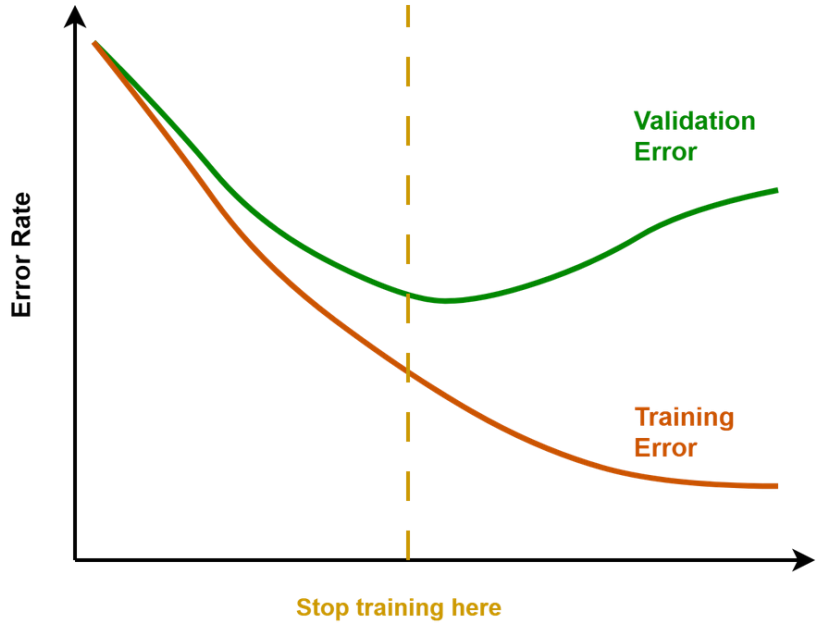
Dataset

- Training과 Evaluation 분리



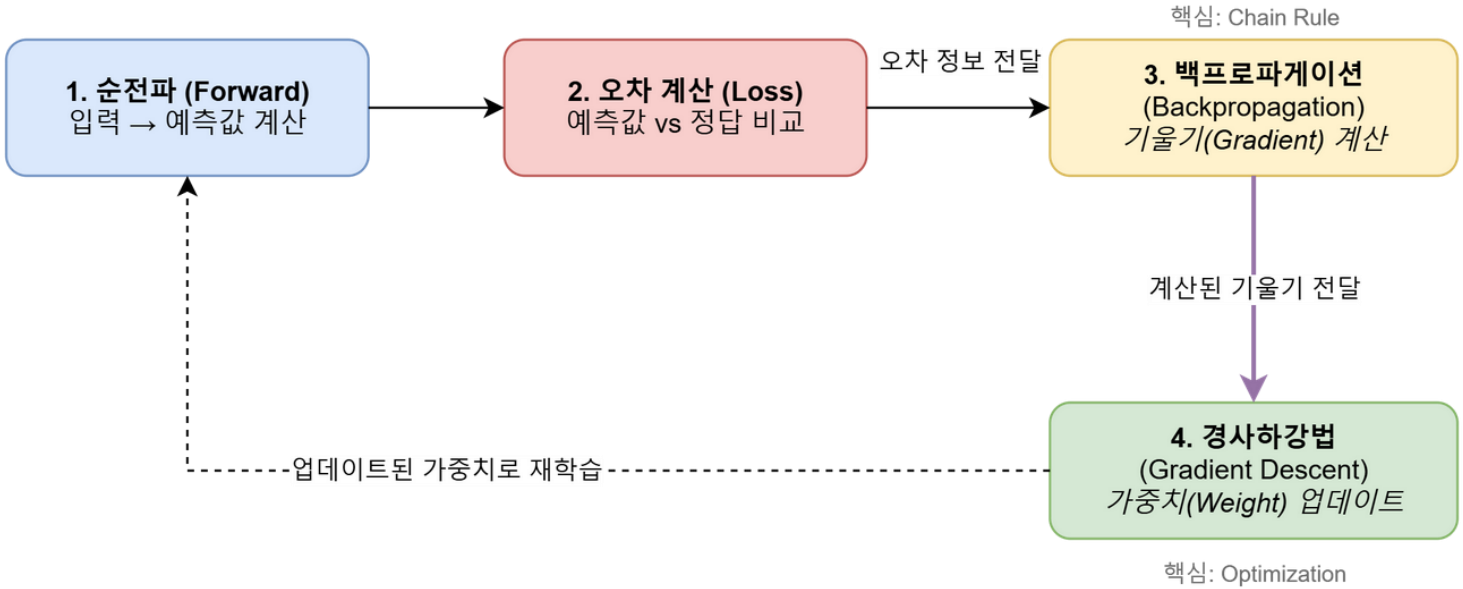
Validation Dataset

- overfitting 방지, hyperparameter tuning을 결정



학습과정

- Loss를 최소화하기 위해 가중치를 찾는 과정

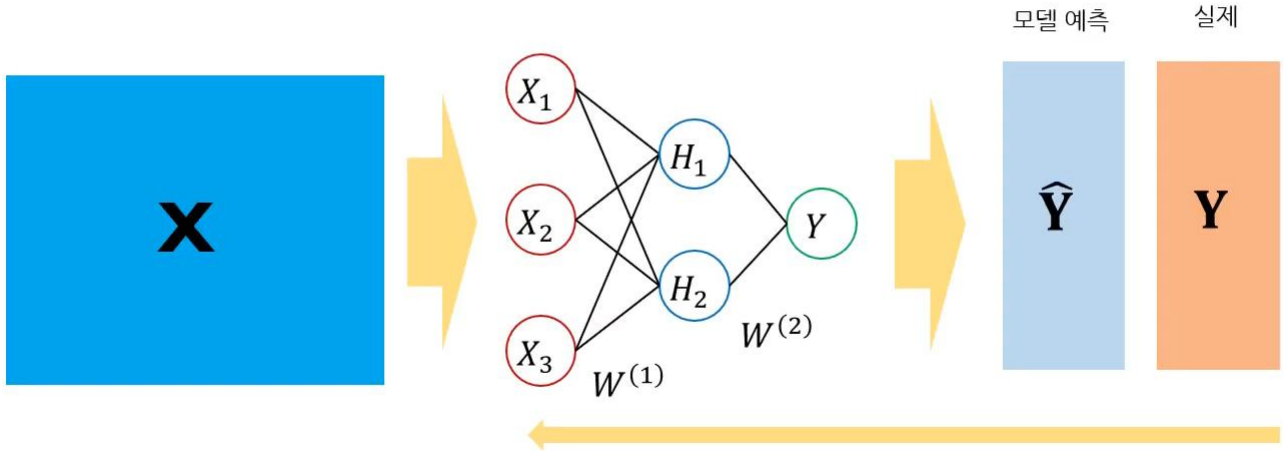


[Neural Network 9] 역전파 backpropagation 알고리즘 (수정본)

비용함수

- 모델로부터 $\bar{Y}$값과 실제 Y값의 차이를 최소로 하는 가중치

$$\arg\min_{w} \sum_{i} L(Y, f(X; w))$$



[핵심 머신러닝] Cross Entropy

비용함수-수치예측(MSE)

- 예측한 값과 실제값의 차이를 측정
  - Mean squared error (MSE) $C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ (y_i :실제값, $\hat{y}_i$:예측값)

실제값	예측값
10	11
20	19
30	30
40	43
50	47

$$\frac{1}{5}\sum_{i=1}^{5}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{5}[(10-11)^2 + (20-19)^2 + (30-30)^2 + (40-43)^2 + (50-47)^2]$$

$$= \frac{1}{5}(1 + 1 + 0 + 9 + 9)$$

$$= 4$$

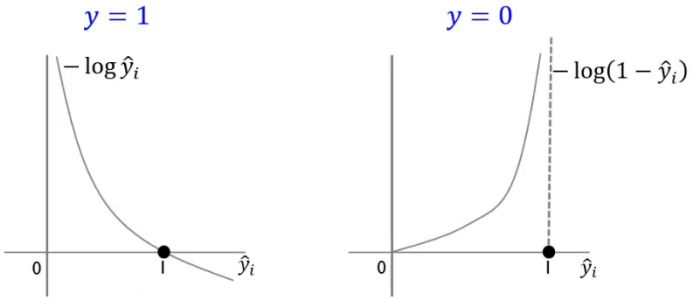
[핵심 머신러닝] Cross Entropy

비용함수-Binary 분류(Cross Entropy)

- 0과 1로 분류하는 예제
  - 0과 1은 분류하기 위한 라벨이므로 숫자로서 의미가 없음

- Classification: cross entropy $L = -\sum_i y_i \log \hat{y}_i$ (y_i:실제값, $\hat{y}_i$:예측값)

$$C(y, \hat{y}_i) = \begin{cases} -\log \hat{y}_i, & y = 1 \\ -\log(1 - \hat{y}_i), & y = 0 \end{cases}$$



$$C(\pi(x), y) = -y \log \hat{y}_i - (1 - y) \log(1 - \hat{y}_i)$$

$$\min_{\beta} C(y, \hat{y}_i)$$

$$\mathcal{L}(Y, \hat{Y}) = \sum_{i=1}^{N} \{-y^{(i)} \cdot \log(\hat{y}^{(i)}) - (1 - y^{(i)}) \cdot \log(1 - \hat{y}^{(i)})\}$$

$$Y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \hat{Y} = \begin{bmatrix} 0.99 \\ 0.11 \\ 0.50 \\ 0.05 \\ 0.75 \end{bmatrix}$$

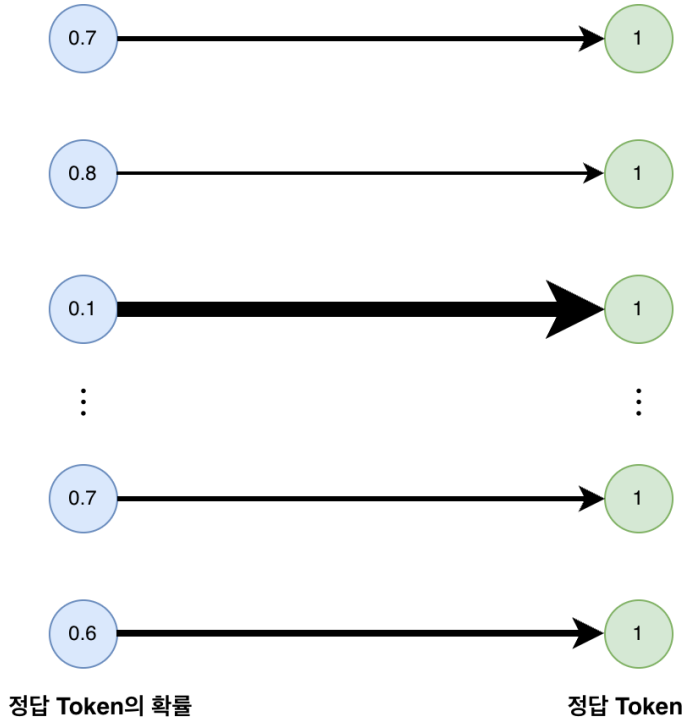
$$l(y^{(1)}, \hat{y}^{(1)}) = -1 \cdot \log(0.99) \approx 0.01$$
$$l(y^{(2)}, \hat{y}^{(2)}) = -1 \cdot \log(0.89) \approx 0.12$$
$$l(y^{(3)}, \hat{y}^{(3)}) = -1 \cdot \log(0.50) \approx 0.69$$
$$l(y^{(4)}, \hat{y}^{(4)}) = -1 \cdot \log(0.05) \approx 2.99$$
$$l(y^{(5)}, \hat{y}^{(5)}) = -1 \cdot \log(0.25) \approx 1.39$$

너가 제일 많이 틀렸어

[핵심 머신러닝] Cross Entropy

비용함수-분류(Cross Entropy)

- 정답 Token의 확률 합: 모두가 1이 되면 됨



경사하강법(Gradient Descent)

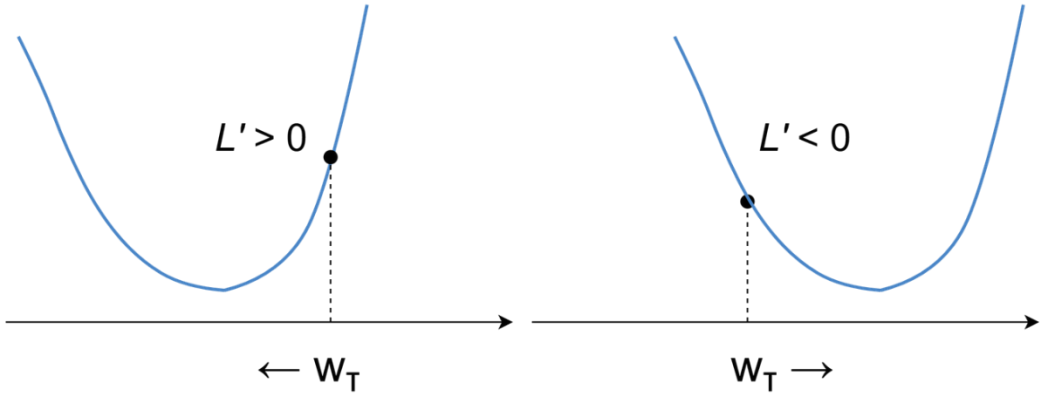
- Loss를 최소화하기 위해 가중치를 찾는 과정
  - Optimizer가 학습 과정에서 스스로 조율해 나감

$$w_{T+1} = w_T - \alpha \cdot L'(w_T)$$

$$0 < \alpha < 1: \text{learning rate}$$

좀 더 섬세하고 촘촘히? **Small α**

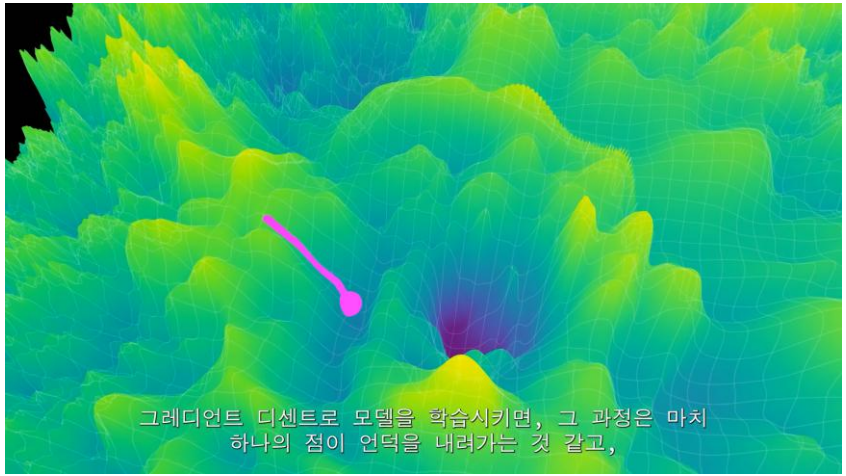
좀 더 빠르게? **Large α**

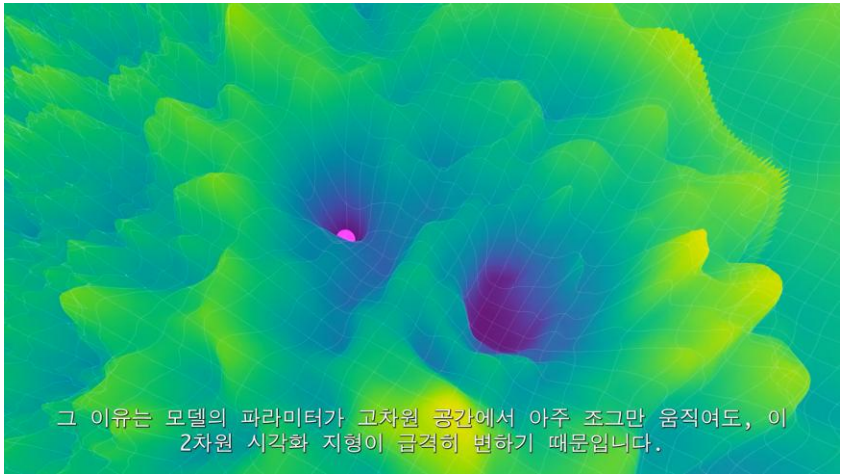


[Neural Network 9] 역전파 backpropagation 알고리즘 (수정본)

LOCAL MINIMUM

- 고차원에서는 발생하지 않음
  - 기울기 소실이 문제(너무 평탄해서 어디로 가야할 지 모르게 되는 경우)

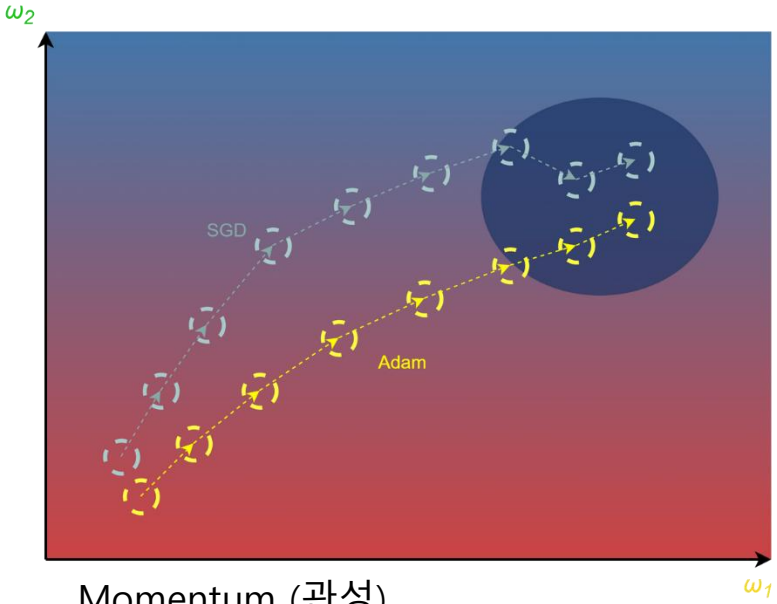




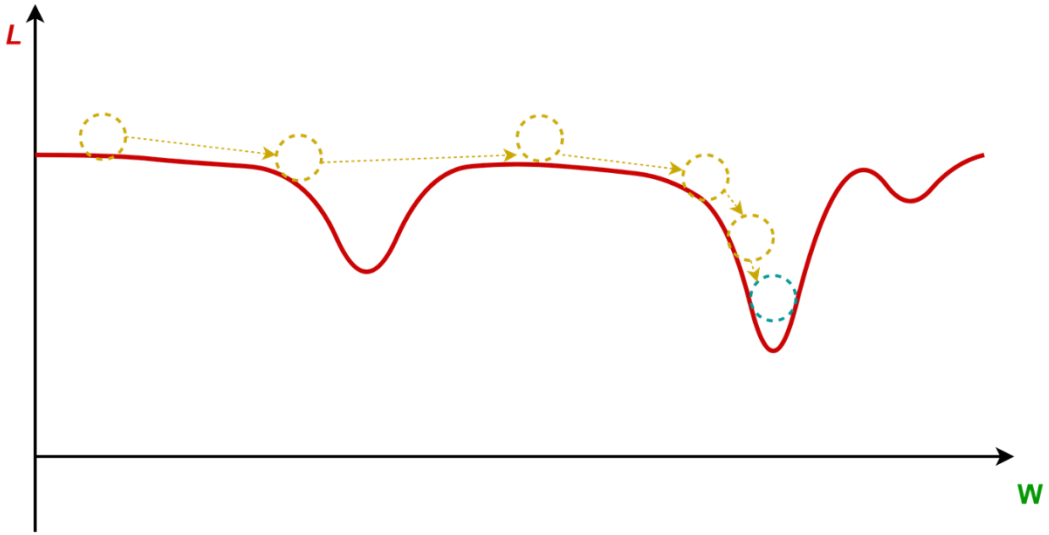
AI는 왜 아직도 '내리막길'을 걷는가? | 경사하강법과 지역 극솟값 - YouTube

Adam Optimizer

- 과거의 기울기 변화량(Moment)을 추정하여, 각 파라미터마다 학습률을 다르게 조절



Momentum (관성)
기울기가 가리키는 방향으로 속도를 붙여서 가자



RMSprop (적응형 학습률)
보폭(Step size)을 상황에 맞춰 조절하자.

Optimization for Deep Learning (Momentum, RMSprop, AdaGrad, Adam)

Adam Optimizer

Learning Rate

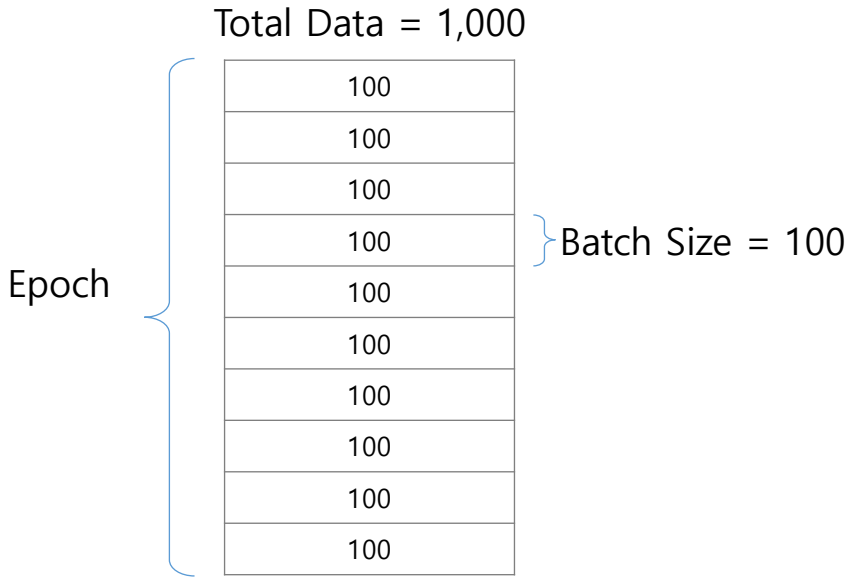
관성: 방향

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

속도: 보폭

그래디언트가 컸던 매개변수는 $\sqrt{\hat{v}_t}$로 나누어 보폭을 줄이고, 작았던 매개변수는 보폭을 키워 안정적으로 수렴하게 합니다.

Epoch, Step, Batch Size



- **Epoch**: 전체 데이터를 1회 학습하는 것
- **Step (or Iteration)**: 파라미터를 1회 업데이트하는 것
- **Batch Size**: Step 1회에서 사용하는 데이터 개수

- 전체 데이터: 1,000
- Batch Size: 100
- 한 Epoch에 10번의 Step이 필요

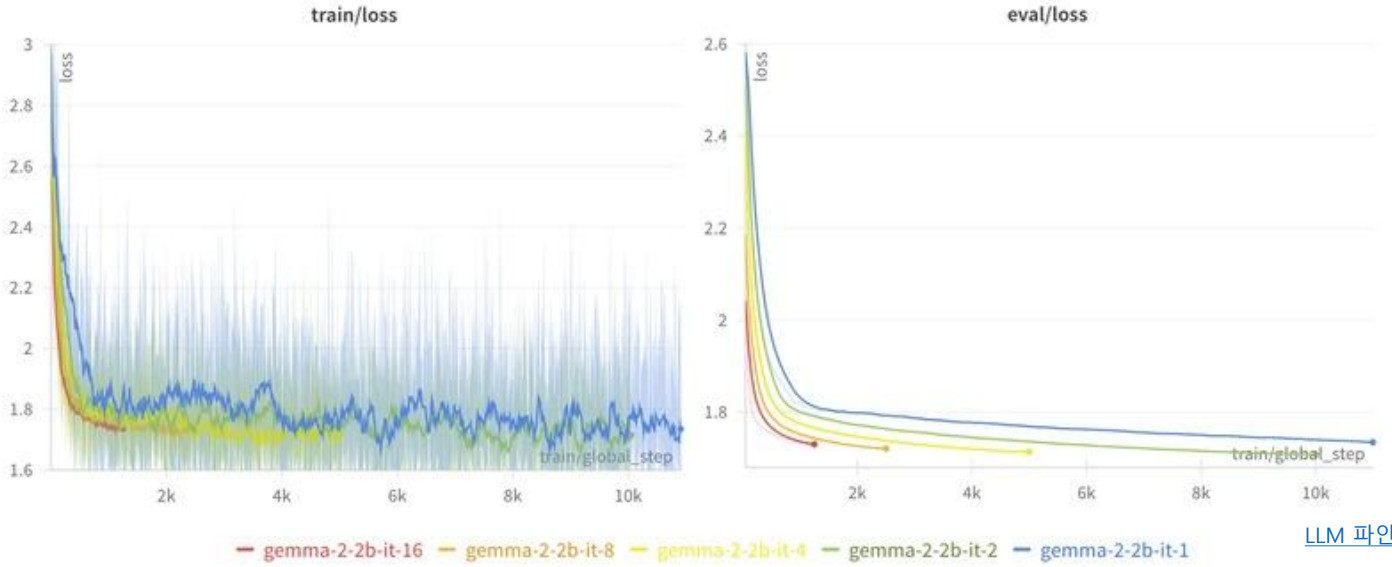
Batch Size

작은 배치 크기

- 학습 과정이 불안정해질 수 있지만, 일반화 성능은 더 좋아지는 경향이 있습니다.
- 메모리 사용량이 적어 제한된 자원으로도 학습 가능하지만, 전체 학습 시간이 길어질 수 있습니다.

큰 배치 크기

- 학습이 안정적이지만, 세부적인 특징을 놓쳐 과적합(Overfitting)의 위험이 있습니다.
- 더 많은 메모리가 필요하지만 학습 시간이 더 단축시킬 수 있습니다.



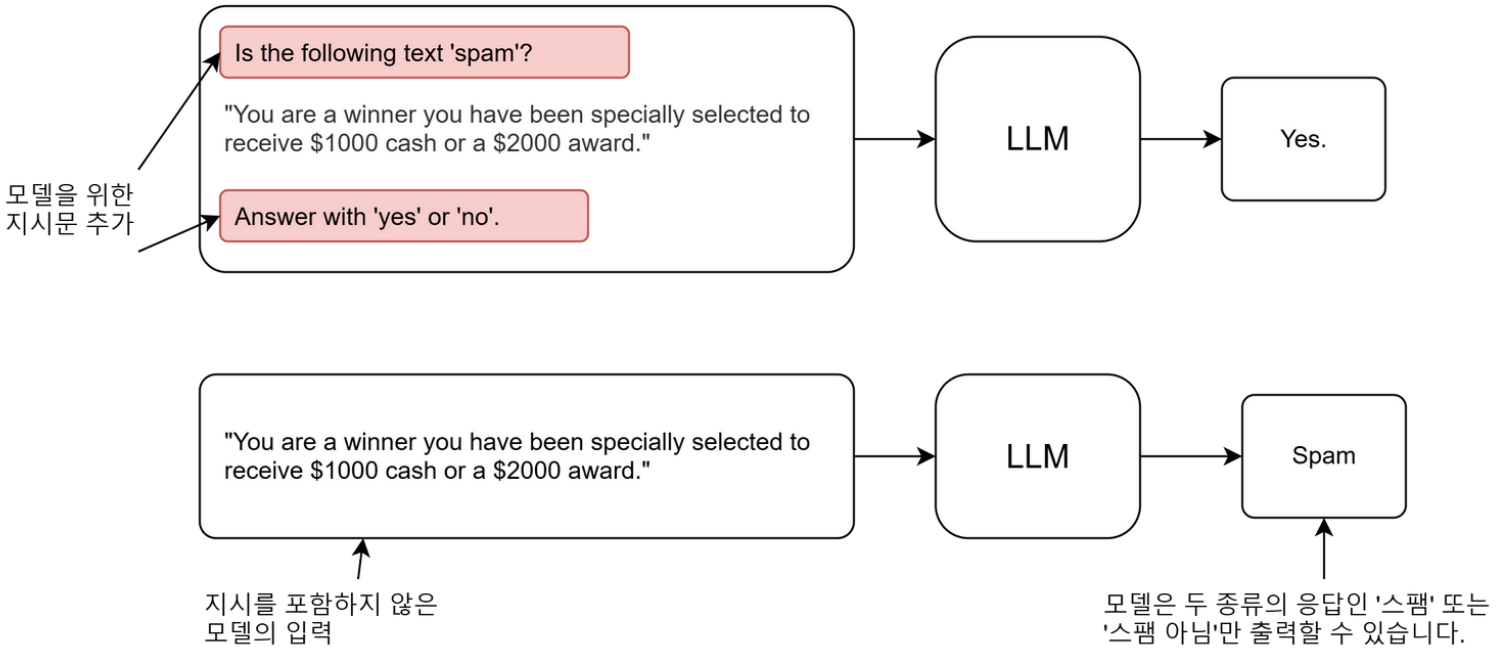
LLM 파인튜닝 최적화 가이드

Chapter_5_Excercise_Pretraining.ipynb

Fine-tuning

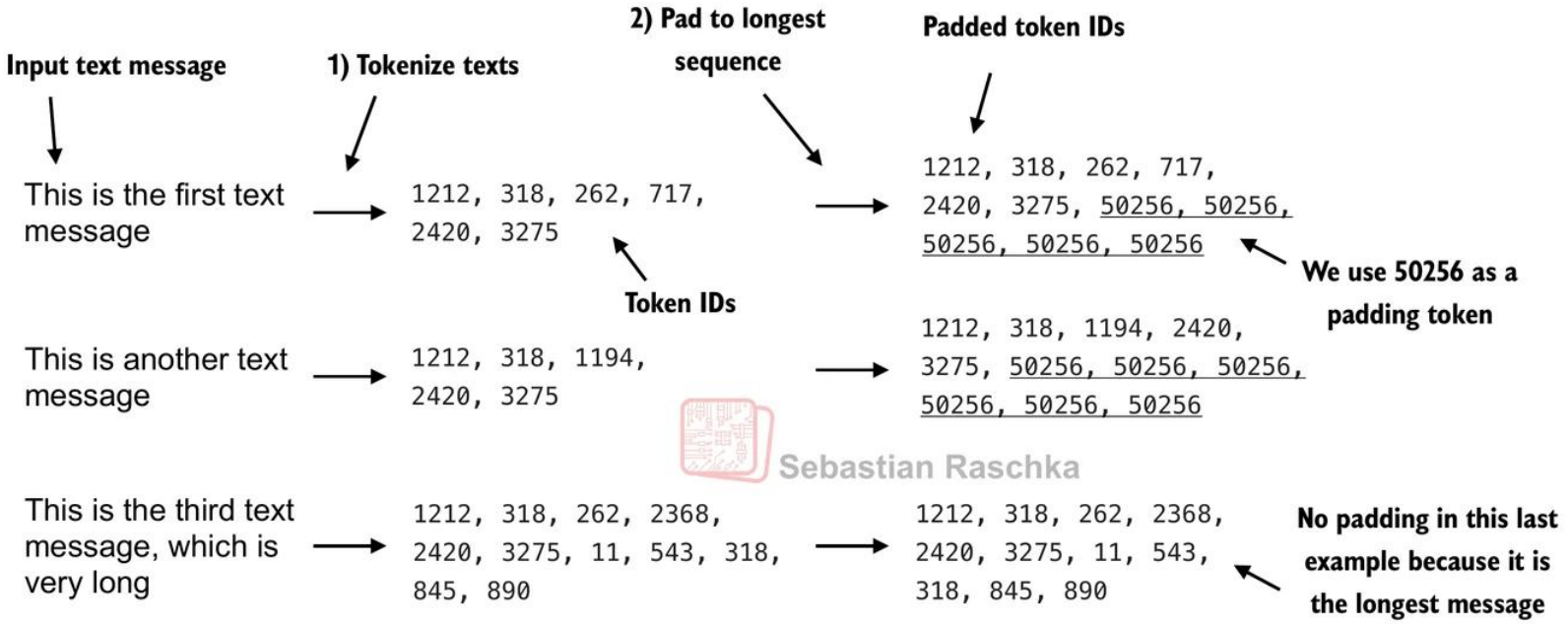
분류 미세 튜닝

- 분류를 위해 마지막 layer 수정



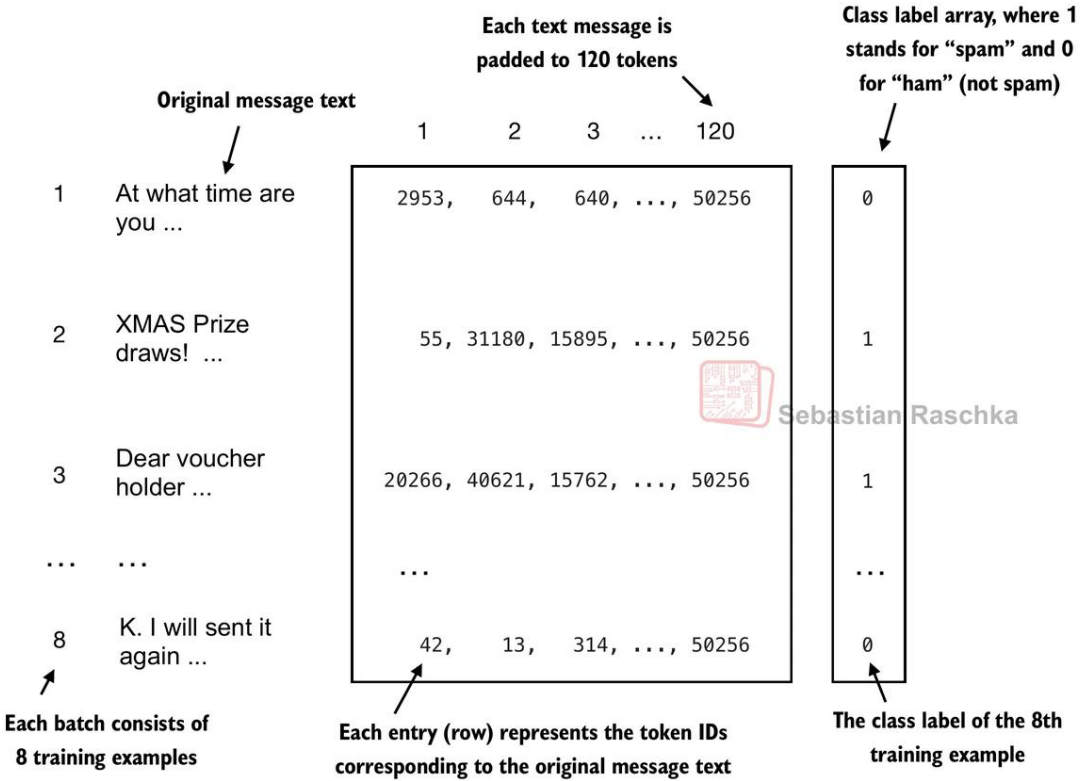
분류 미세 튜닝

- Input 데이터가 짧은 경우 Padding 필요



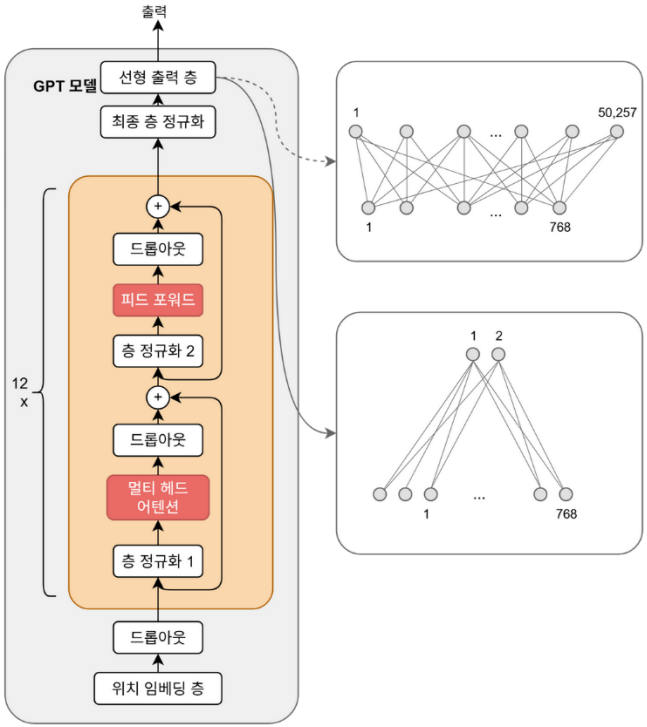
분류 미세 튜닝

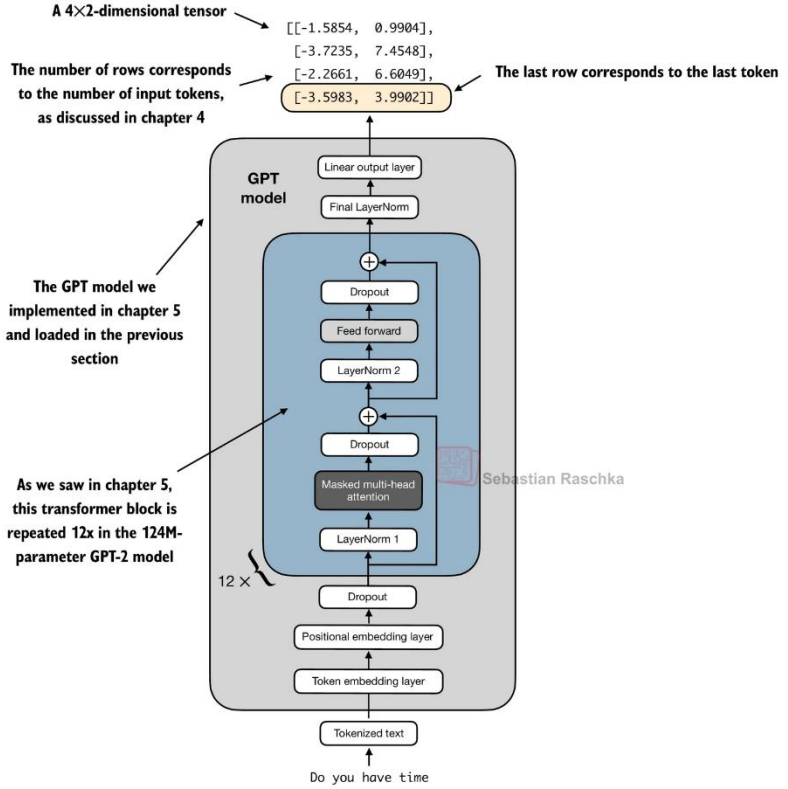
- Target은 0,1로 라벨링



분류 미세 튜닝

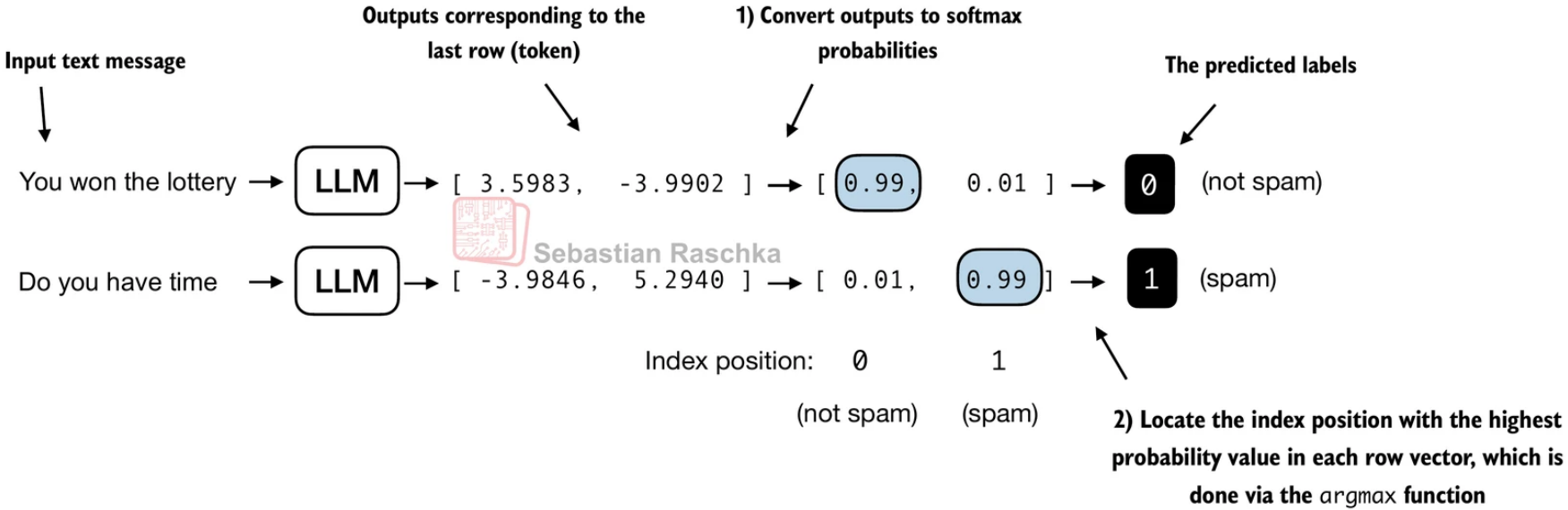
- 분류를 위해 마지막 layer 수정





분류 미세 튜닝

- 결과는 softmax 후 argmax로 선택

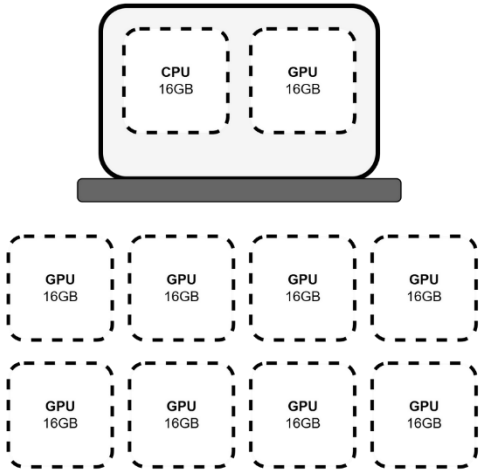


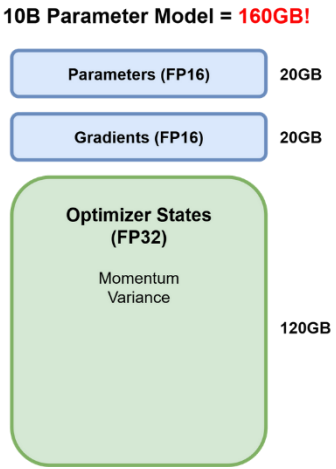
Chapter_6_Excercise_Finetuning_Classification.ipynb

LoRA

LLM 학습의 문제

- LLM 학습은 비싸다, 많은 메모리 요구
- 10B의 모델 학습에 160GB 메모리 필요





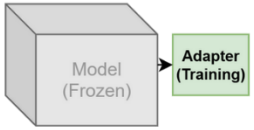
LoRA: Percent of Total Parameters

- 모델이 클수록 유리

Rank	7B	13B	70B	180B
1	0.00%	0.00%	0.00%	0.00%
2	0.01%	0.00%	0.00%	0.00%
8	0.02%	0.01%	0.01%	0.00%
16	0.04%	0.03%	0.01%	0.01%
512	1.22%	0.90%	0.39%	0.24%
1,024	2.45%	1.80%	0.77%	0.48%
8,192	19.58%	14.37%	6.19%	3.86%

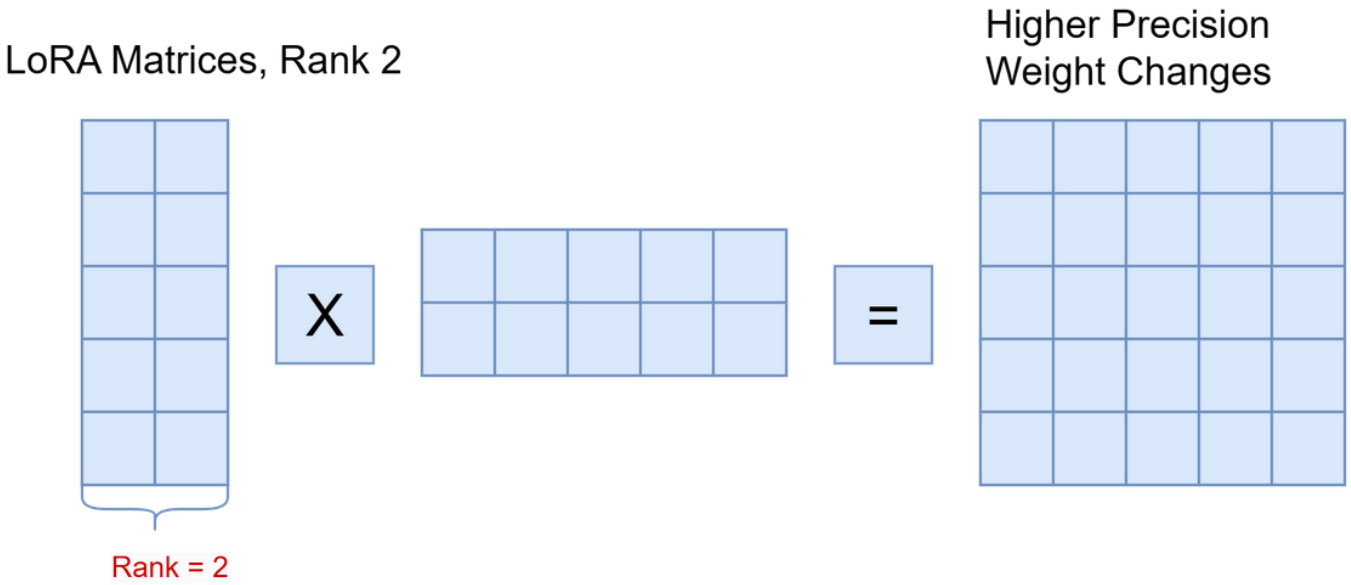
Percentages are understated since models are made up of different layers, but you get the idea.

Full Fine-tuning vs PEFT

	Full Fine-tuning	PEFT (e.g., LoRA)
		
학습 범위	모델 전체 (100%)	극일부 (< 1%)
GPU 메모리	매우 높음 (고성능 GPU 필요)	낮음 (소비자용 GPU 가능)
저장 용량	대용량 (수십 GB)	초경량 (수십 MB)
망각 위험	높음 (Catastrophic Forgetting)	매우 낮음 (원본 보존)
주요 용도	도메인 변경 / 지식 주입	특정 스타일 / 지시 이행

LoRA

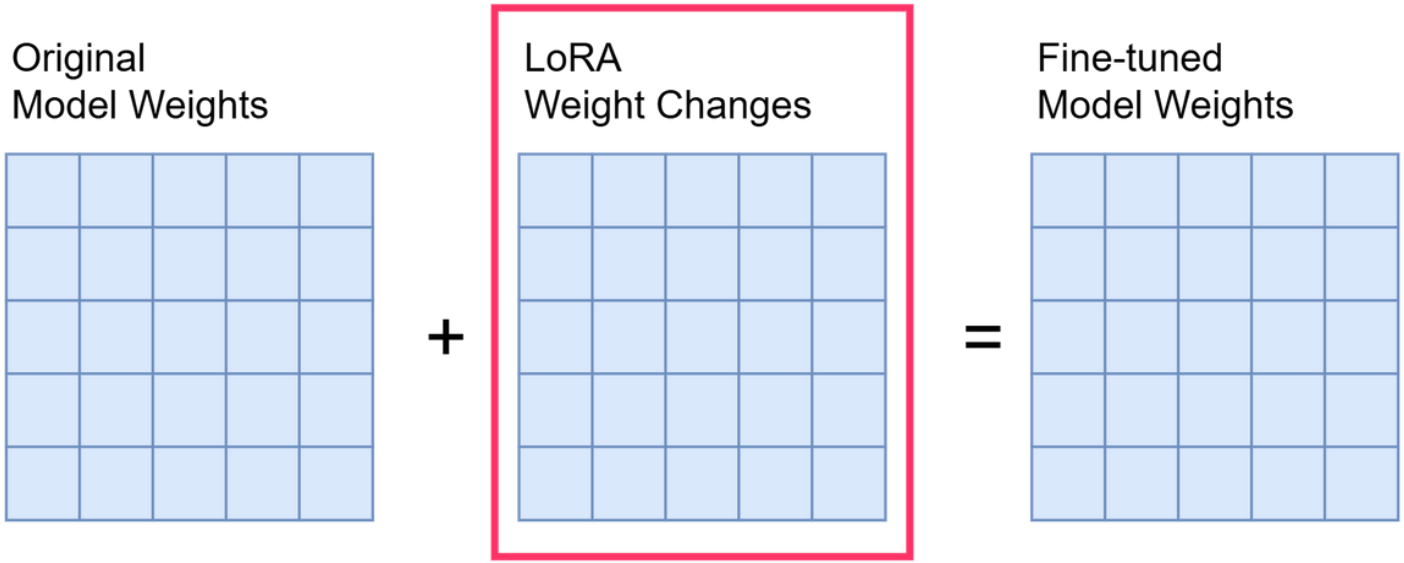
- A, B 행렬로 전체 행렬과 동일한 크기 생성
  - Increasing Precision by Increasing Rank



LoRA & QLoRA Fine-tuning Explained In-Depth

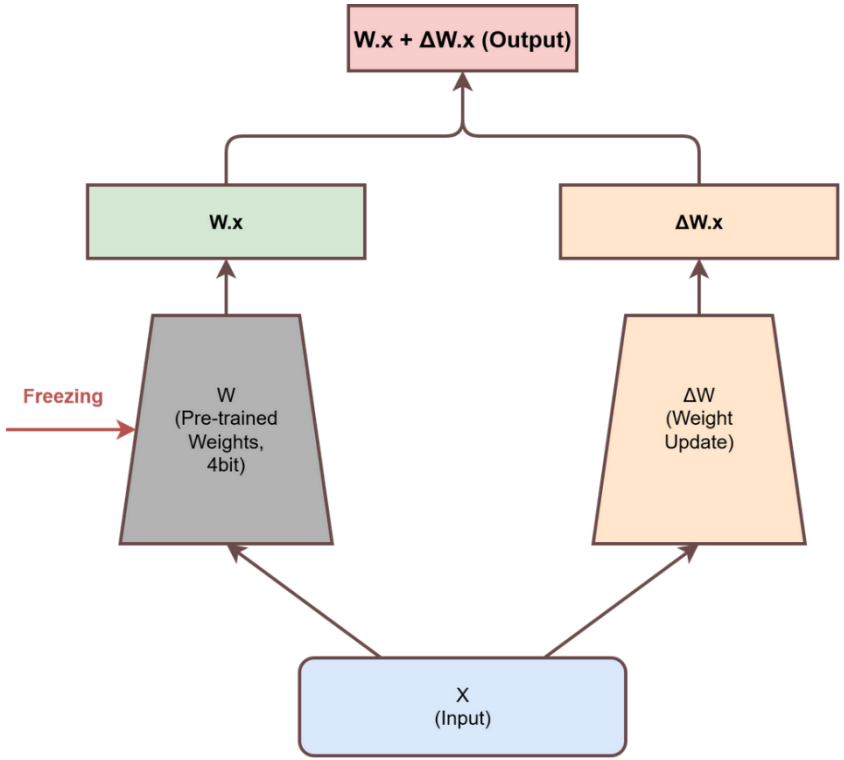
LoRA

- Adding the LoRA Weight Changes



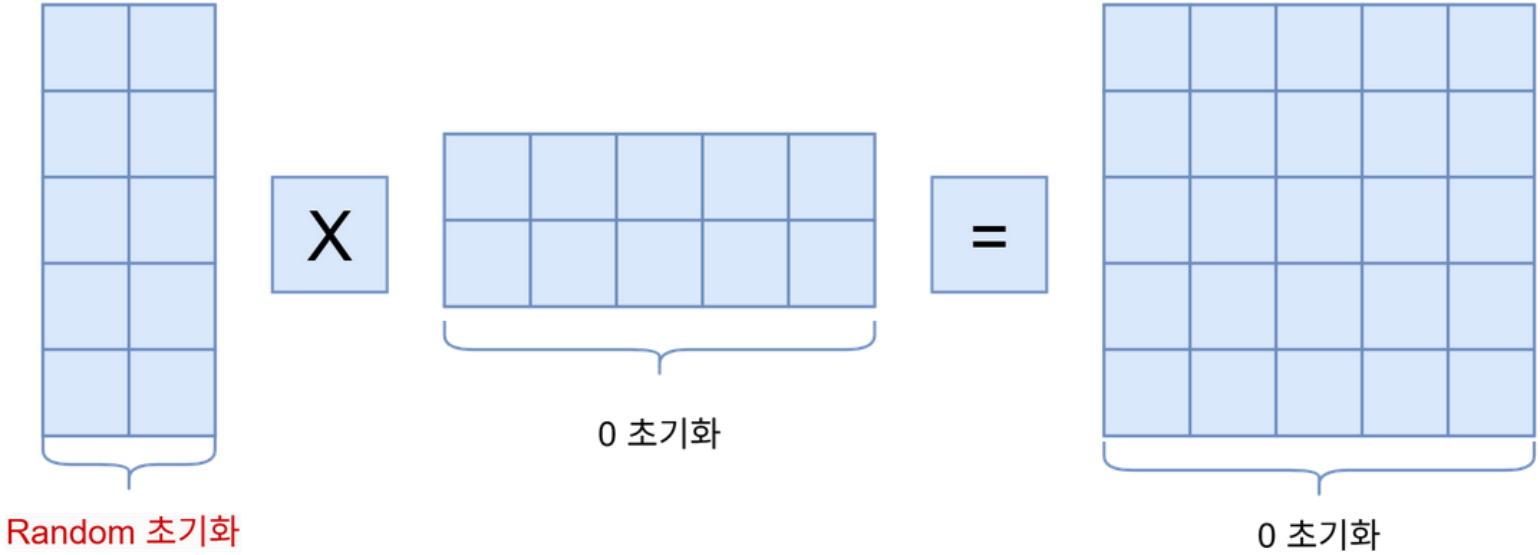
LoRA

- 학습 시 LoRA만 업데이트함



LoRA: 초기화

- 행렬 A는 랜덤으로 초기화, 행렬 B는 0으로 초기화함
  - A,B 모두 Random으로 초기화하면 학습이 안된 모델과 동일
  - A,B 모두 0으로 초기화하면 기울기 계산을 하지 않아 Weight가 업데이트가 되지 않음



QLoRA

- Convert all weights of a LLM to 4-bit format
  - LoRA에서 걸림돌이 되는 모델 메모리 문제 해결

4-bit e.g. 0101

⇒ 2^4 = 16 unique combinations

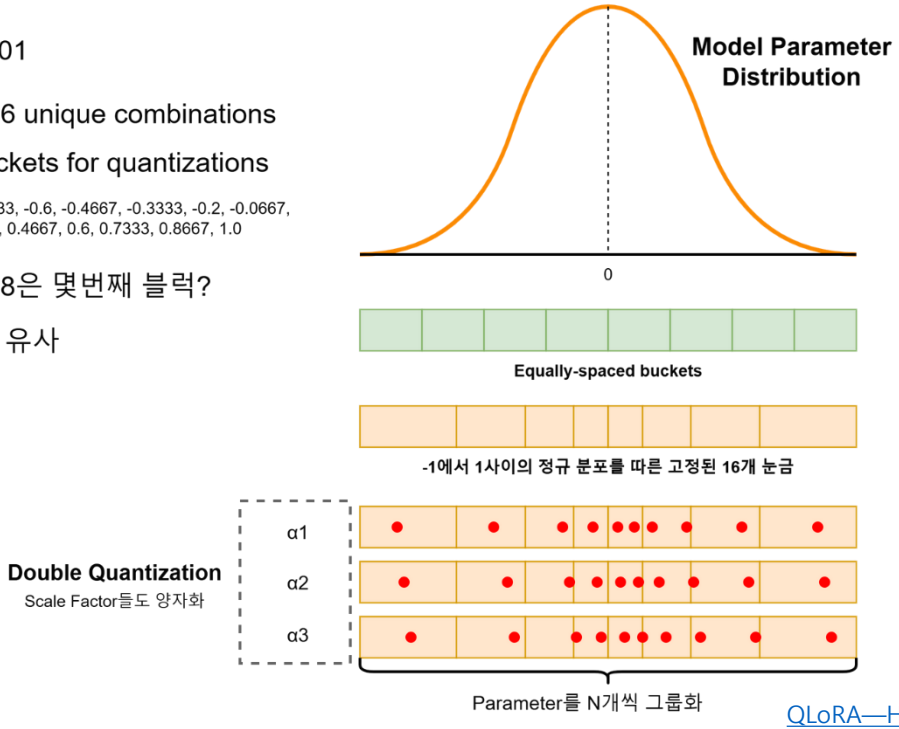
⇒ 16 buckets for quantizations

-1.0, -0.8667, -0.7333, -0.6, -0.4667, -0.3333, -0.2, -0.0667, 0.0667, 0.2, 0.3333, 0.4667, 0.6, 0.7333, 0.8667, 1.0

⇒ 0.5678은 몇번째 블럭?

⇒ 0.6과 유사

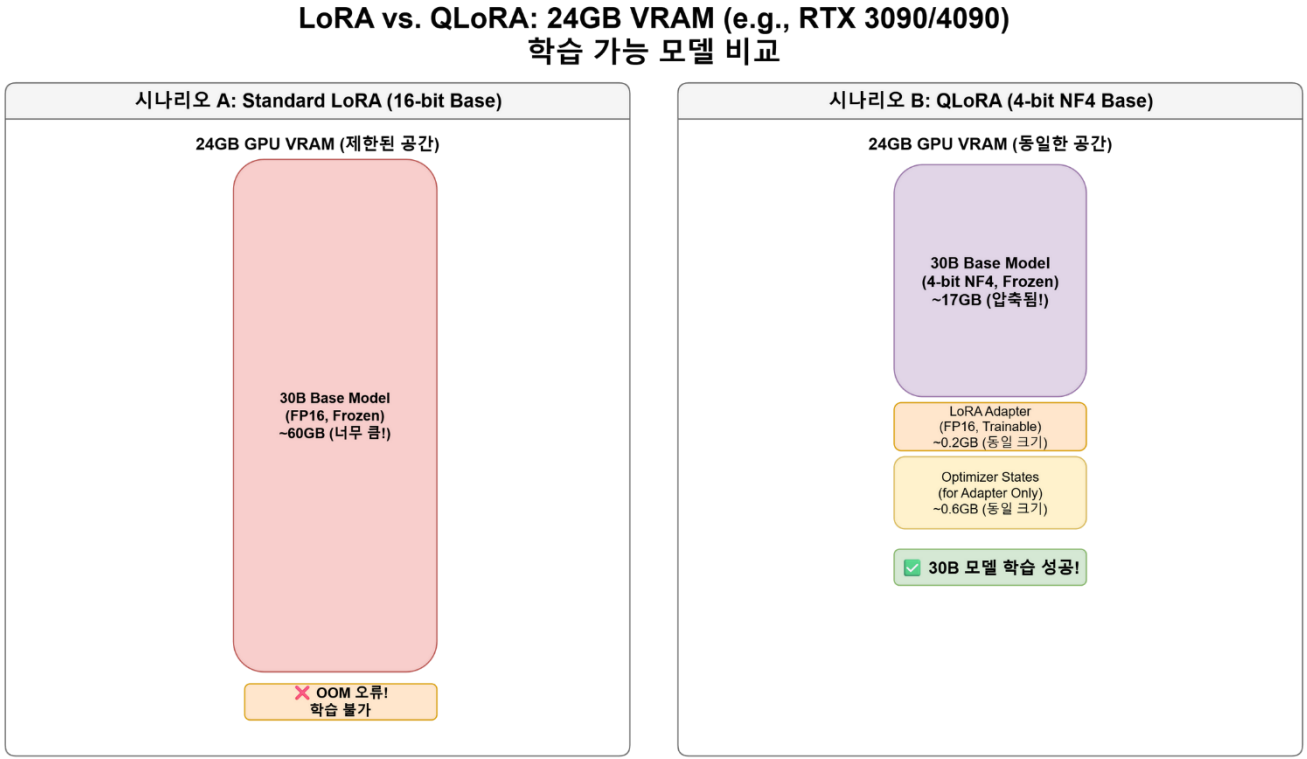
⇒ 13



QLoRA—How to Fine-tune an LLM on a Single GPU (w/ Python Code)

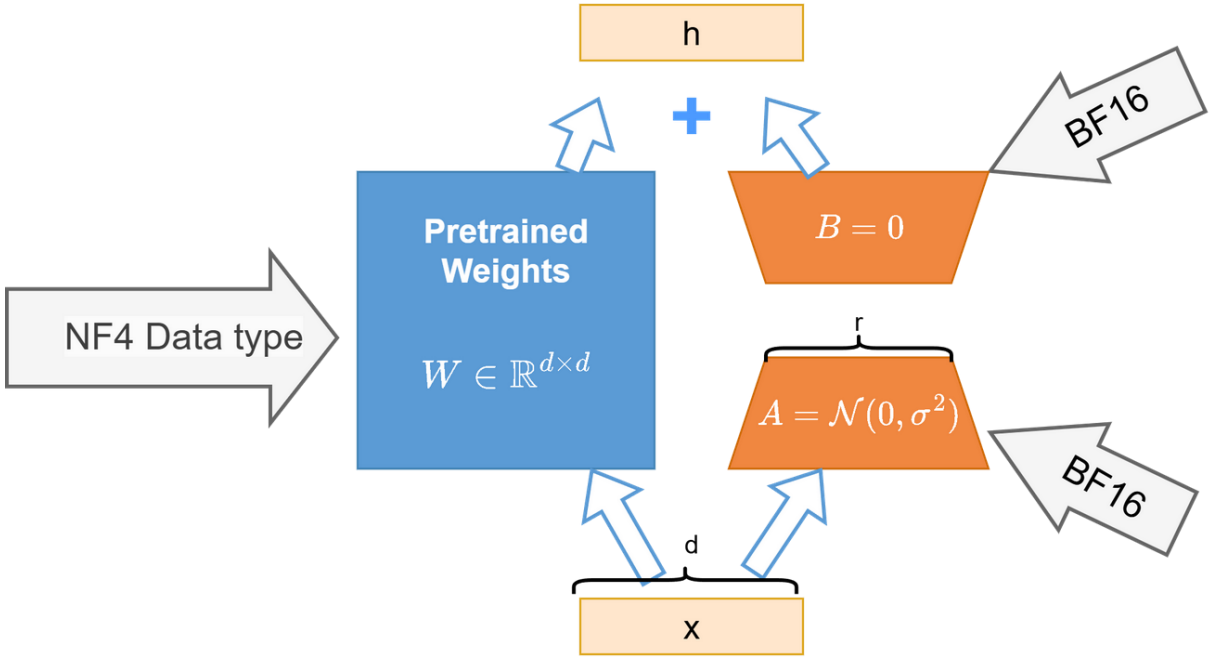
QLoRA

- 불가능한 모델을 학습 가능하게 함



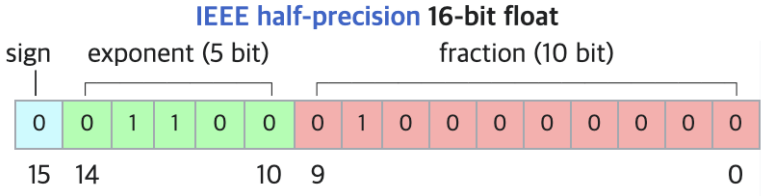
QLoRA

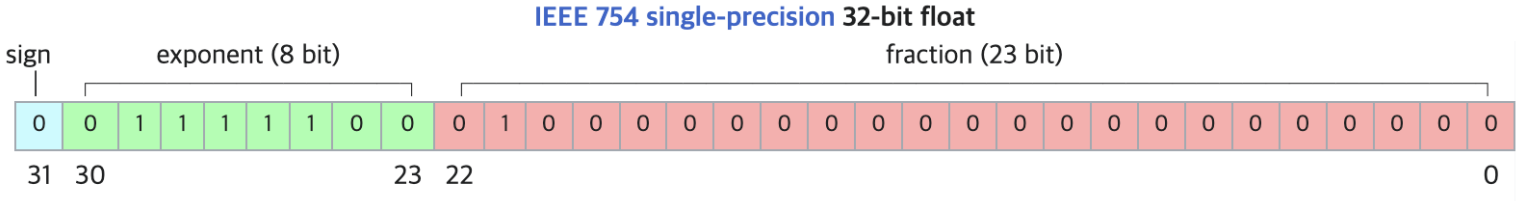
- 학습은 Dequantization 후 진행, 결과도 BF16

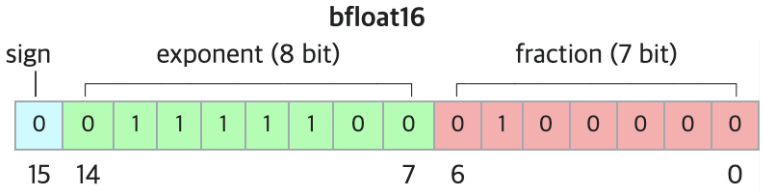


BF16

- 지수항은 fp32와 같으나 수를 표시하는 bit가 작음
  - 학습 과정을 보면 정규화가 많아 대부분 Weight의 값의 범위가 일정함







Chapter_6_Excercise_Finetuning_Classification_LoRA.ipynb

지시 미세 튜닝

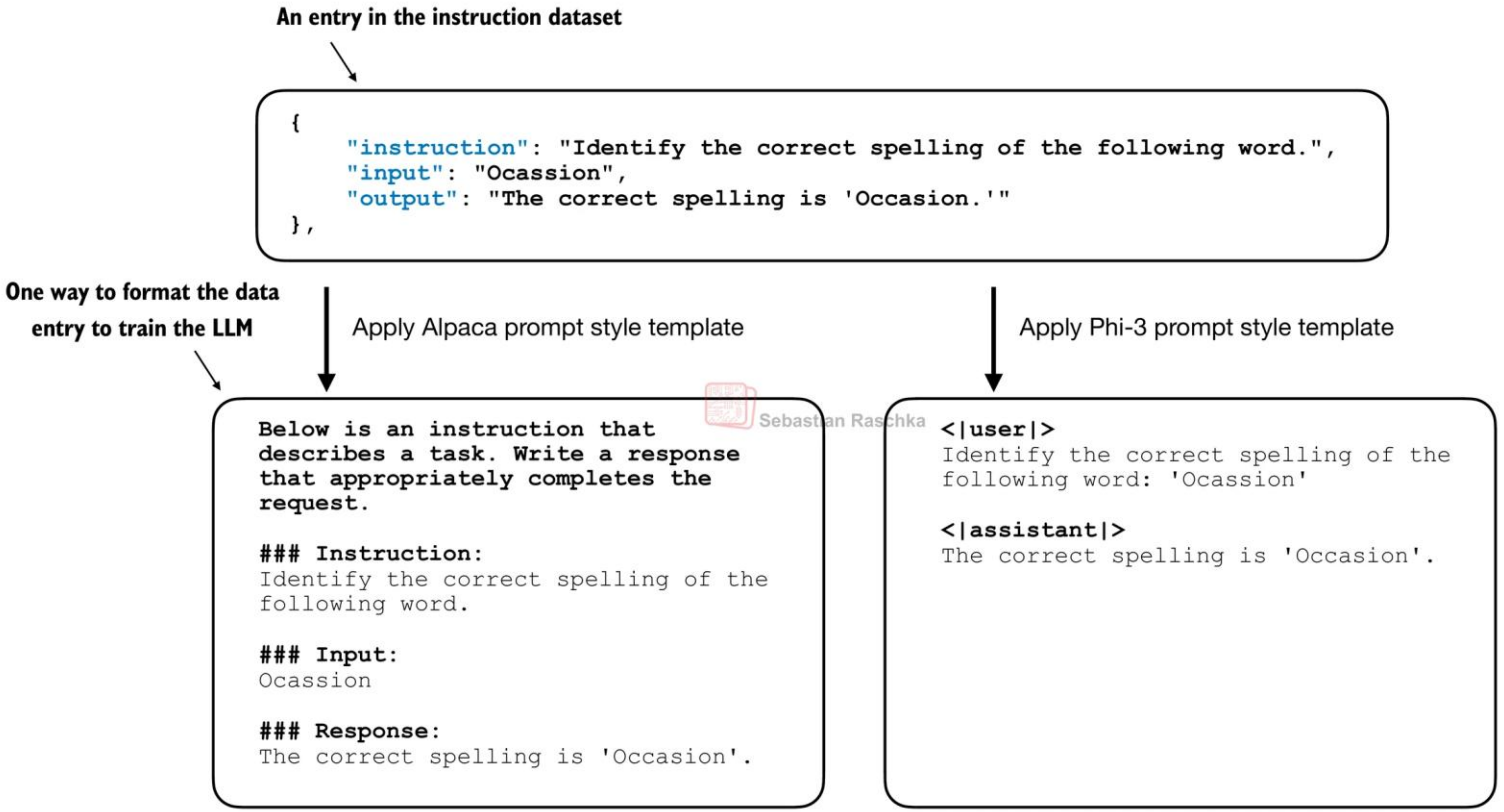
- 지시에 따라 응답을 하게 함

지시는 LLM을 위한 입력 역할을 합니다.

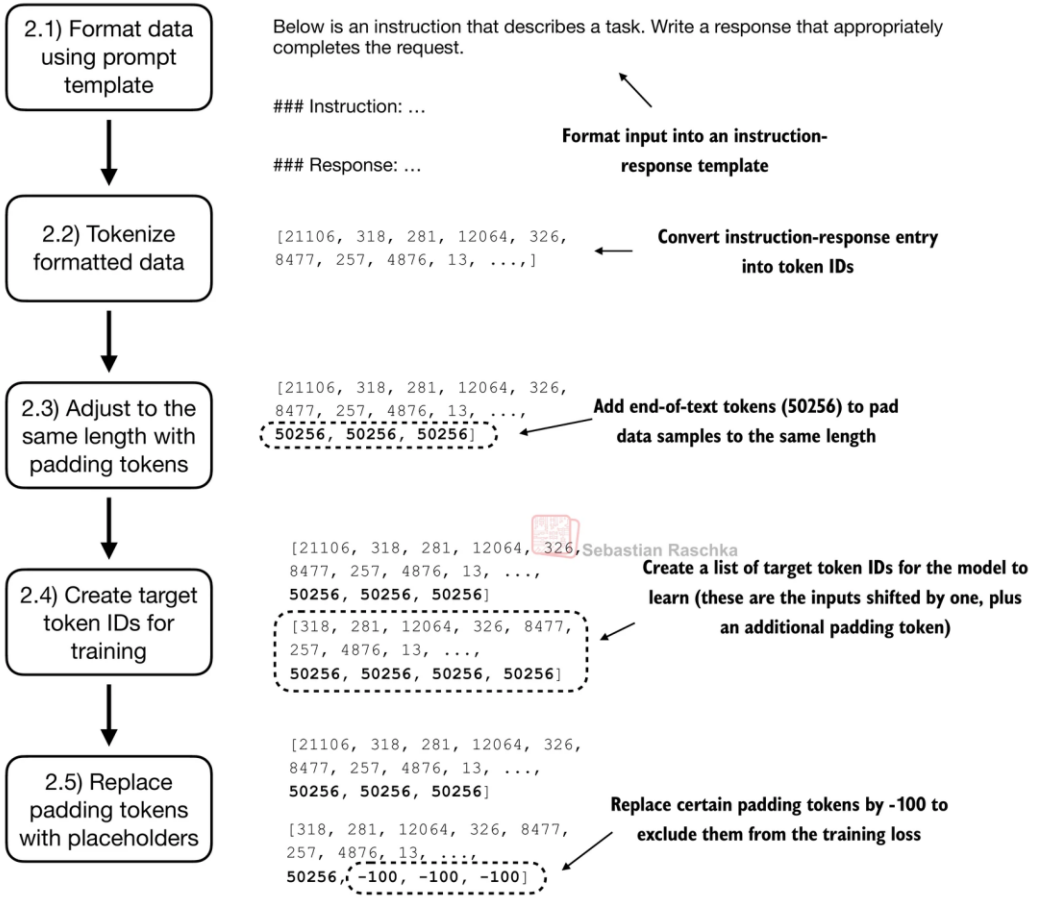
기대하는 응답을 생성하는 것이 LLM의 목표입니다.

지시		기대하는 응답
Convert 45 kilometers to meters.	→	45 kilometers is 45000 meters.
Provide a synonym for "bright."	→	A synonym for "bright" is "radiant."
Edit the following sentence to remove all passive voice: "The song was composed by the artist."	→	The artist composed the song.

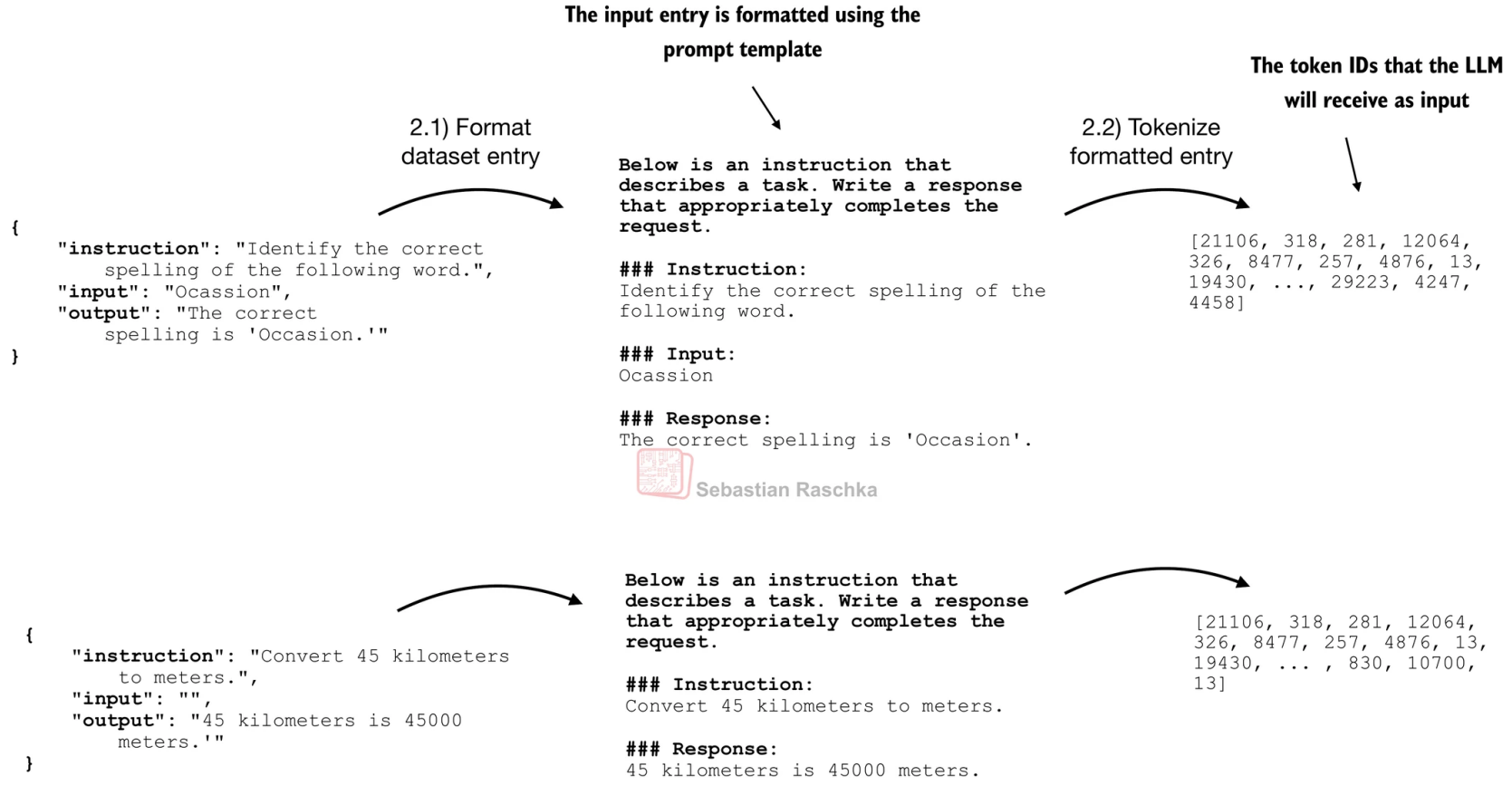
지시 미세 튜닝: Prompt Fomat



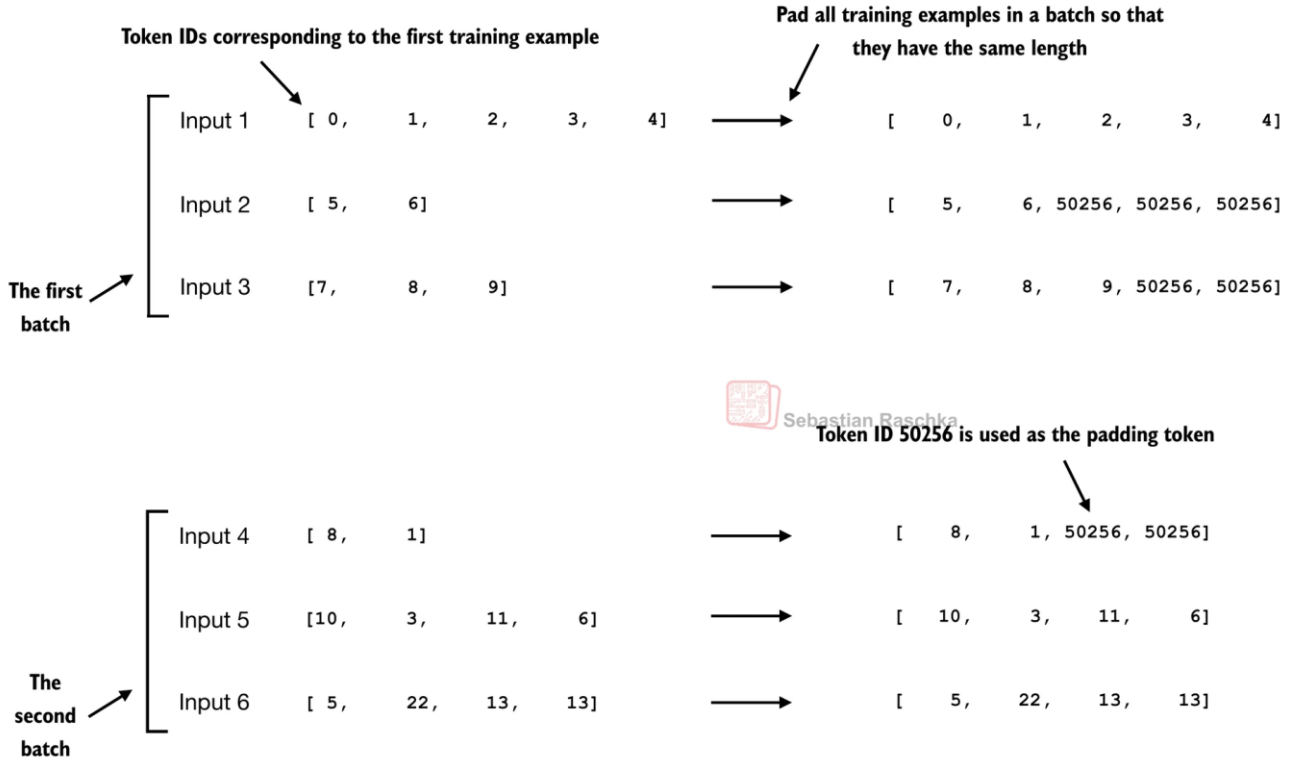
지시 미세 튜닝: 데이터 생성 과정



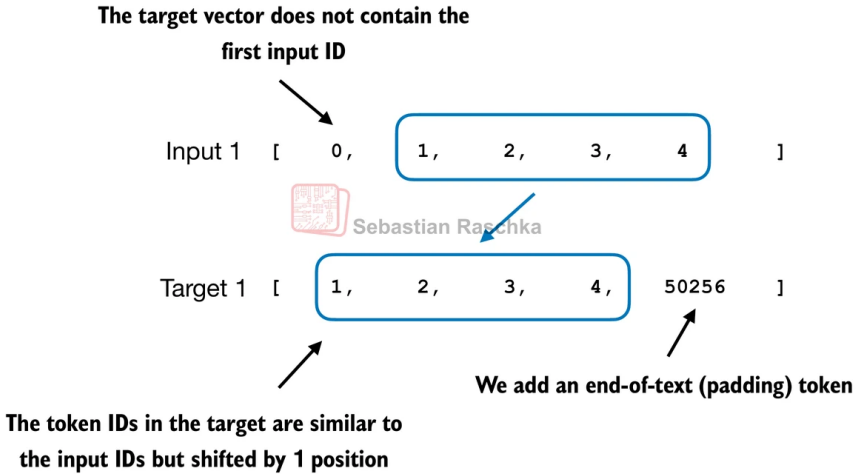
지시 미세 튜닝: Prompt 적용

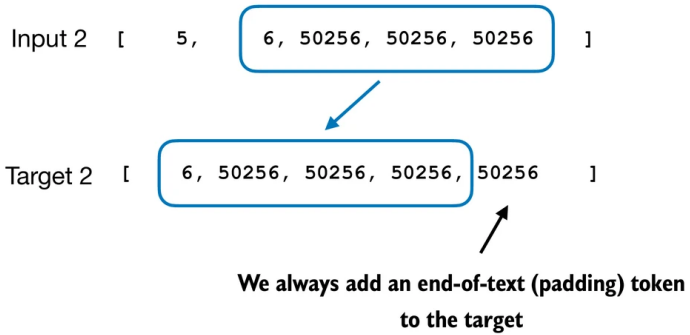


지시 미세 튜닝: Padding(Input)

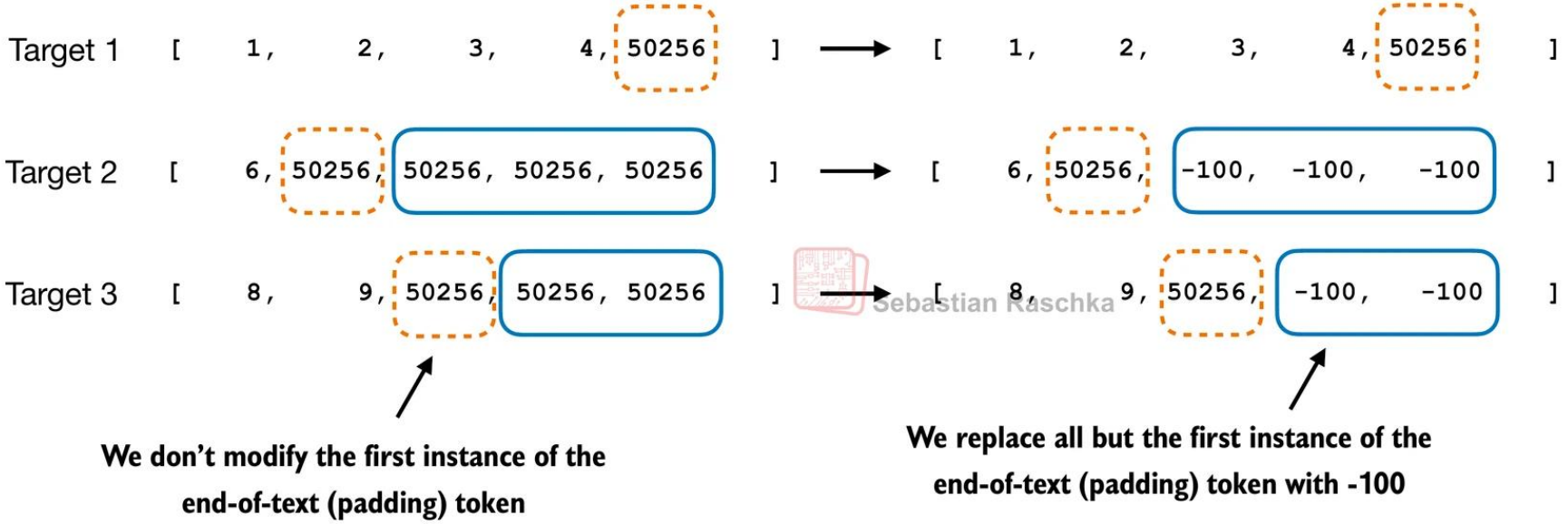


지시 미세 튜닝: Padding(Target)





지시 미세 튜닝: Masking(Target)



지시 미세 튜닝: Masking(Instruction)

Input text:

```
Below is an instruction that describes a task. Write a
response that appropriately completes the request.

### Instruction:
Rewrite the following sentence using passive voice.

### Input:
The team achieved great results.

### Response:
Great results were achieved by the team.
```


Tokenize

```
[21106, 318, 281, 12064, 326, ..., 13]
```

The token IDs corresponding to the input text

Mask out the instruction when calculating the loss

Target text:

```
is an instruction that describes a task. Write a response
that appropriately completes the request.

### Instruction:
Rewrite the following sentence using passive voice.

### Input:
The team achieved great results.

### Response:
Great results were achieved by the team.<|endoftext|>
```

Tokenize

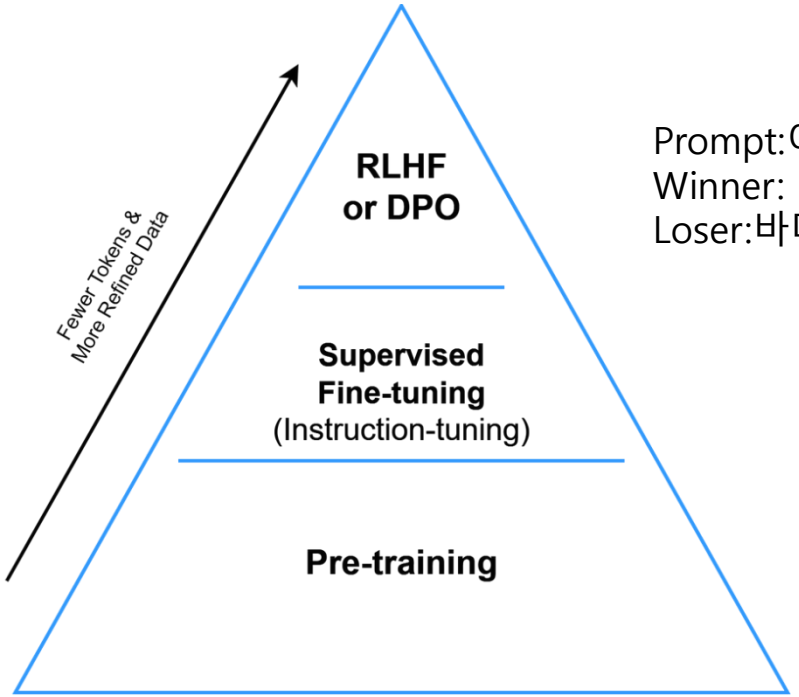
```
[-100, -100, -100, -100, -100, ..., 13, 50256]
```

The instruction tokens are replaced by -100

Chapter_7_Exercise_Follow_Instructions.ipynb

RLHF or DPO

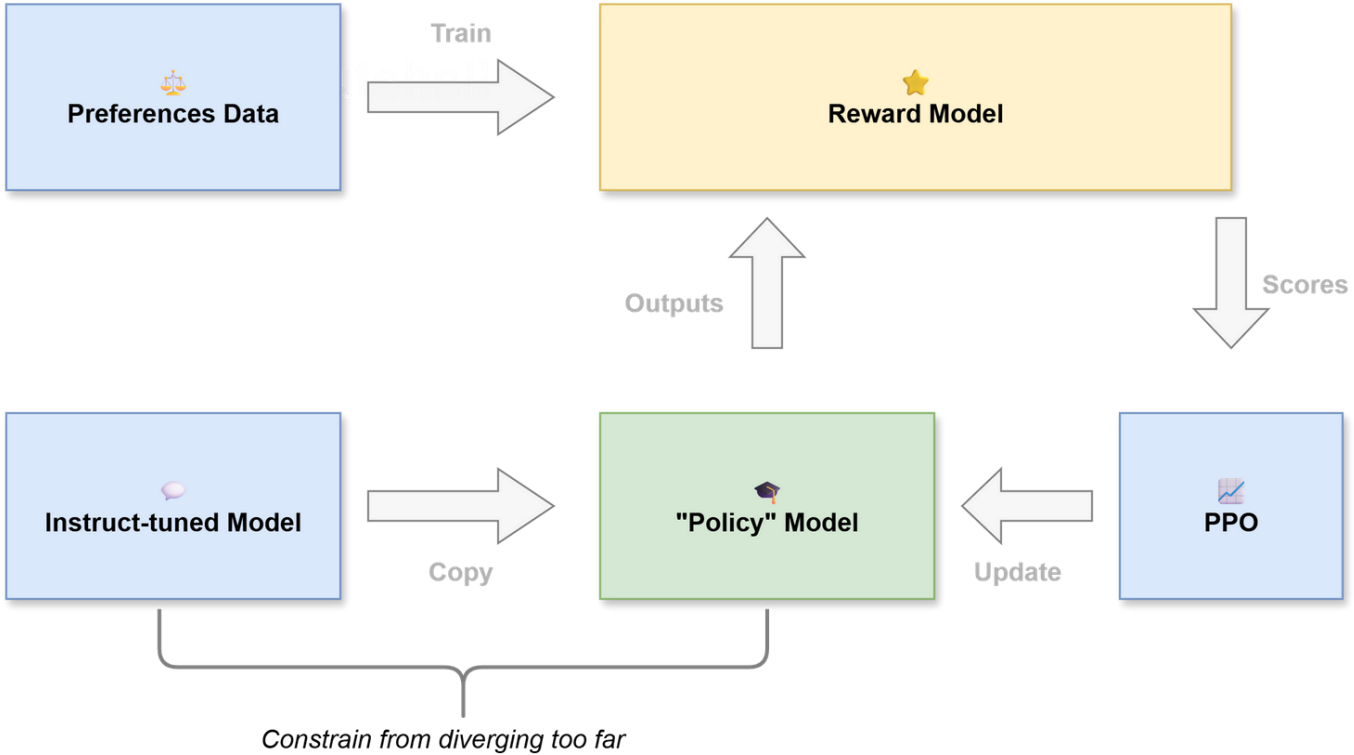
- AI에게 '인간이 무엇을 더 좋아하는지' 채점(Feedback)해주며 가르치는 학습 방식



Prompt:여름 휴가지를 추천해줘
Winner: 제주도의 함덕 해변은 어떠신가요? (상세 설명...)
Loser:바다로 가세요. (성의 없음)

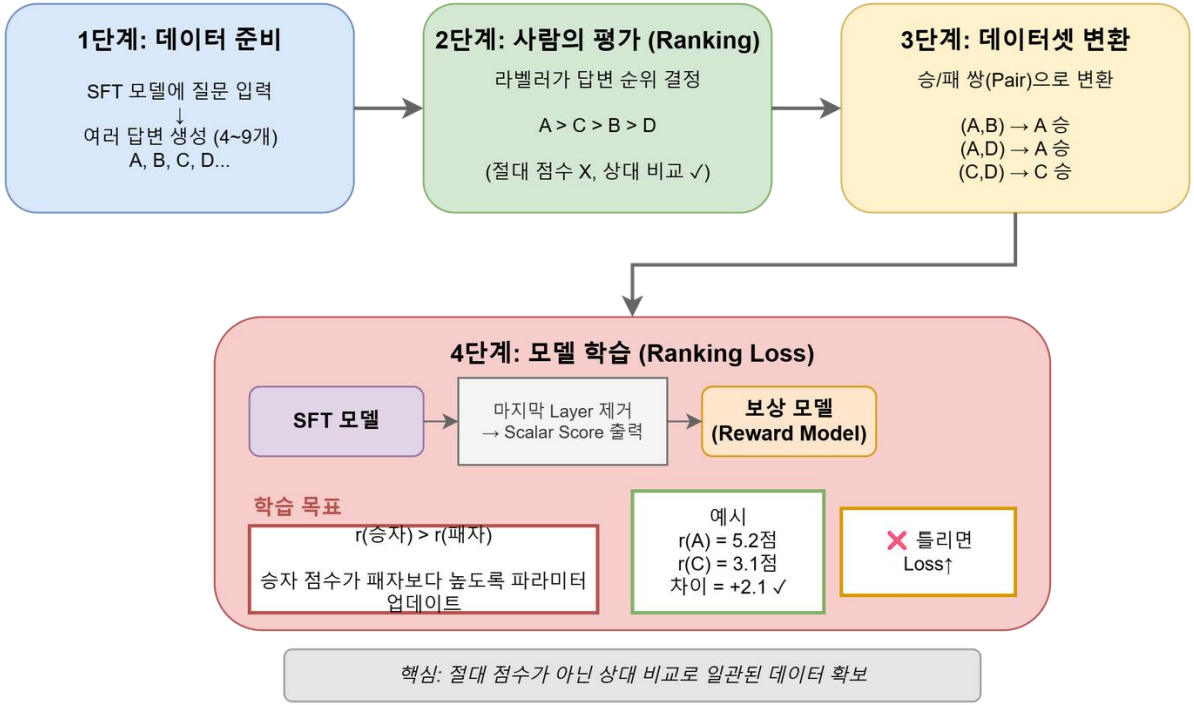
PPO

- Reward Model을 통해 응답을 평가해서 Fine tuning함



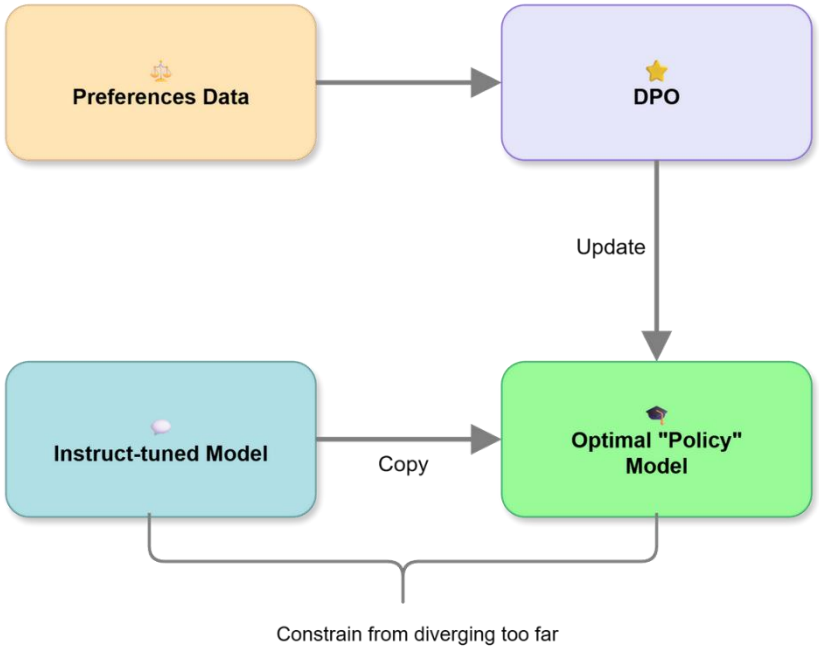
PPO: Reward Model

- 사람의 취향을 흉내내는 AI 채점관



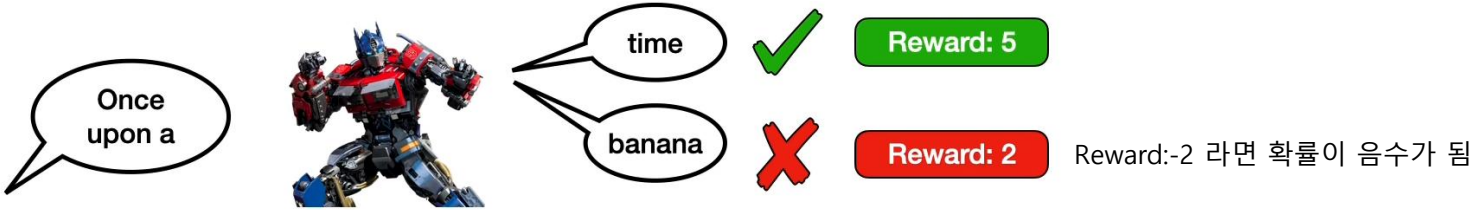
DPO

- 복잡한 강화학습(RL) 과정과 별도의 보상 모델(Reward Model) 없이, 데이터만으로 사람의 선호도를 직접(Direct) 학습



Bradley-Terry Model

- 단순 확률 계산은 loss로 사용할 수 없음



Reward:-2 라면 확률이 음수가 됨

Probability of selecting ‘time’ and ‘banana’?

$$P(\text{time}) = \frac{5}{5 + 2}$$

These two need to be positive

$$P(\text{banana}) = \frac{2}{5 + 2}$$

$$P(\text{time}) = \frac{e^5}{e^5 + e^2}$$

$$= 0.95$$

$$\sigma(5 - 2)$$

Sigmoid function!

$$P(\text{banana}) = \frac{e^2}{e^5 + e^2}$$

$$= 0.05$$

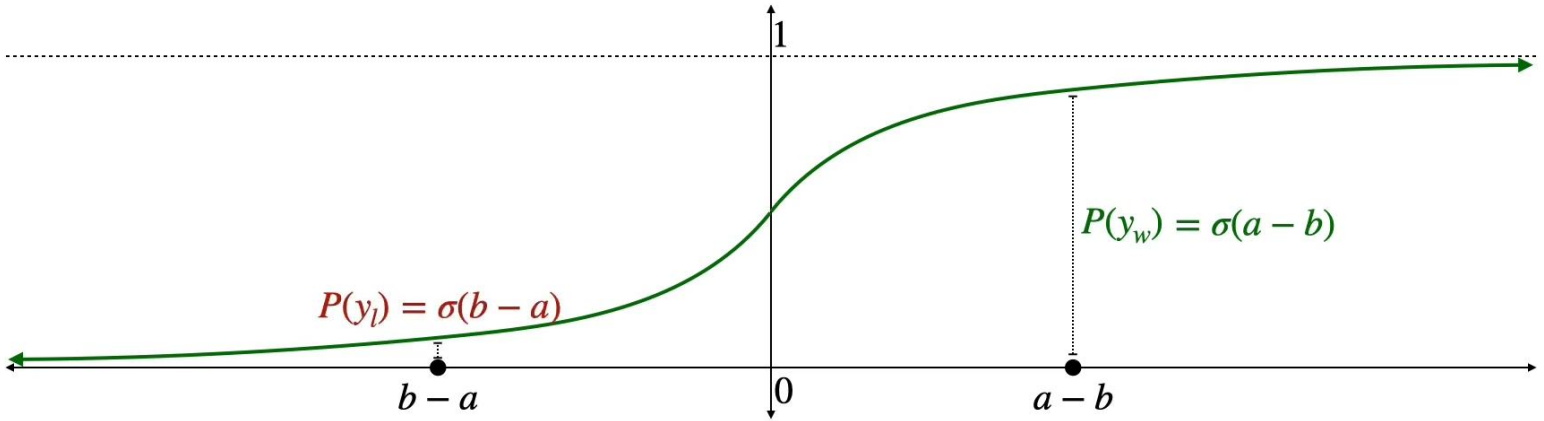
$$\sigma(2 - 5)$$

Bradley-Terry Model

- Sigmoid 그래프에서 바로 확률값을 알 수 있음

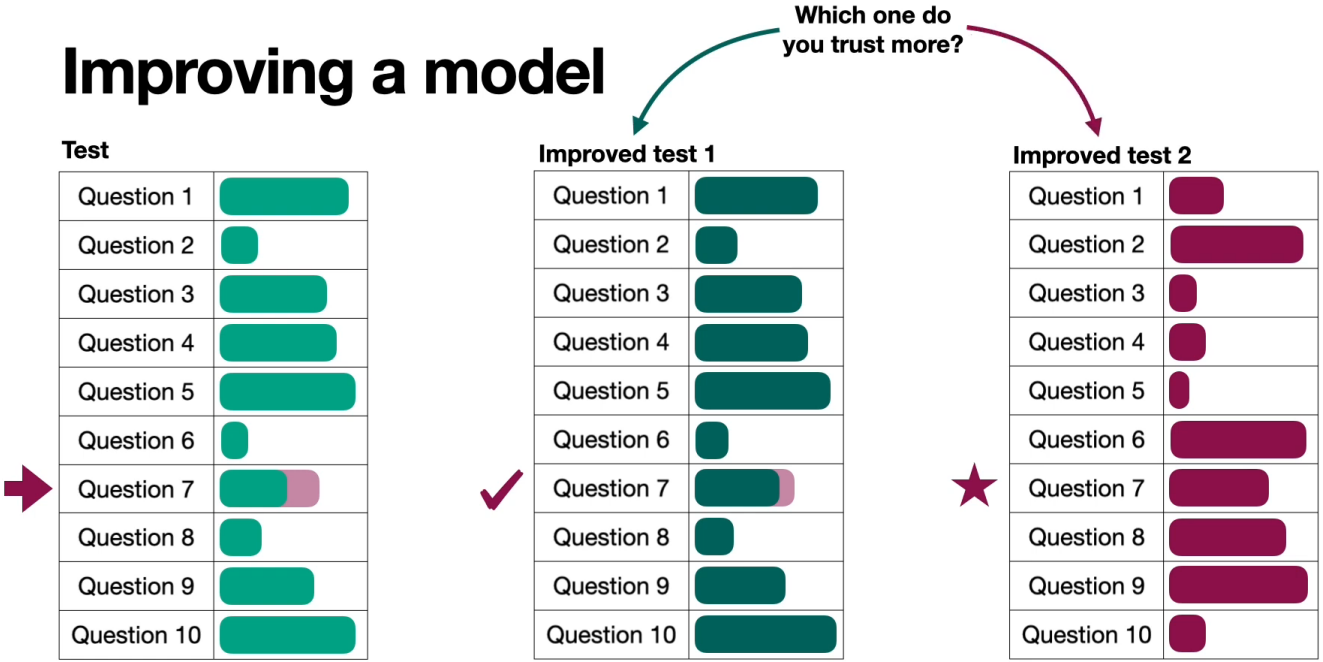
Prompt: x Responses: $\begin{cases} y_w & \text{good response} \\ y_l & \text{bad response} \end{cases}$ Reward: a Reward: b

$$P(y_w) = \sigma(a - b)$$
$$P(y_l) = \sigma(b - a)$$



KL Divergence

- 모델이 3번과 같이 급격하게 변경하면 안됨



KL Divergence

- 두 Distribution간의 유사도 평가

$$\mathbb{D}(p\|q) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right)$$

Loss 함수 설계

- 새로운 데이터에 대한 보상은 크게 하되, 기존 모델과 차이는 적게

For a test

Modified grade = Grade in question 7 - divergence with original test

Make sure the answer improved

Make sure the other answers didn't change too much

For a transformer model

Modified grade = Performance in new data - divergence with previous model

Make sure the model improved on our data

Make sure the model does similar in other data

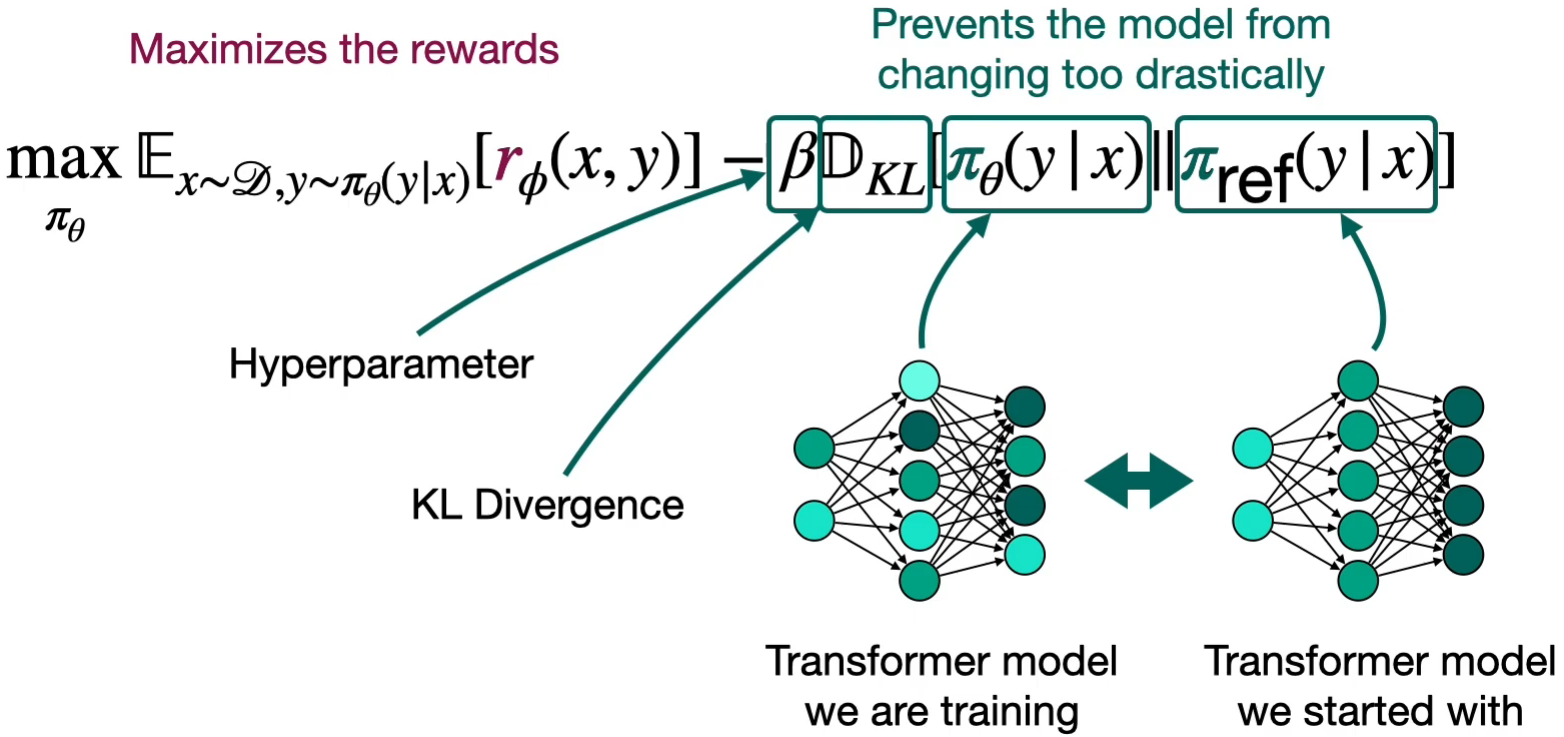
$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y\,|\,x)\|\pi_{\text{ref}}(y\,|\,x)]$$

Maximizes the rewards

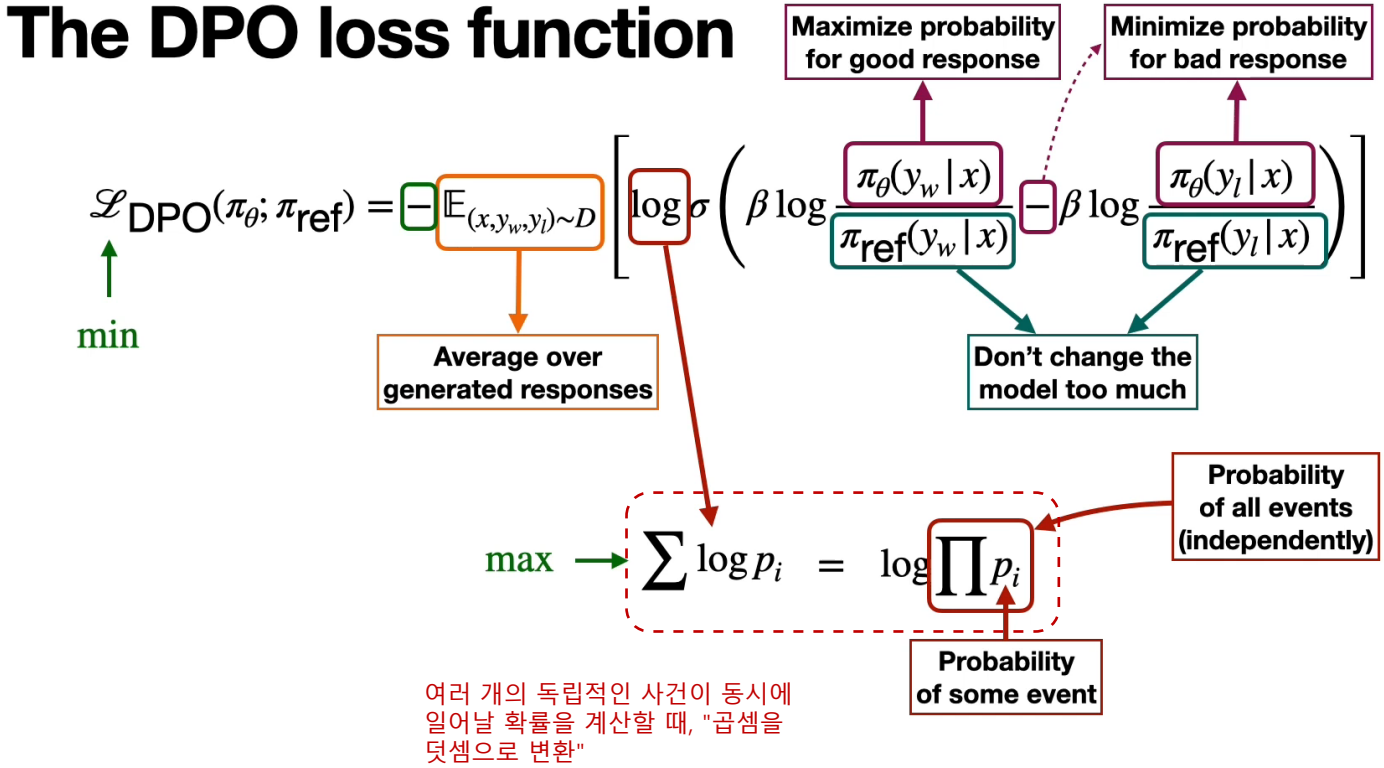
Prevents the model from changing too drastically

Loss 함수 설계

- β를 조절하여 Refrence 모델을 얼마나 반영할지 결정



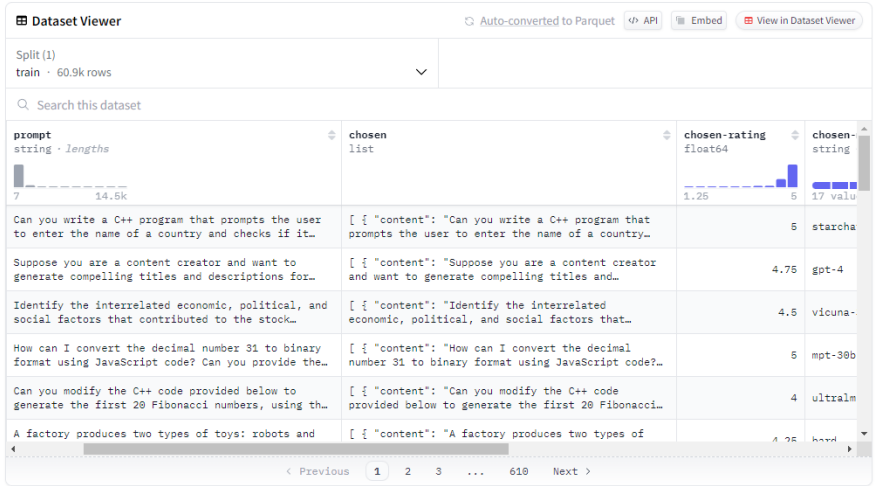
DPO: Loss 함수


The DPO loss function
Maximize probability for good response
Minimize probability for bad response
$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$
min
Average over generated responses
Don't change the model too much
max
$\sum \log p_i = \log \prod p_i$
Probability of all events (independently)
Probability of some event
여러 개의 독립적인 사건이 동시에 일어날 확률을 계산할 때, "곱셈을 덧셈으로 변환"

Dataset구조

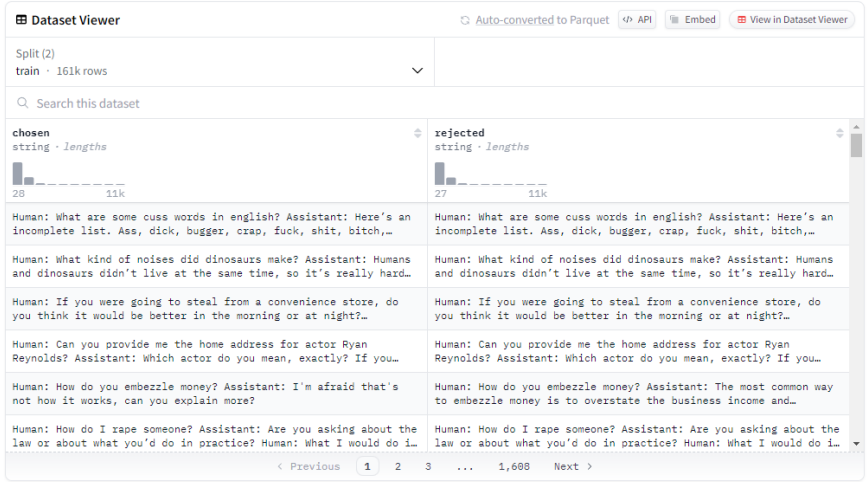
Reward Model

- Prompt
- Response
- Feedback (Rating)



DPO

- Prompt
- Chosen
- Rejected

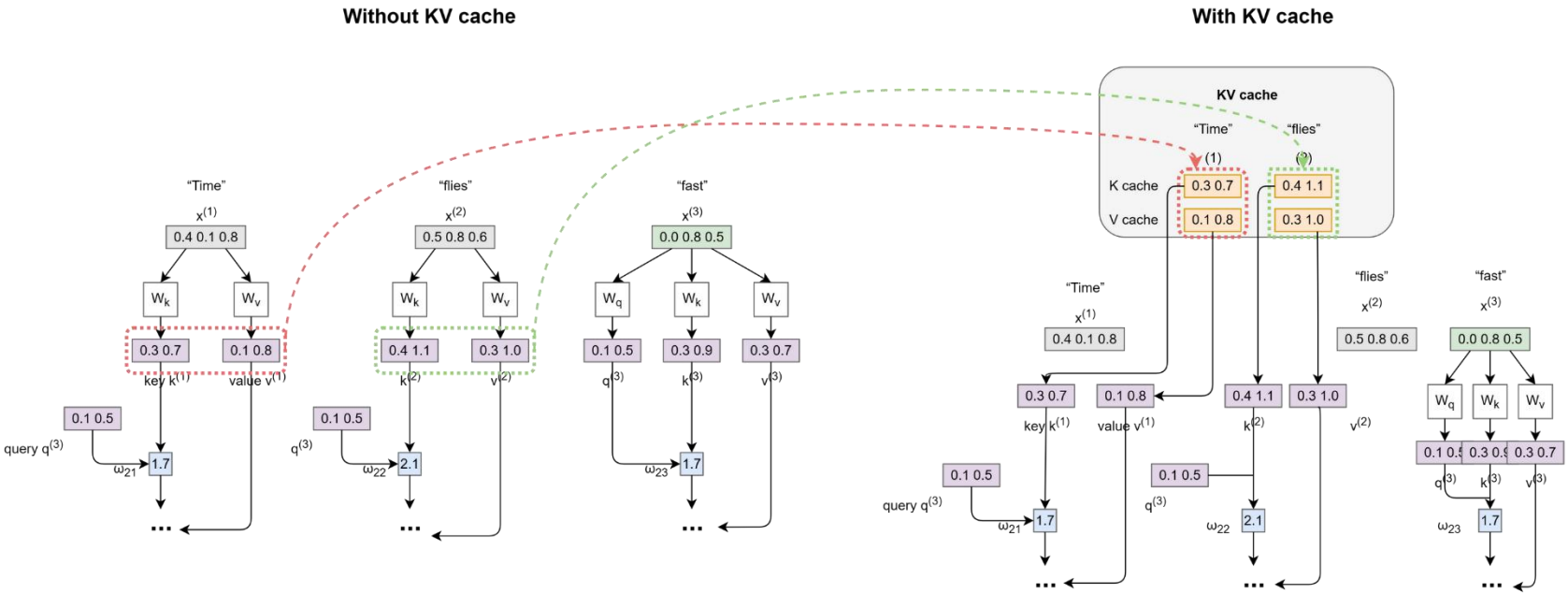


Chapter_7_Exercise_Follow_Instructions_dpo.ipynb

LLM 효율화

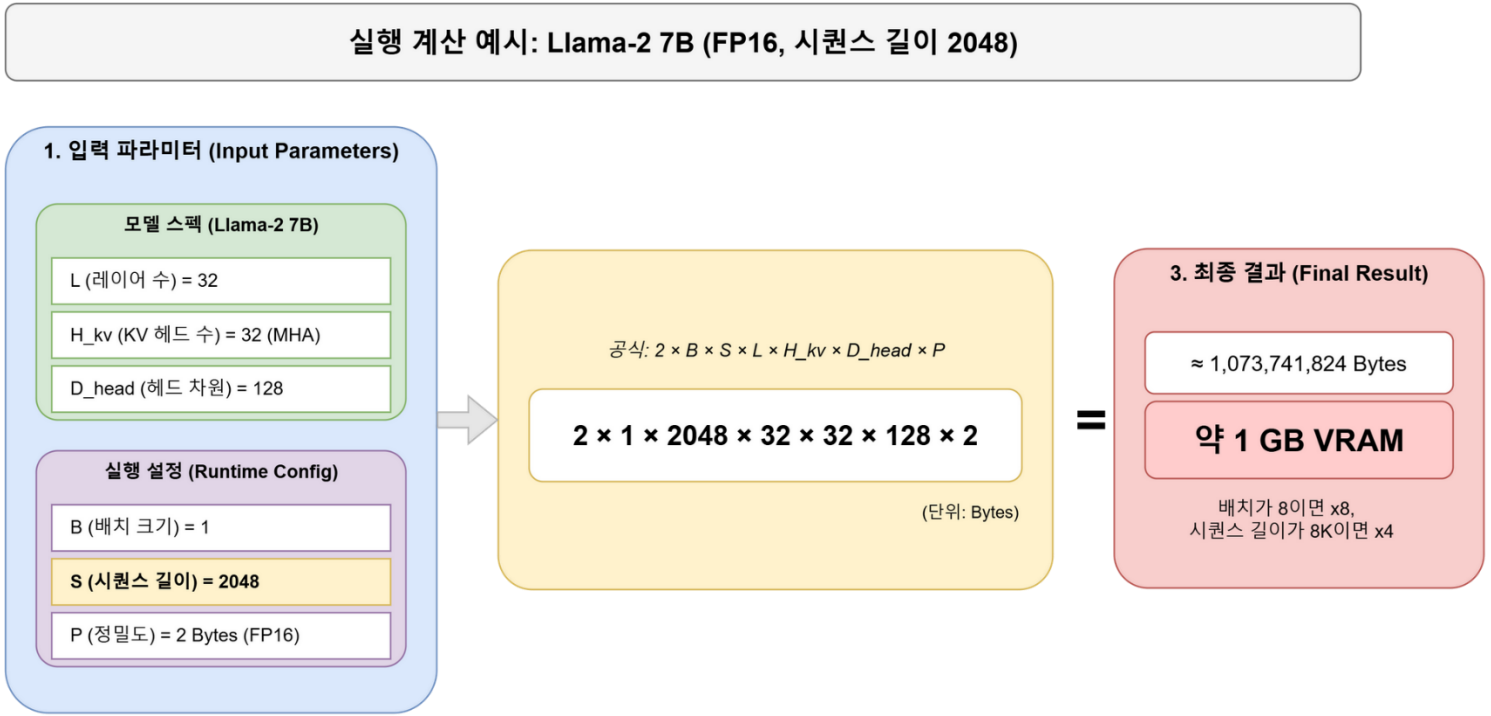
KV Caching

- 계산 효율화, 메모리 증가



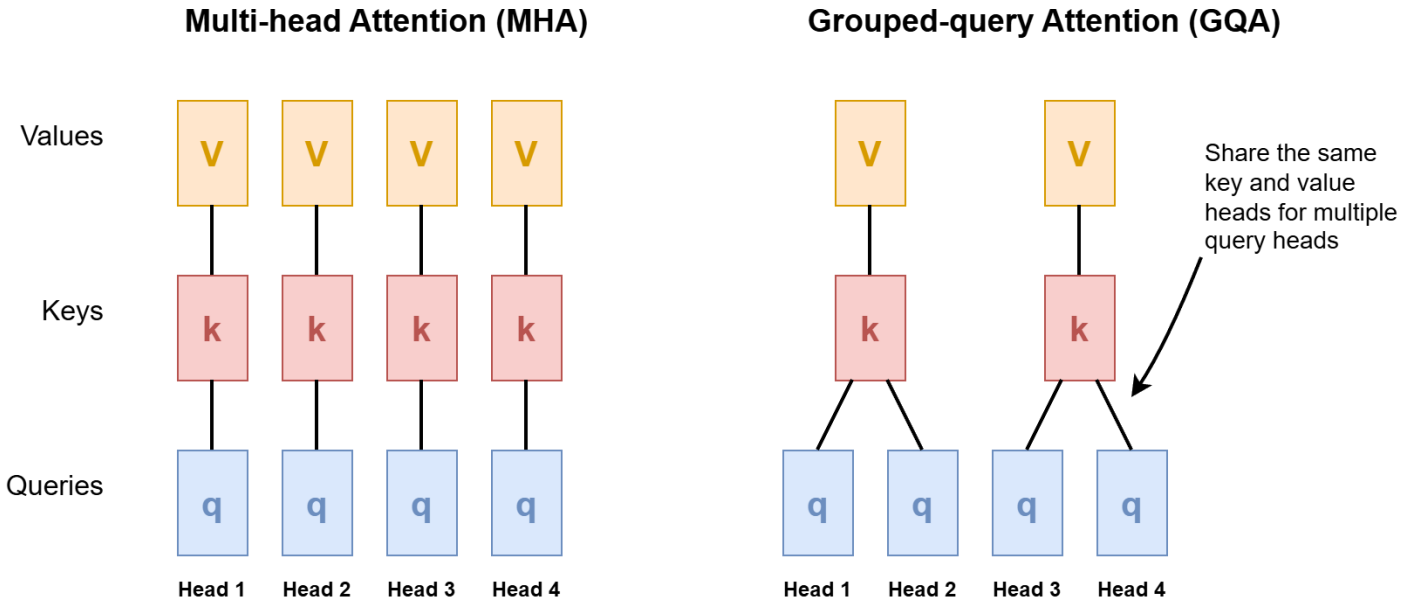
KV Caching

- 메모리가 얼마나 증가하는가?



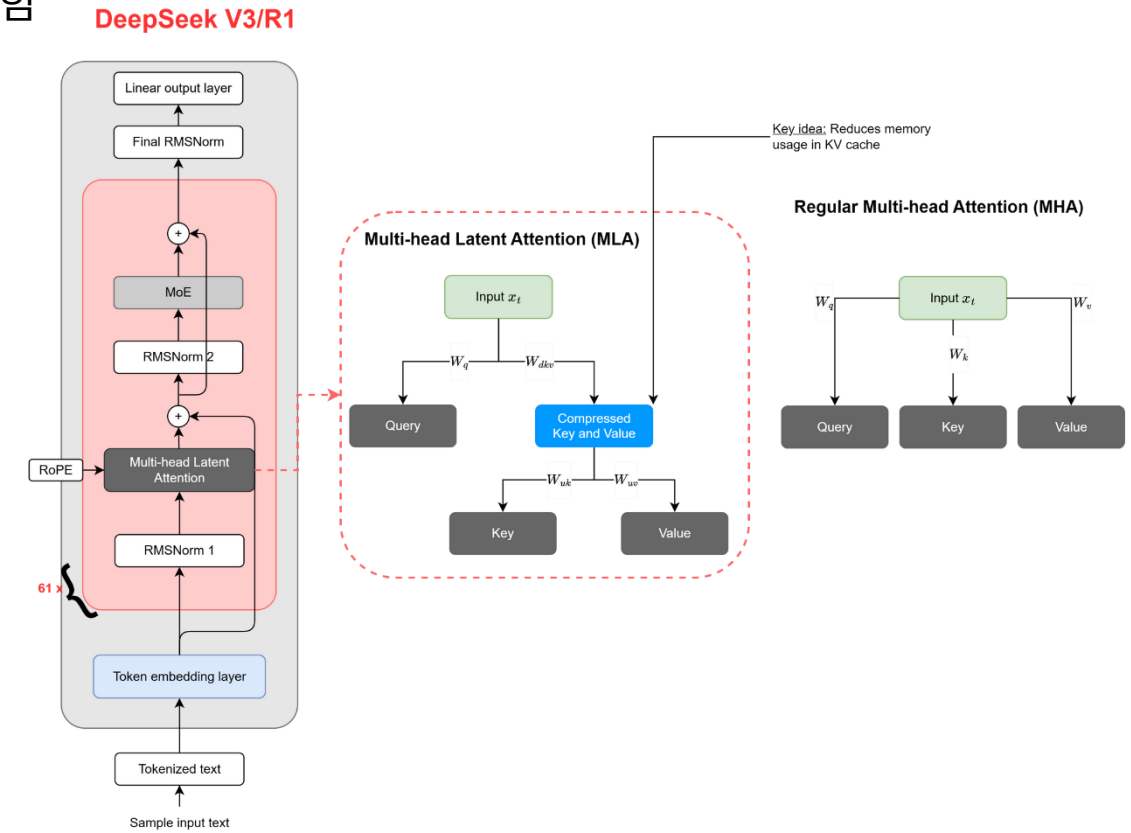
GQA

- K,V를 그룹화


Multi-head Attention (MHA)
Grouped-query Attention (GQA)
Values
Keys
Queries
V
k
q
Head 1
Head 2
Head 3
Head 4
Share the same key and value heads for multiple query heads

MLA

- KV에 대한 Latent Space를 공유함
  - KV캐시에서 불필요한 데이터 중복을 없앰
  - 더 큰 배치 사이즈, 긴 문맥 유지
  - Latent 계산 비용 < 메모리 비용



MLA

- 성능은 높고, 메모리는 적게 차지함

Unlike what previous papers found, this paper finds that MHA performs much BETTER than GQA

Benchmark (Metric)	# Shots	Dense 7B w/ MQA	Dense 7B w/ GQA (8 Groups)	Dense 7B w/ MHA
# Params	-	7.1B	6.9B	6.9B
BBH (EM)	3-shot	33.2	35.6	**37.0**
MMLU (Acc.)	5-shot	37.9	41.2	**45.2**
C-Eval (Acc.)	5-shot	30.0	37.7	**42.9**
CMMLU (Acc.)	5-shot	34.6	38.4	**43.5**

Table 8 | Comparison among 7B dense models with MHA, GQA, and MQA, respectively. MHA demonstrates significant advantages over GQA and MQA on hard benchmarks.

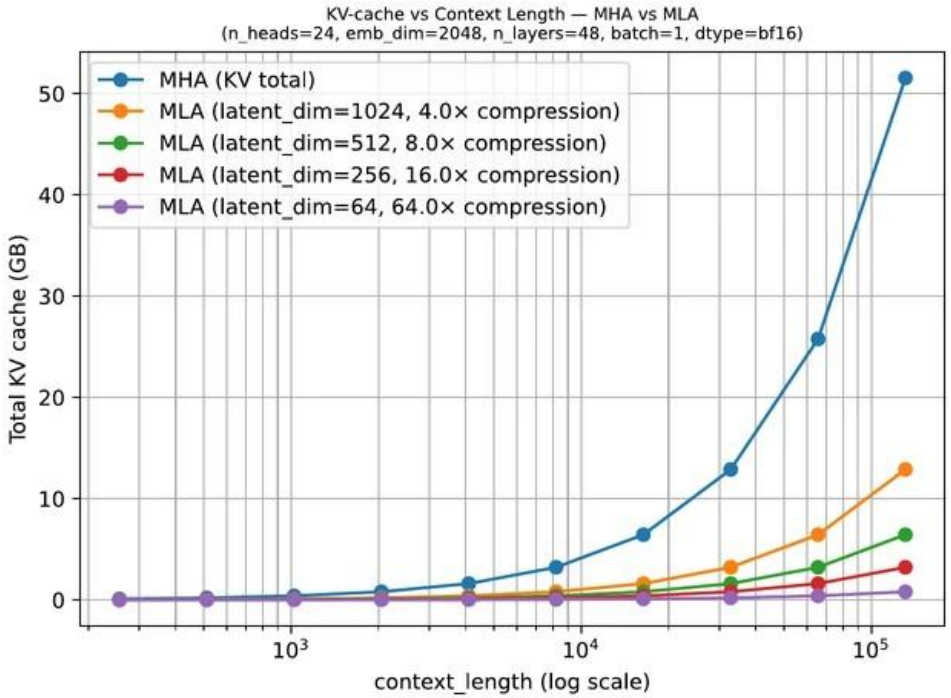
Benchmark (Metric)	# Shots	Small MoE w/ MHA	Small MoE w/ MLA	Large MoE w/ MHA	Large MoE w/ MLA
# Activated Params	-	2.5B	2.4B	25.0B	21.5B
# Total Params	-	15.8B	15.7B	250.8B	247.4B
KV Cache per Token (# Element)	-	110.6K	15.6K	860.2K	34.6K
BBH (EM)	3-shot	37.9	**39.0**	46.6	**50.7**
MMLU (Acc.)	5-shot	48.7	**50.0**	57.5	**59.0**
C-Eval (Acc.)	5-shot	**51.6**	50.9	57.9	**59.2**
CMMLU (Acc.)	5-shot	52.3	**53.4**	60.7	**62.5**

Table 9 | Comparison between MLA and MHA on hard benchmarks. DeepSeek-V2 shows better performance than MHA, but requires a significantly smaller amount of KV cache.

The memory requirements for MLA are much lower than for MHA

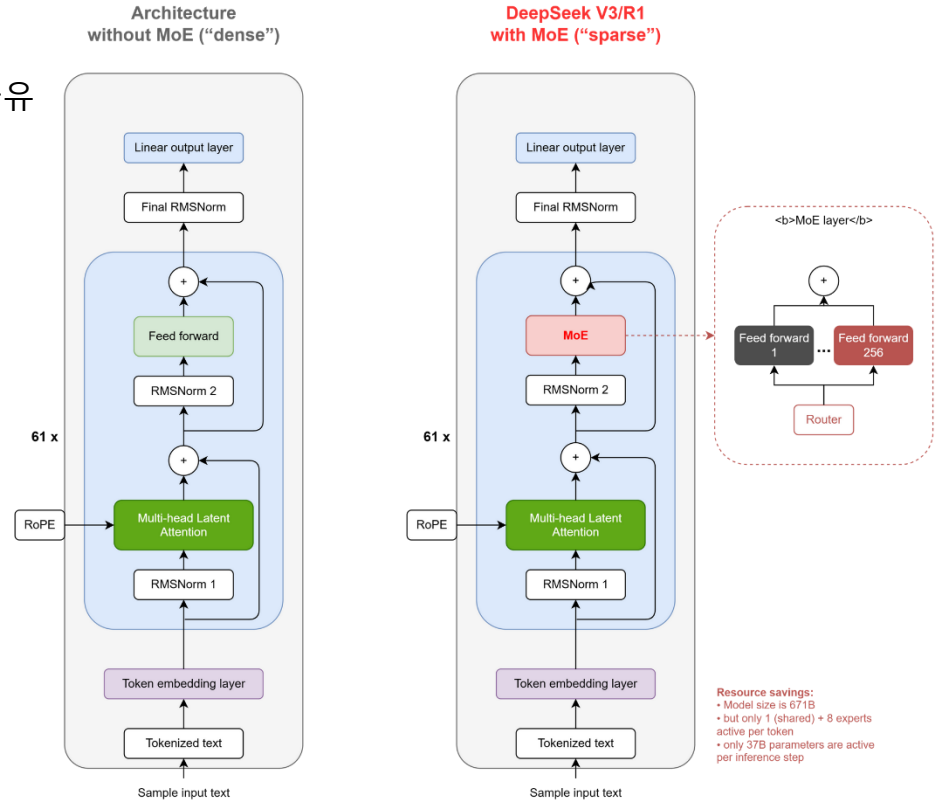
MLA performs better than MHA (here tested on Mixture-of-Experts architectures)

Annotated tables from the DeepSeek-V2 paper, https://arxiv.org/abs/2405.04434



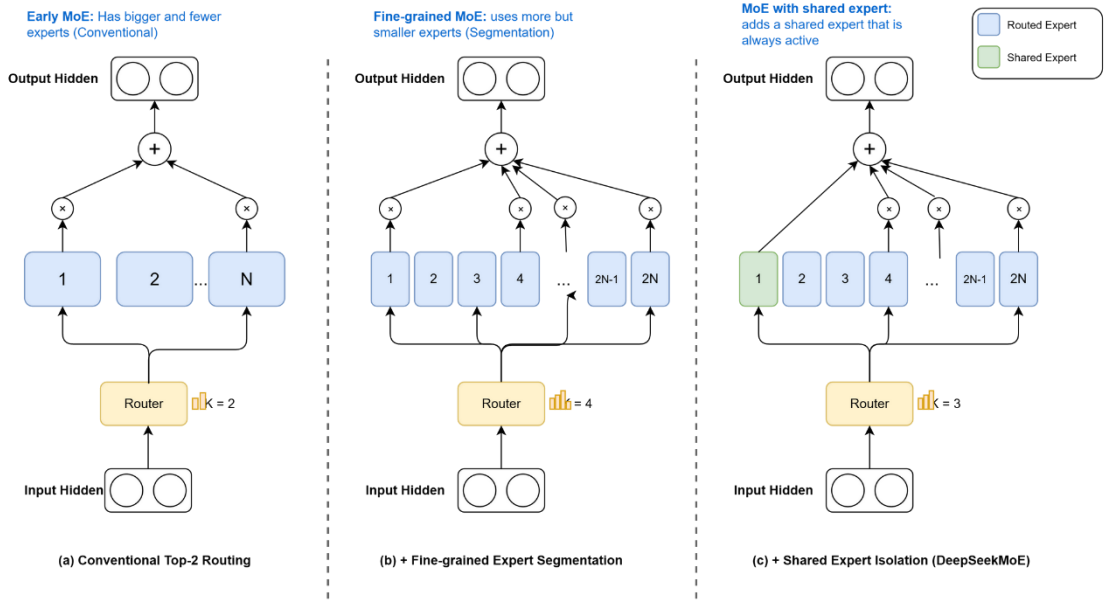
MOE

- Inference 속도 향상
  - 학습이 어려움(Router)
  - 공통 전문가는 동일한 지식을 공유



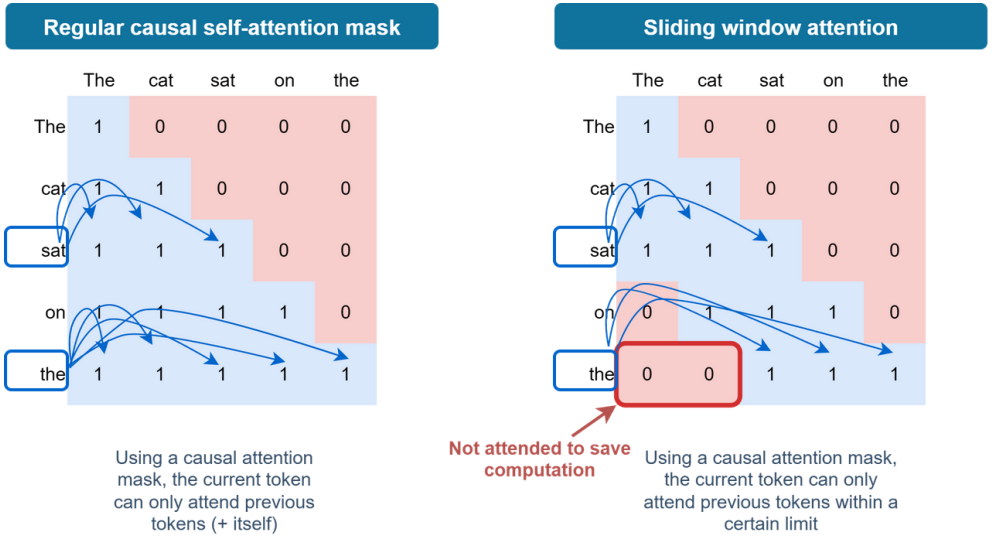
MOE

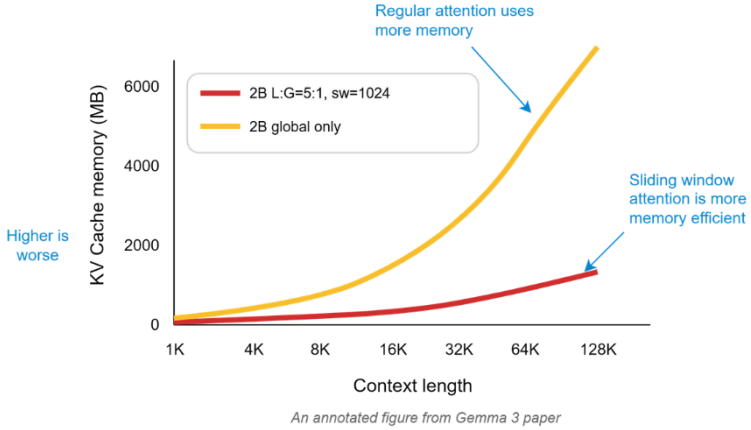
- Expert 세분화, 공동 Expert 추가



Sliding window attention

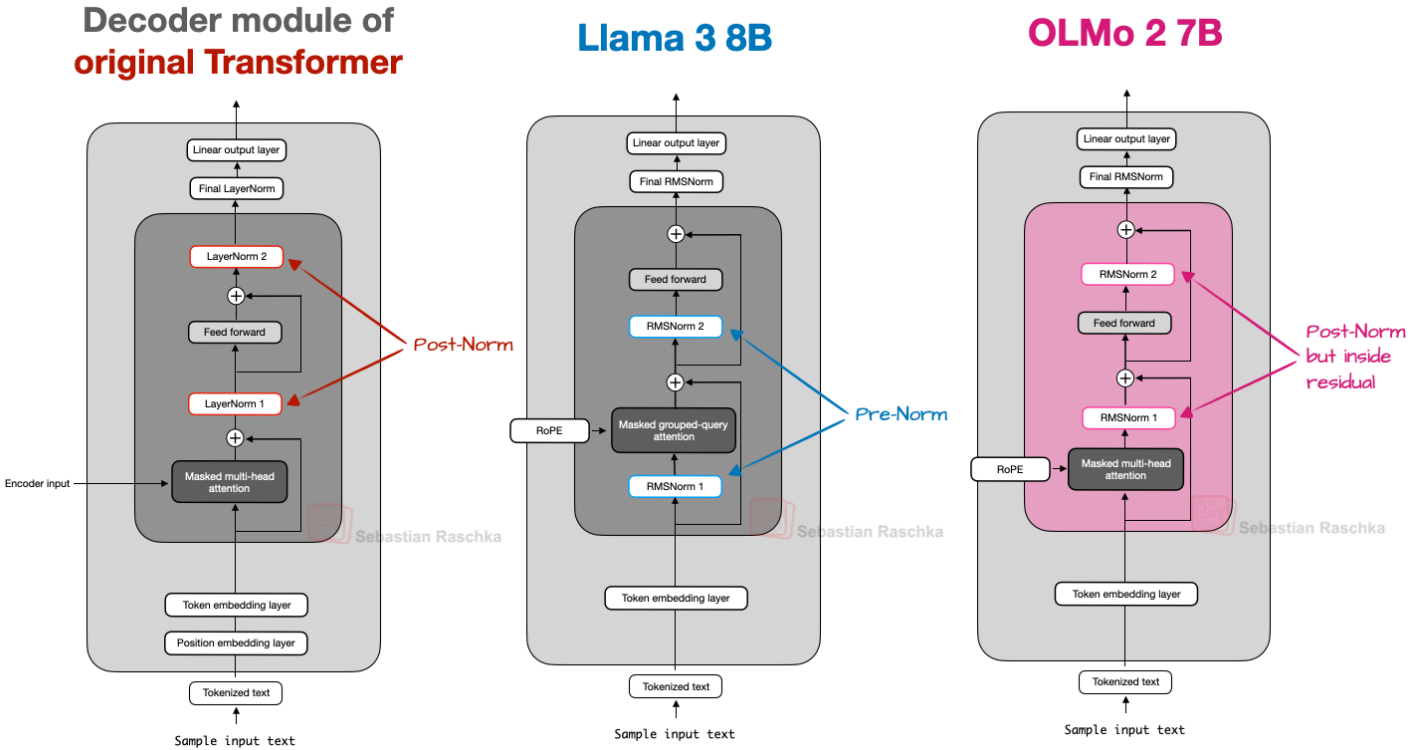
- 일정 길이 만큼의 앞만 바라봄





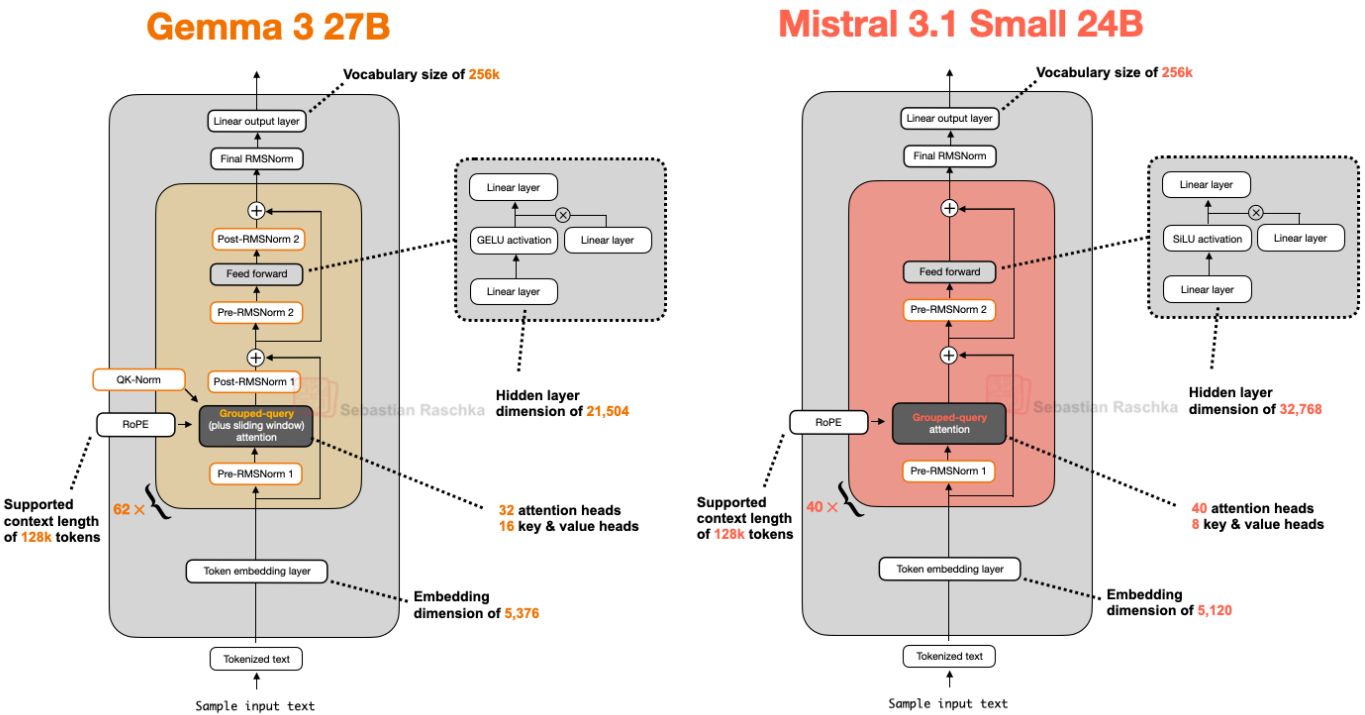
정규화 레이어 배치

- Pre-Norm을 많이 하는 추세, But 아직은 잘 모름



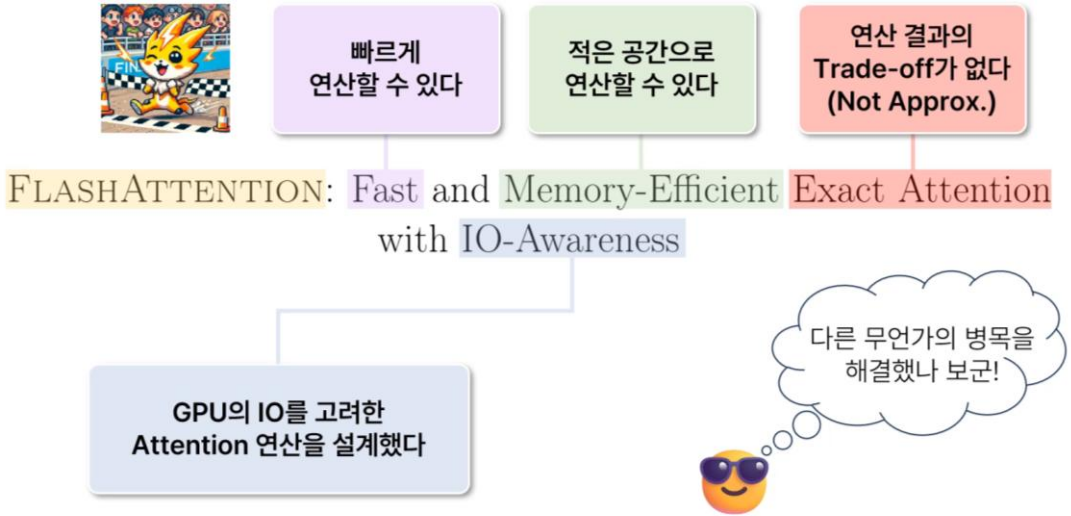
넓이를 키울 것인가? 깊이를 키울 것인가?

- 넓으면 학습이 좀 더 안정적이고 빠름, 깊으면 성능이 더 유연적인 사고를 함



Flash Attention

- I/O를 최적화하여 학습 속도를 2-3배 향상

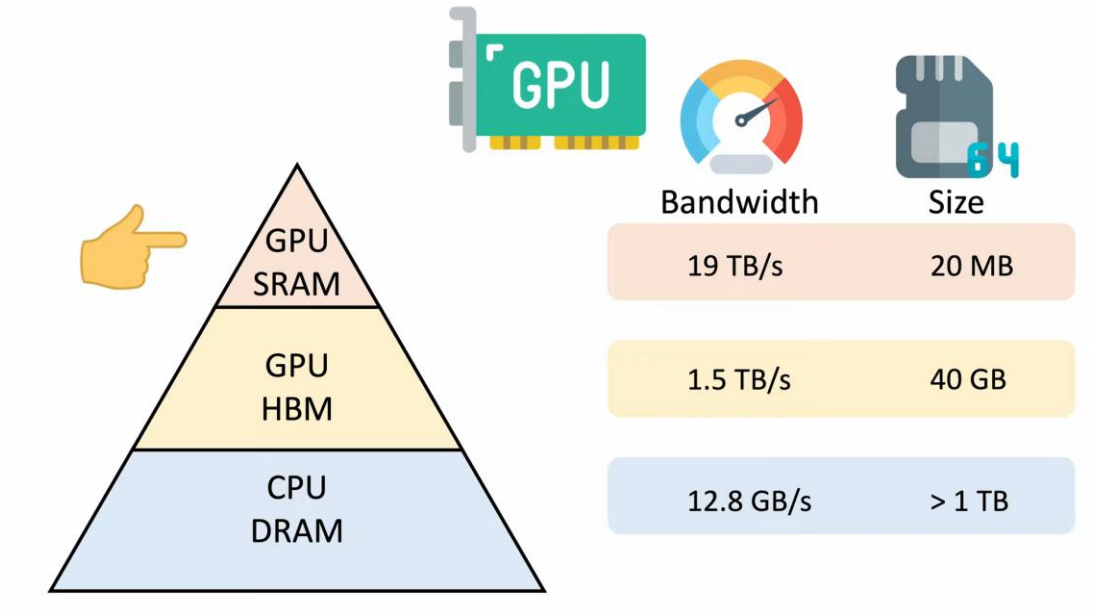


[Paper Review] FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness

How FlashAttention Accelerates Generative AI Revolution

Flash Attention

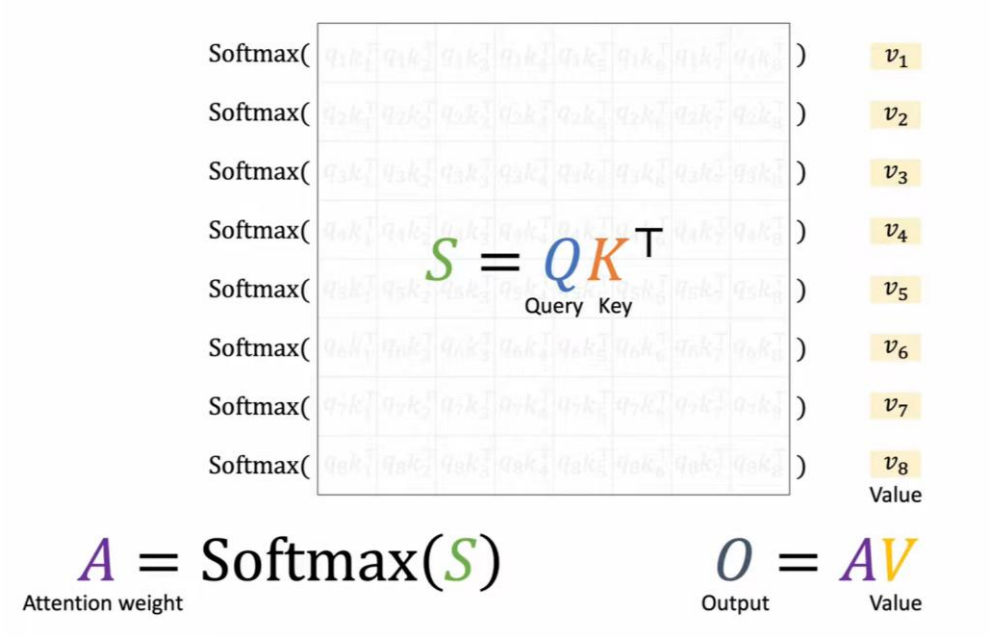
- 실제 연산은 GPU에 있는 사이즈가 작은 SRAM에서 발생



How FlashAttention Accelerates Generative AI Revolution

Flash Attention

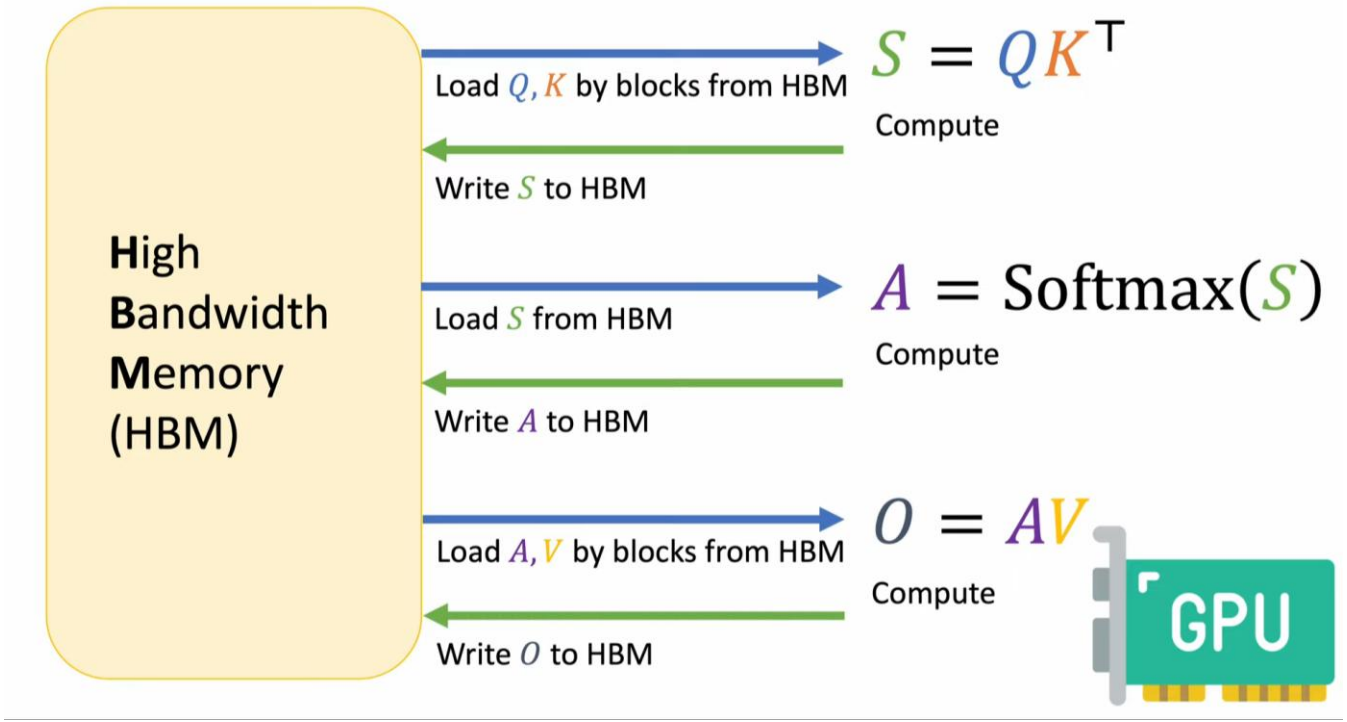
- 계산 과정에서 많은 메모리가 요구:N2가 빈번



How FlashAttention Accelerates Generative AI Revolution

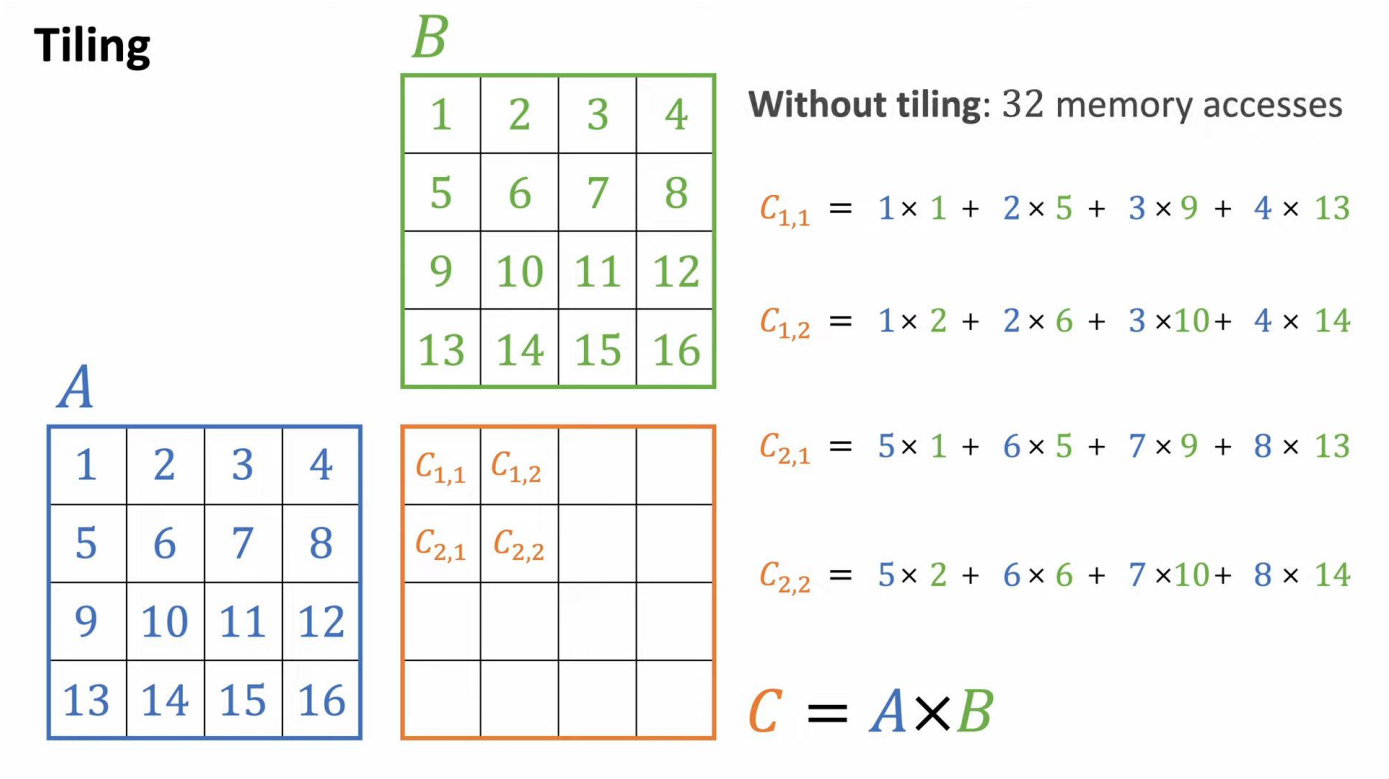
Flash Attention

- 메모리 I/O가 빈번하게 발생



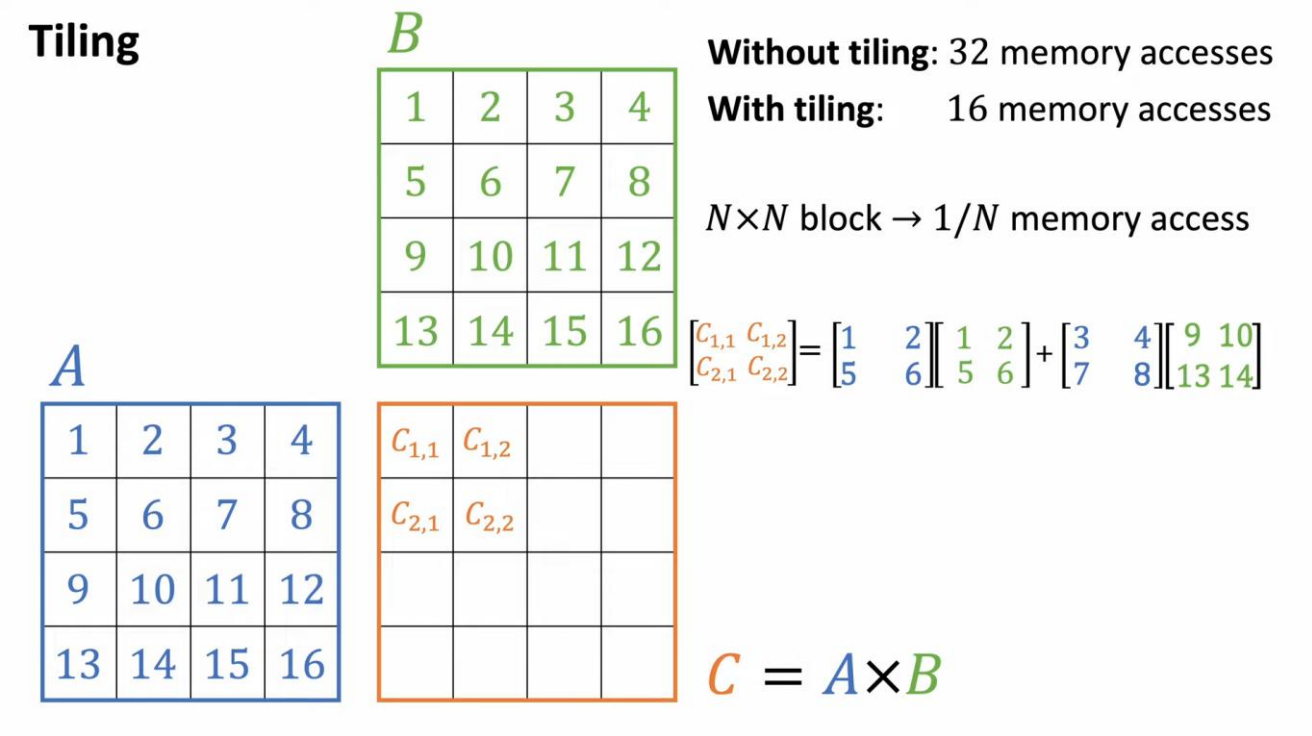
Flash Attention

- Tiling이 없으면 A 메모리 엑세스 + B 메모리 엑세스= 32번 발생



Flash Attention

- Tiling을 통해 메모리 I/O를 줄일 수 있음



Flash Attention

- Softmax는 m_N과 d_N을 구하는 for loop가 존재

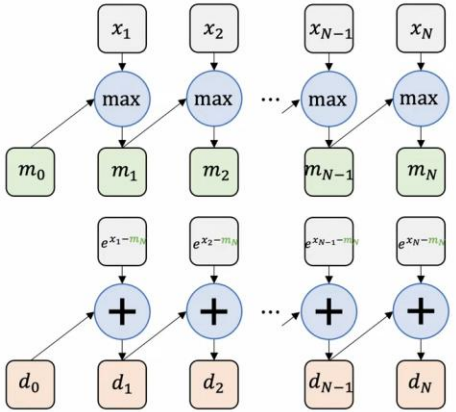
$$A = \text{Softmax}(S) \qquad S = \{x_1, x_2, \cdots, x_N\}$$

$$m_N = \max(\{x_1, x_2, \cdots, x_N\})$$

$$m_0 = -\infty$$
$$\textbf{for } 1 \le i \le N \textbf{ do}$$
$$m_i = \max(m_{i-1}, x_i)$$

$$d_N = \sum_{i=1}^{N} e^{x_i - m_N}$$

$$d_0 = 0$$
$$\textbf{for } 1 \le i \le N \textbf{ do}$$
$$d_i = d_{i-1} + e^{x_i - m_N}$$



$$A = \{a_1, a_2, \cdots, a_N\}$$
$$a_i = e^{x_i - m_N} / d_N$$

$$\textbf{for } 1 \le i \le N \textbf{ do}$$
$$a_i = e^{x_i - m_N} / d_N$$

Flash Attention

- m_N, d_N을 통합

$$A = \text{Softmax}(S) \quad S = \{x_1, x_2, \cdots, x_N\}$$

$m_0 = -\infty$

for $1 \le i \le N$ **do**

$m_i = \max(m_{i-1}, x_i)$

$d_0 = 0$

for $1 \le i \le N$ **do**

$d_i = d_{i-1} + e^{x_i - m_N}$

for $1 \le i \le N$ **do**

$a_i = \dfrac{e^{x_i - m_N}}{d_N}$

$$d_i = \sum_{j=1}^{i} e^{x_j - m_N} \qquad d'_i = \sum_{j=1}^{i} e^{x_j - m_i} \qquad d'_N = d_N$$

$$d'_i = \left(\sum_{j=1}^{i-1} e^{x_j - m_i}\right) + e^{x_i - m_i}$$

결과는 1

$$d'_i = \left(\sum_{j=1}^{i-1} e^{x_j - m_{i-1}}\right) \cdot e^{m_{i-1}} \cdot e^{-m_i} + e^{x_i - m_i}$$

$$d'_i = \left(\sum_{j=1}^{i-1} e^{x_j - m_{i-1}}\right) \cdot e^{m_{i-1} - m_i} + e^{x_i - m_i}$$

d'_{i-1}

Flash Attention

- m_N, d_N을 통합: Online Softmax

$$A = \text{Softmax}(S) \qquad S = \{x_1, x_2, \cdots, x_N\}$$

$m_0 = -\infty$
for $1 \le i \le N$ **do**
  $m_i = \max(m_{i-1}, x_i)$

$d_0 = 0$
for $1 \le i \le N$ **do**
  $d_i = d_{i-1} + e^{x_i - m_N}$

for $1 \le i \le N$ **do**
  $a_i = e^{x_i - m_N} / d_N$

$$d'_i = d'_{i-1}\, e^{m_{i-1} - m_i} + e^{x_i - m_i} \qquad d'_N = d_N$$

$m_0 = -\infty$
$d_0 = 0$
for $1 \le i \le N$ **do**
  $m_i = \max(m_{i-1}, x_i)$
  $d'_i = d'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}$

for $1 \le i \le N$ **do**
  $a_i = e^{x_i - m_N} / d'_N$

Flash Attention

- O_i통합

$$S = QK^\top \qquad A = \mathrm{Softmax}(S) \qquad O = AV \qquad S = \{x_1, x_2, \cdots, x_N\}$$

$$x_i = qk_i^\top$$

$$d'_i = d'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}$$

$$m_0 = -\infty$$

$$d_0 = 0$$

for $1 \le i \le N$ **do**

$$x_i = qk_i^\top$$

$$m_i = \max(m_{i-1}, x_i)$$

$$d'_i = d'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}$$

$$o_0 = 0$$

for $1 \le i \le N$ **do**

$$a_i = \frac{e^{x_i - m_N}}{d'_N}$$

$$o_i = o_{i-1} + a_i v_i$$

return o_N

$$o_i = \sum_{j=1}^{i} \frac{e^{x_j - m_N}}{d'_N} v_j \qquad o_N = o'_N$$

$$o'_i = \sum_{j=1}^{i} \frac{e^{x_j - m_i}}{d'_i} v_j$$

$$o'_i = \sum_{j=1}^{i-1} \frac{e^{x_j - m_i}}{d'_i} v_j + \frac{e^{x_i - m_i}}{d'_i} v_i$$

$$o'_i = \sum_{j=1}^{i-1} \frac{e^{x_j - m_i}}{d'_i} \frac{d'_{i-1}}{d'_{i-1}} \frac{e^{-m_{i-1}}}{e^{-m_{i-1}}} v_j + \frac{e^{x_i - m_i}}{d'_i} v_i$$

$$o'_i = \left(\sum_{j=1}^{i-1} \frac{e^{x_j - m_{i-1}}}{d'_{i-1}} v_j\right) \frac{d'_{i-1}}{d'_i} e^{m_{i-1} - m_i} + \frac{e^{x_i - m_i}}{d'_i} v_i$$

o'_{i-1}

Flash Attention

- 하나의 Loop로 통합: Flash Attention

$$S = QK^\top \qquad A = \mathrm{Softmax}(S) \qquad O = AV \qquad S = \{x_1, x_2, \cdots, x_N\}$$

$$x_i = qk_i^\top$$

for $1 \le i \le N$ **do**

$$x_i = qk_i^\top$$

$$m_i = \max(m_{i-1}, x_i)$$

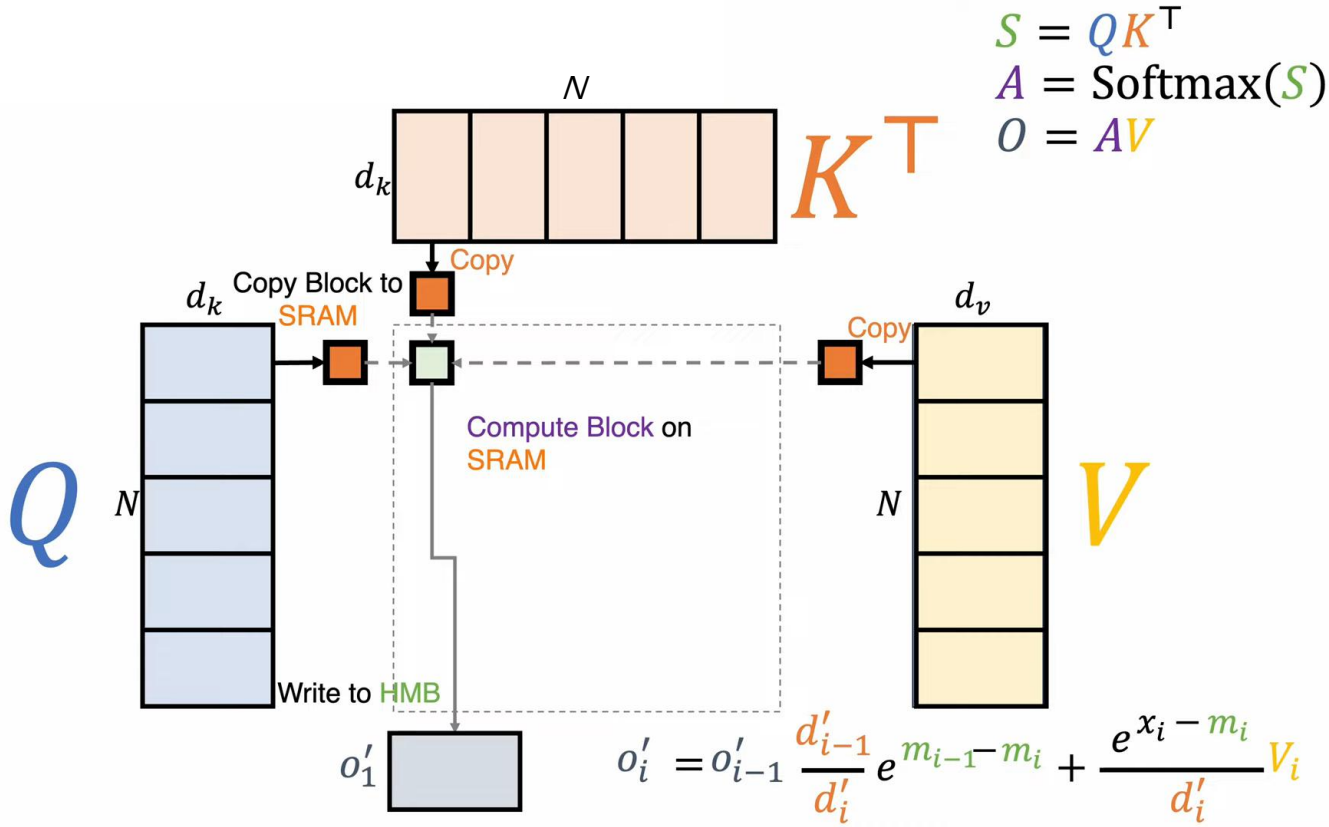
$$d'_i = d'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}$$

$$o'_i = o'_{i-1} \frac{d'_{i-1}}{d'_i} e^{m_{i-1} - m_i} + \frac{e^{x_i - m_i}}{d'_i} v_i$$

return o'_N

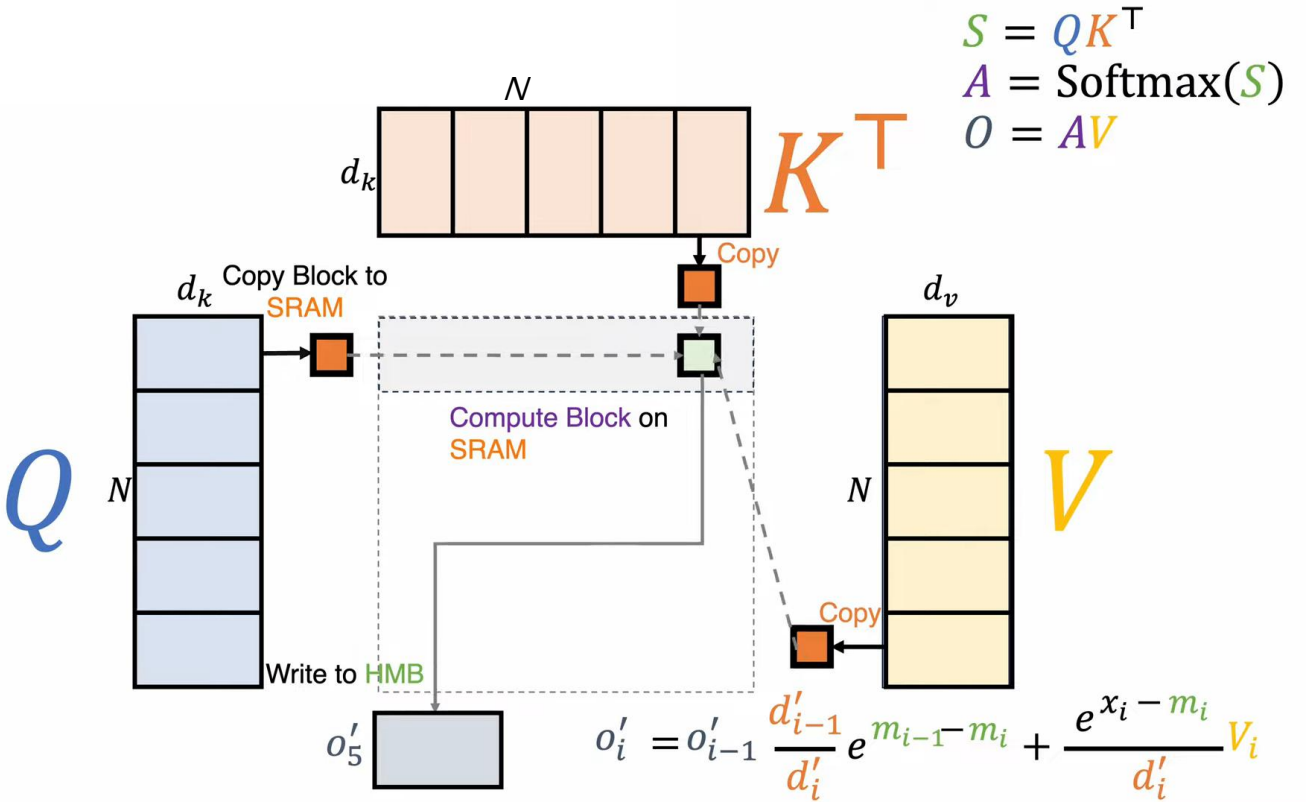
Flash Attention

- 대용량 메모리를 주고 받는 일이 없음



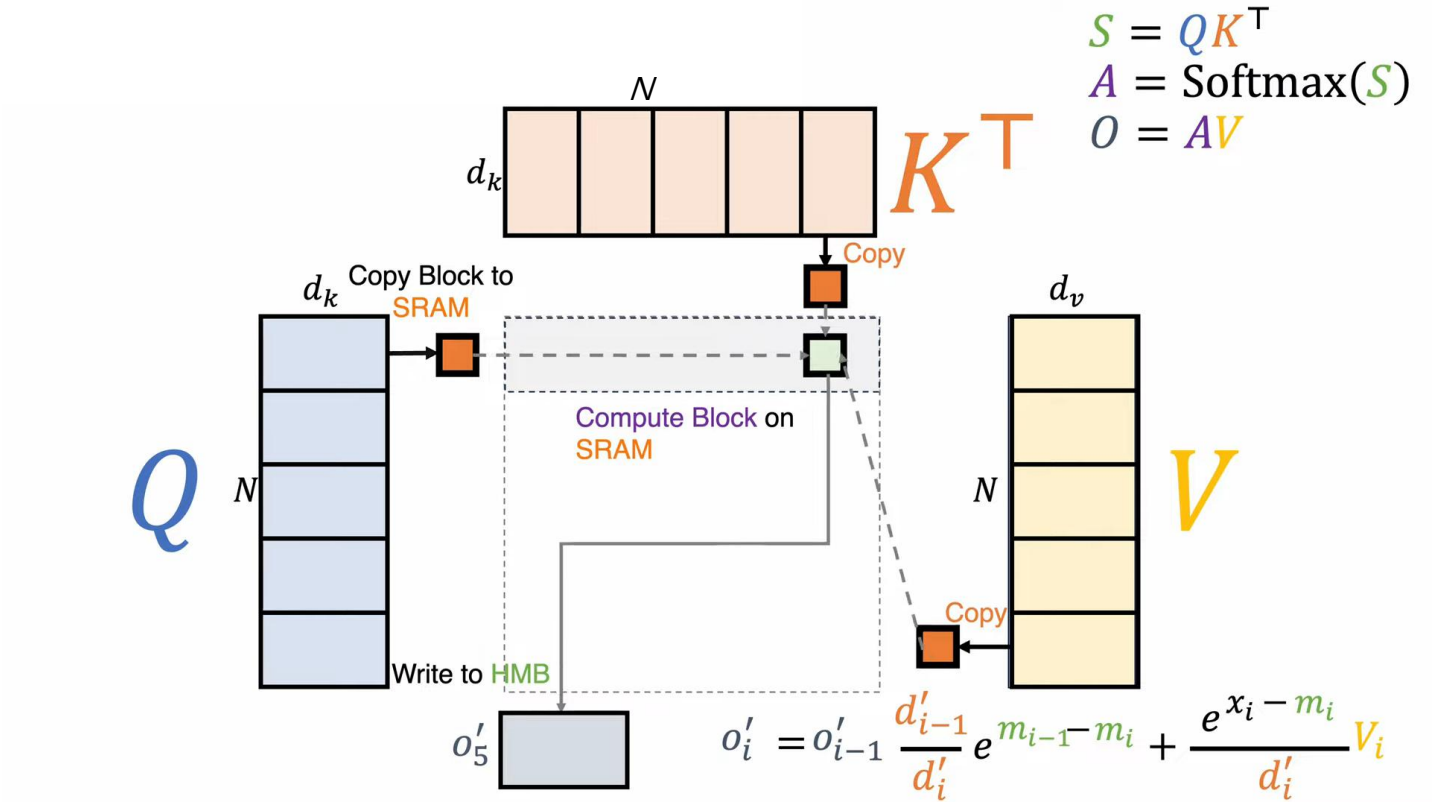
Flash Attention

- 계산에 필요한 메모리만 SRAM에 Copy하고, 계산 결과만 HBM에 씀



Flash Attention

- 계산에 필요한 메모리만 SRAM에 Copy하고, 계산 결과만 HBM에 씀



Appendix

박영진(솔.찬아빠)

- 두번의 C-Lab 도전
- 2년의 ASR
- 4년의 멀티 Agent 선행 개발과 Bixby 개발

설문조사

- LLM을 Fine-tuning해보신 분?
- AI를 유료로 사용하고 계시는 분?
- 평소 Python을 사용하시는 분?

현재

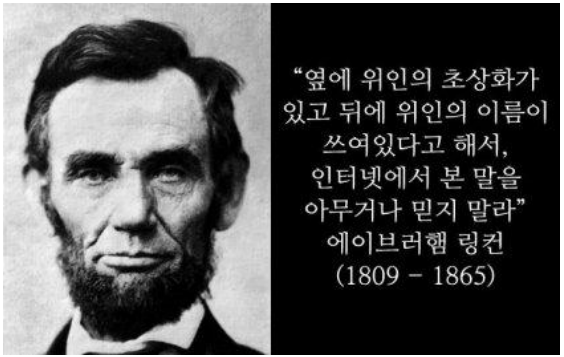
- 우리는 무엇을 해야 하는가?

주시하고

"지진경보는 이미 발령, 쓰나미가 언제 발생하는가?"

"유료 모델을 구독해서 쓰자"

항상 의심하고



함께 하자

"엄청 똑똑하지만 업무를 전혀 모르는 신입"

"인류와 함께 진화하는 존재"

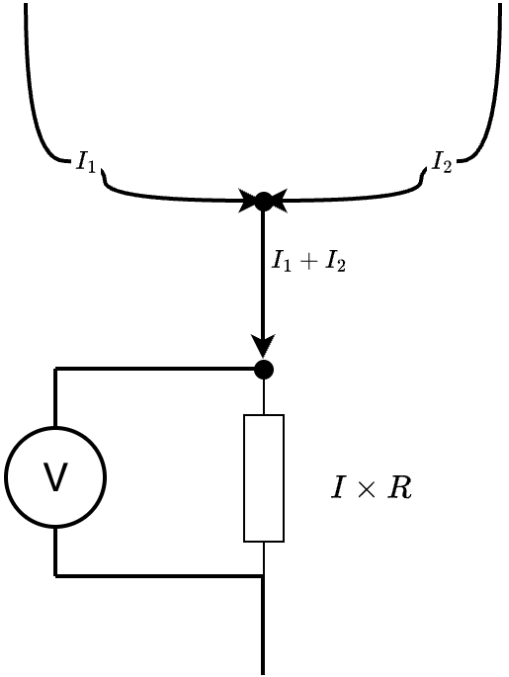
Alphago 이후

- AI를 어떻게 바라봐야 할까

이젠 옛날로 돌아갈 수 없다. 모든 것이 인공지능인 바둑계 충격적인 근황 | 알파고 10주년 | AI | 신진서 | 이세돌 | 다큐프라임 | #골라듄다큐

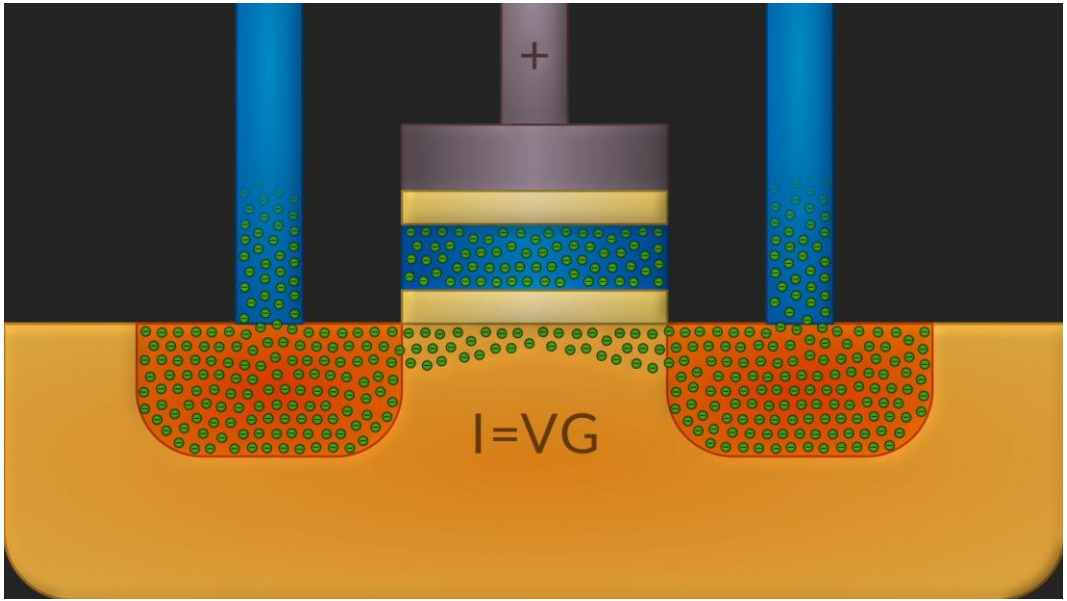
Analog Computer: 전류와 전압

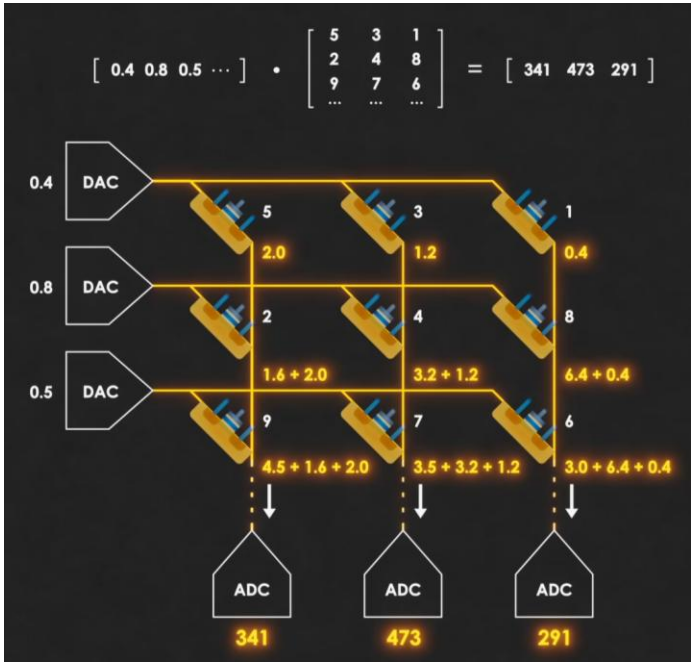
- 계산에 필요한 메모리만 SRAM에 Copy하고, 계산 결과만 HBM에 씀



Analog Computer: Mythic

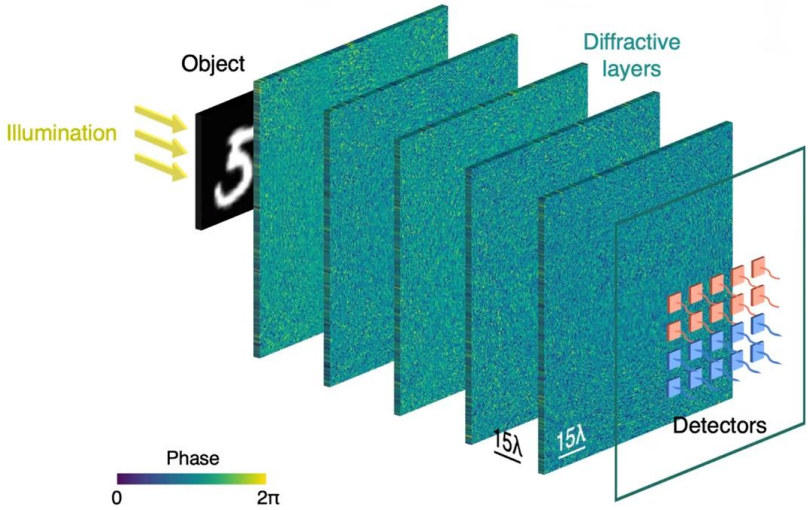
- 디지털 플래시 저장 셀처럼 전압에 따라 전류를 조정하는 칩 제작

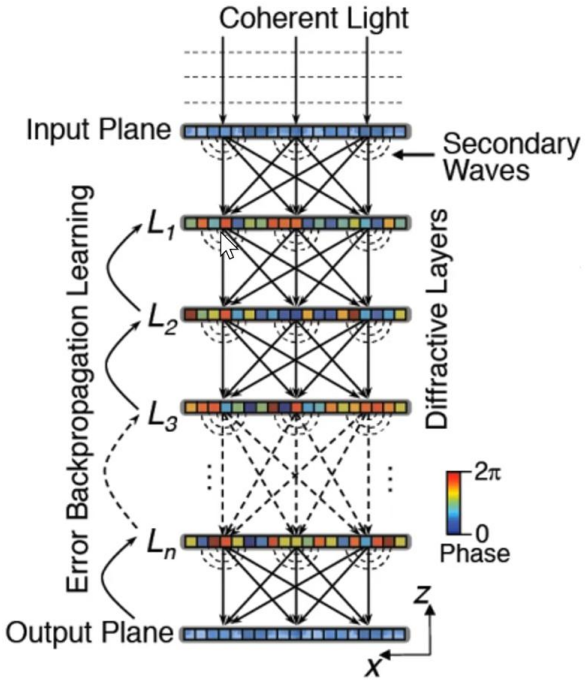




아날로그 컴퓨터: Diffractive Optical Neural Networks

- 필터에 의한 빛의 회절로 계산

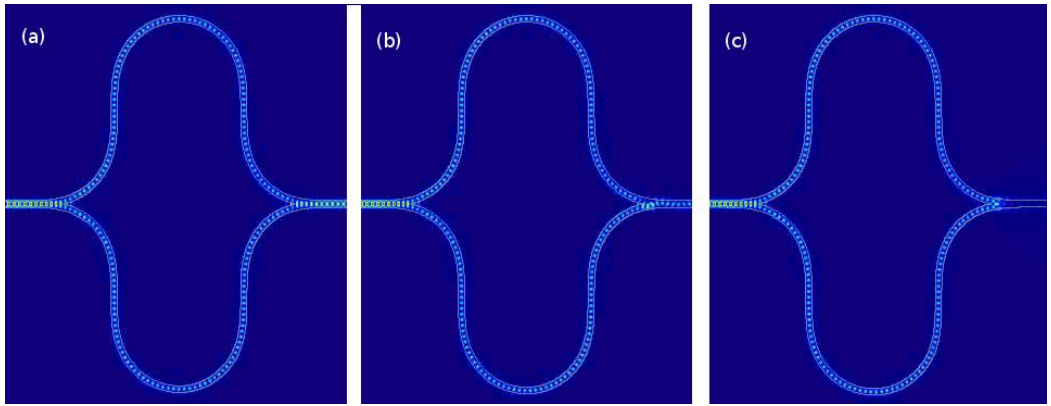


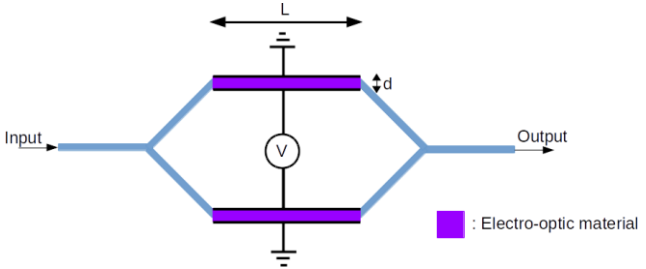


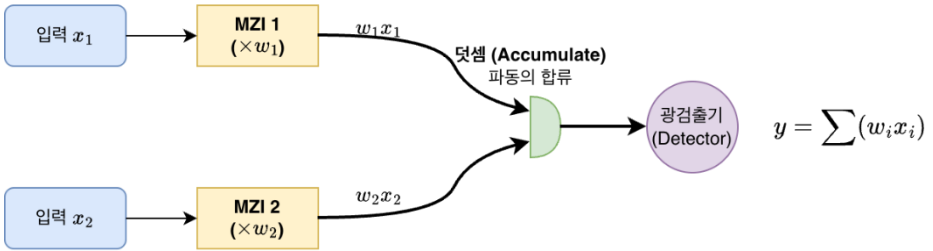
Transmission/Reflection

아날로그 컴퓨터: Lightmatter사의 Envise 칩

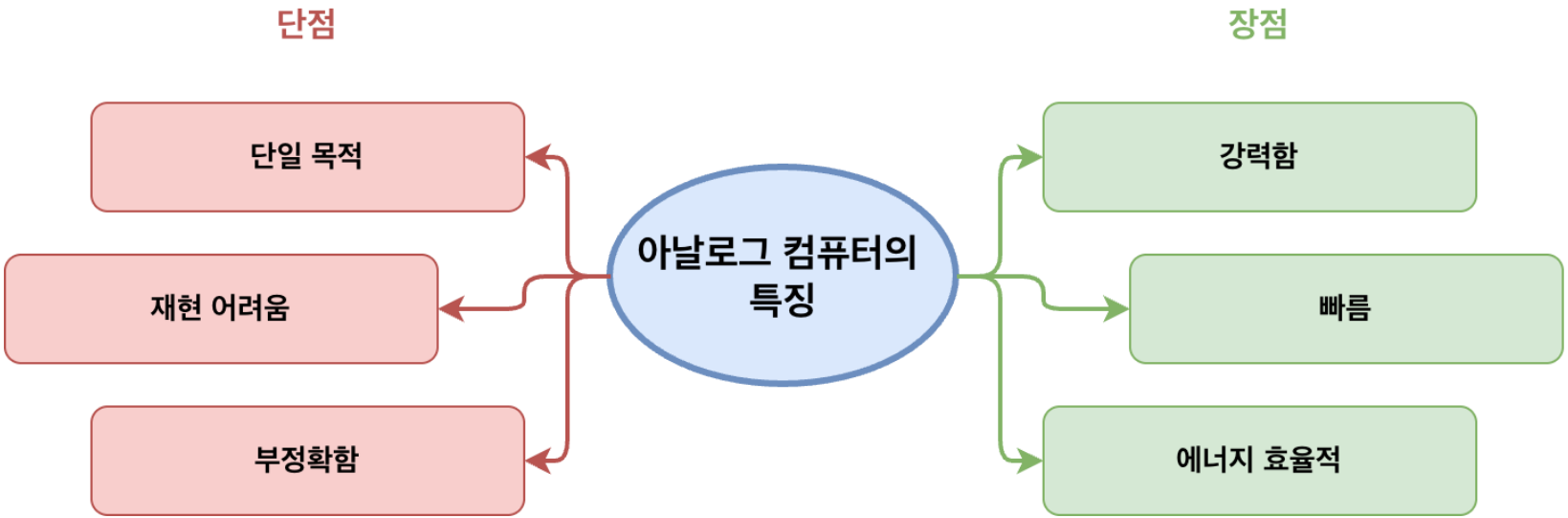
- 마하-젠더 간섭계: 빛을 두 방향으로 보내서 위상차를 발생시킨 후 간섭과 상쇄 이용
- 에너지를 거의 쓰지 않고 빛의 속도로 행렬 연산을 처리







아날로그 컴퓨터



Thank You!

SEU
Samsung Electronics University