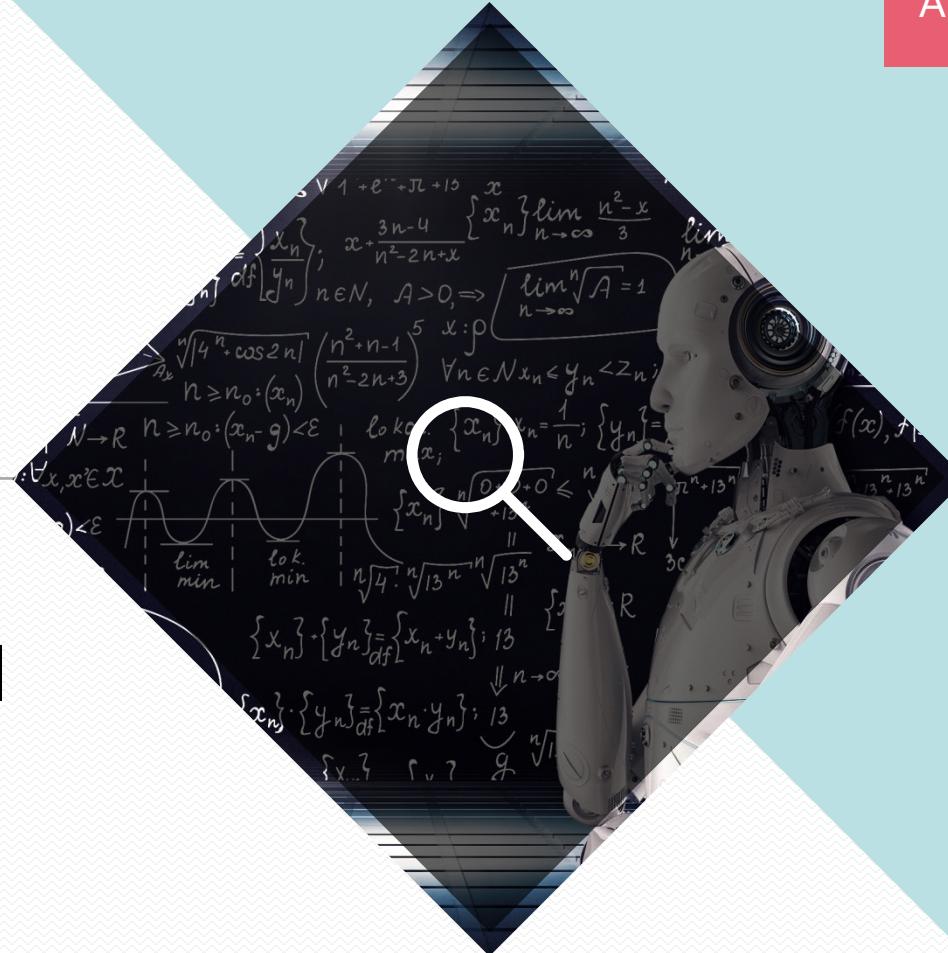


LLM Large Language Model

What is Large Language Model? How Does LLM Work?



목차

I. 언어모델이란?

- 언어 모델의 발전 과정
- Research 분야
- LLM Main models

II. 임베딩의 개념

- 단어 임베딩
- Word2Vec
- GloVe/FastText
- ELMo

III. 언어모델 기본개념

- 순환 신경망(RNN)
- LSTM
- Seq2seq
- Attention
- 문제점

IV. Transformer

- Attention Is All You Need
- Transformer
- BERT
- GPT

V. (실습)챗GPT API 활용

- 챗GPT API 활용 사례
- 챗GPT API 실습 – 예제코드 및 환경
- 챗GPT API 실습 - 나만의 음성 비서
- 챗GPT API 실습 - 카카오톡 챗봇

VI. (실습)랭체인

- 랭체인실습 - 우리 회사 챗봇

VII. 오픈소스 활용

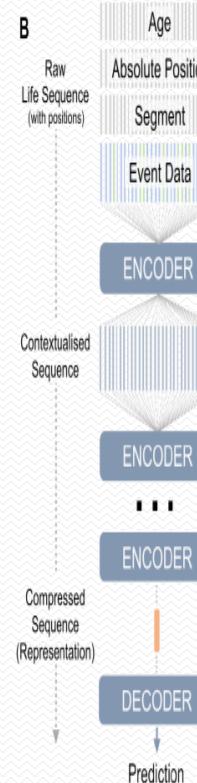
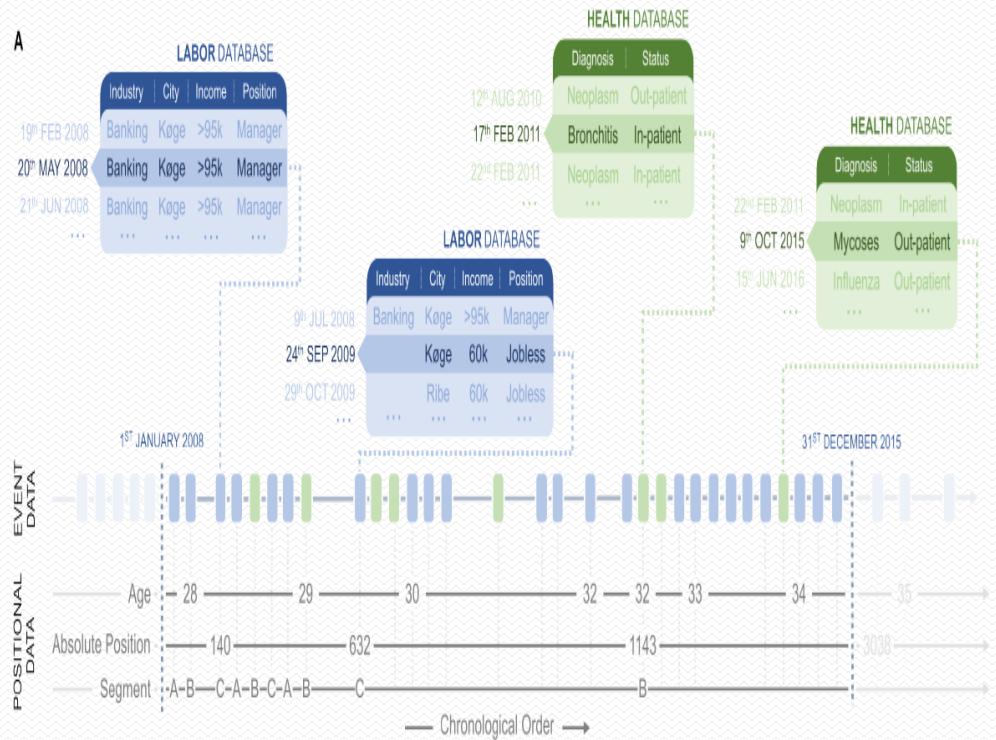
- AI 챗봇 '허깅챗'

ICE BREAKING

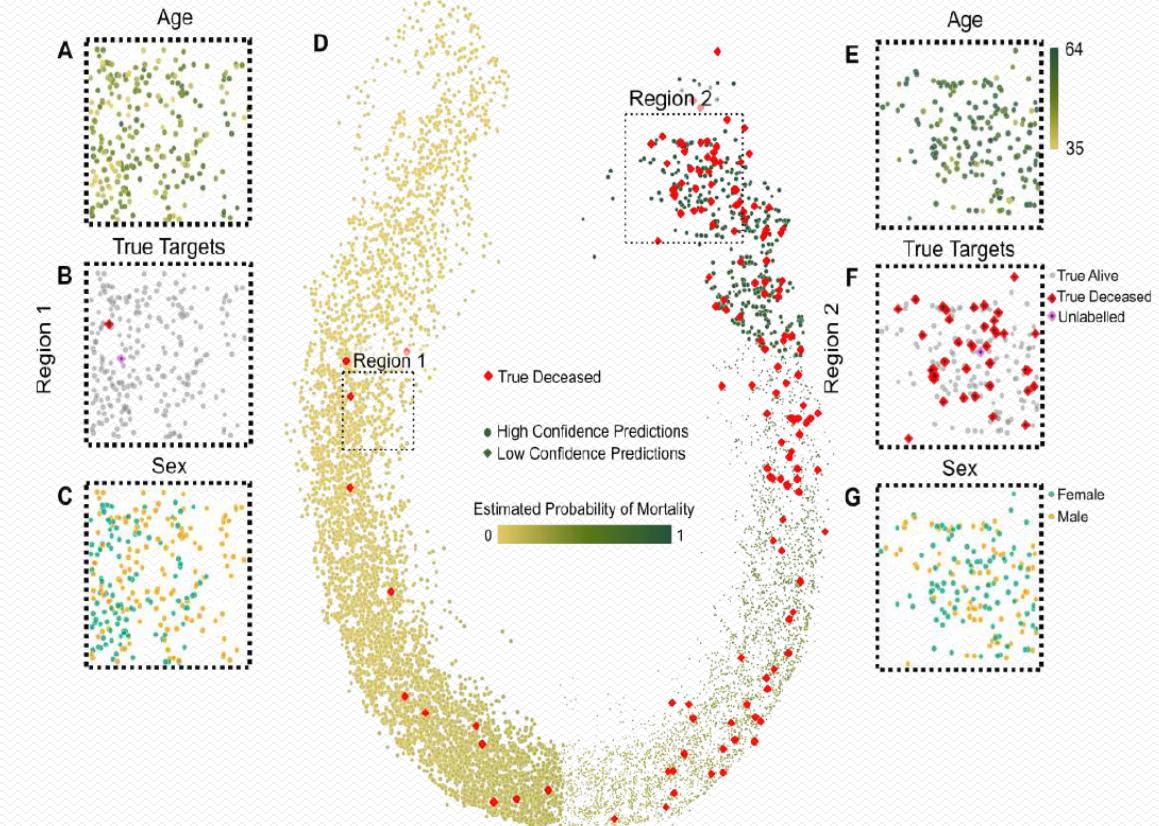


"LLM으로 사람의 미래까지 예측 가능"...덴마크공대, 미래 생성하는 모델 개발

A schematic individual-level data representation for the life2vec model.



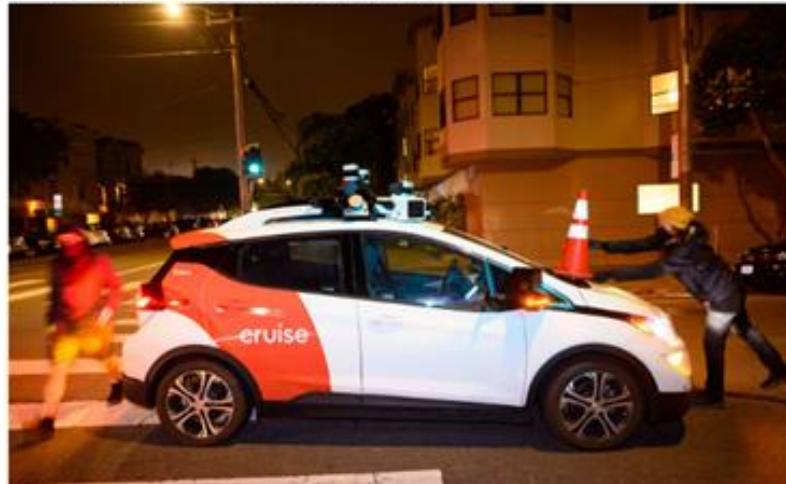
Person Embedding Space (*projected with DensMap*)
conditioned on the Mortality Task



고깔 쓰고 먹통된 자율주행 택시

A 봐 기자 | 2023.07.17 17:50

샌프란시스코 자율주행 택시 운행 저지 위한 캠페인 벌어져



7월 11일(현지시간) 미국 캘리포니아 샌프란시스코에서 자율주행차 운행을 반대하는 운동가들이 무인 택시 크루즈에 문을 썩워 작동을 멈추게 하고 있다.(사진=AFP)

[자율주행 반대 시위자가 리바운드 임의로 차에 올려놓는 경우]

구글 자율주행차, 주행 중 사고 "무사 고 기록 깨졌다"

| 입력 : 2016-03-02 12:09 | 수정 : 2016-03-02 12:15



이번에 사고를 일으킨 구글의 자율주행차와 동일한 차종

[뛰어든 개 피하기 위해 뒤따라오던 차량과 충돌]

뺑소니 당한 여성에 '쾅'...美샌프란 로보택시 또 사고 쳤다

교차로서 보행자 중상..."브레이크 걸었지만 제동거리 짧아"

세계 첫 24시간 무인영업 허용 후 교통흐름·용급차량에 방해

(서울=뉴스1) 김성식 기자 | 2023-10-04 14:01 송고



미국 샌프란시스코 시내에서 2일(현지시간) 제네랄모터스(GM)의 자율주행 무인택시로보택시(크루즈)가 신호를 어기고 교차로를 건너던 보행자 1명을 들이받아 중상을 입히는 사건이 발생했다. 사진은 출동한 구조대원들이 차량에 칼린 여성 을 구출하는 모습. 2023.10.02 © 로이터=뉴스1 © News1 김성식 기자

[학습되지 않은 "공사 중 사람의 수신호를 이해"]

미국 캘리포니아 DMV(교통국)에서 제공하는 **165건의 자율주행차 실사고 데이터인 DMV Collision Report 분석 결과**

자율주행차량/일반차량 충돌 형태가 **총 327,502개의 사고 연관규칙 도출**. 이는 결론적으로 기존 학습방법으로 모든 사고 예측이 어려움

따라서 **새로운 환경이나 예기치 못한 사고**에 대해서는 자율주행차량과 자율협력주행 관제센터에서 **스스로 예측하고 대응할 수 있는 상식 추론 모델의 개발이 필요**

구글_세계 최초_로봇 제어 인공지능 모델_사전학습 필요없이 시각과 언어 통해 스스로 작동 모델 공개

(#출처 : Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., ... & Zitkovich, B. (2023). RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. *arXiv preprint arXiv:2307.15818.*)



VLA모델 :
광범위한
시각적해석,
추론수행



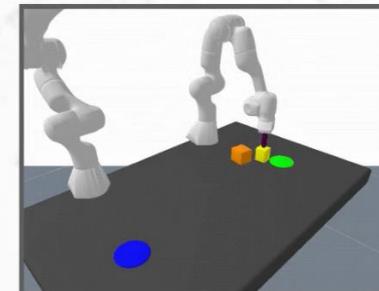
개별개체와
다른
개체와의
관계에
대한 Q&A
수행

Robot Mobile Manipulation



Task: give me the chips from the drawer
Next step: Close the drawer

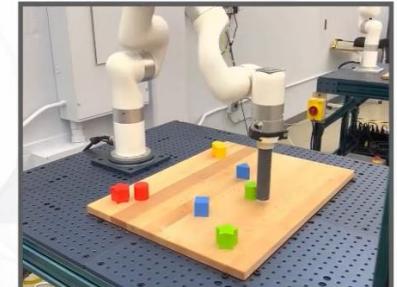
Task and Motion Panning



Q: How to put yellow block on blue plate?
A: Place the orange block on the side

PaLM-E
562B

Robot Tabletop Manipulation



Task: sort blocks by colors into corners
Next step: Push green blocks to the left

Visual Question Answering



Q: Tell me a joke about the image.
A: Rooster booster

연구개념도 – 전체 개념도

자율주행 소프트웨어에 사람의 언어를 학습시켜 스스로 인지·판단·제어할 수 있는 모델 개발

Vision 자율주행 소프트웨어

■ 자율주행 차량SW 모델 적용

- 1) Base Model
 - ① 해외
 - RT-2(Google)
 - NVIDIA
 - 모빌아이
 - ② 국내
 - 오토노마스에이투지
 - ETRI 인지/판단/제어 SW
- 2) 신뢰성 지원(Hallusion:착시)
 - ① 적대적 공격에 대한 강건한 모델 구축

Language 초기대AI언어모델

■ 상식 추론을 위한 LLM 모델

- 1) Base Model
 - ① 해외
 - ChatGPT(OPEN AI)
 - PaLM-E(Google)
 - LLaMA2(Meta)
 - ② 국내
 - ETRI 엑소브레인
- 2) 정확성 지원(Hallucination:환각)
 - ① LLM 환각 예방을 위한 지식그래프 모델 구축

Action 전이학습/미세조정

■ Prompt Engineering 기반 전이학습/미세조정

- 1) Prompt Engineering
 - ① LangChain Python Framework 활용 개발
 - ② 영상과 텍스트 통합 Cross Attention 모델 적용
- 2) 전이학습/미세조정 (Hallusion : 착시+환각)
 - ① 도로교통법, 자동차관리법
 - ② 교통안전공단
 - 자율주행사고 분석데이터
 - ③ 자율주행차 실사고 데이터
DMV collision report

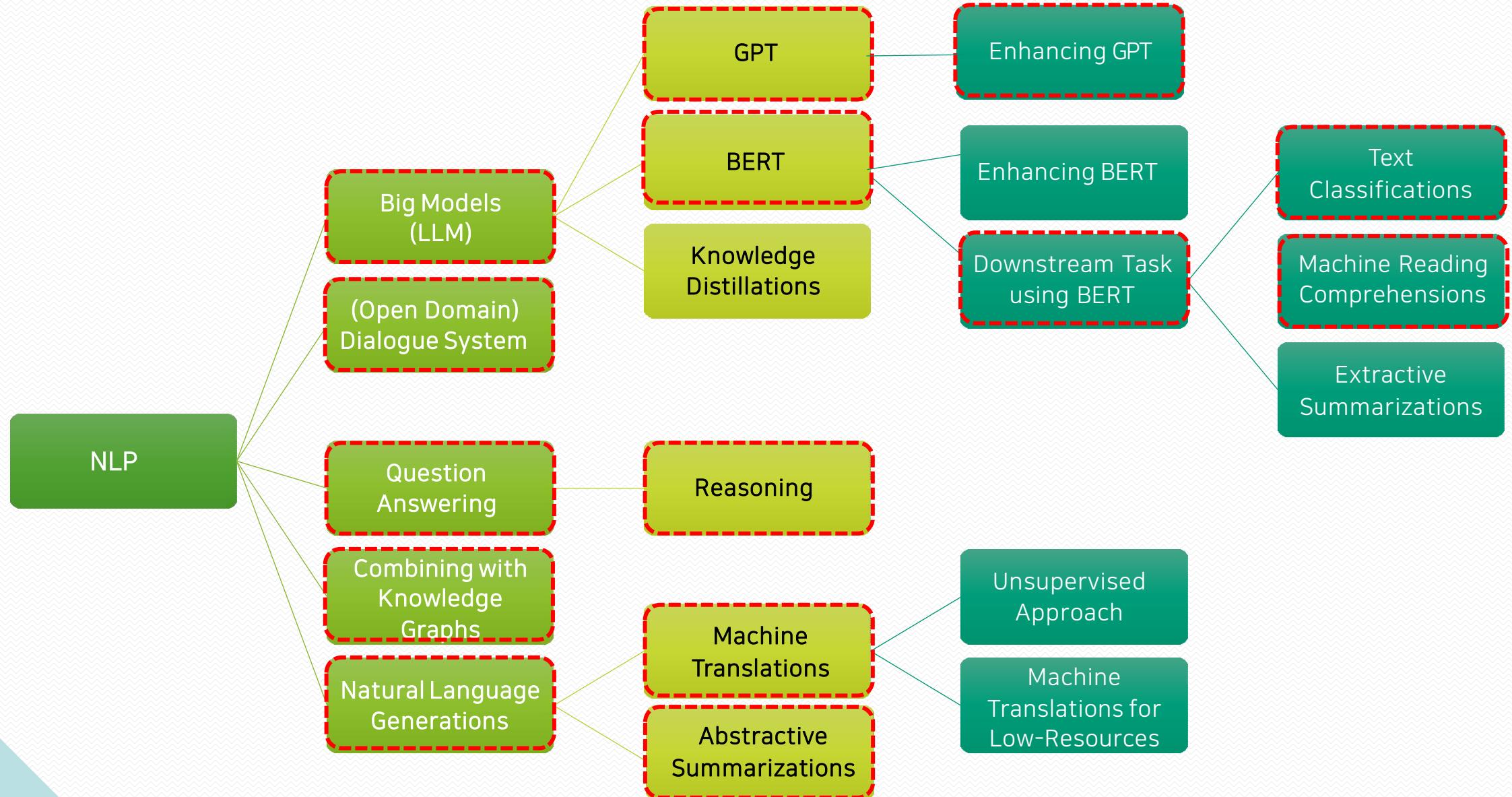
언어 모델의 발전 과정

긴 문장의 이해와 대용량 데이터에 대한 복잡도와 연산속도를 병렬로 처리하여 초기대 AI 모델의 기반이 됨.

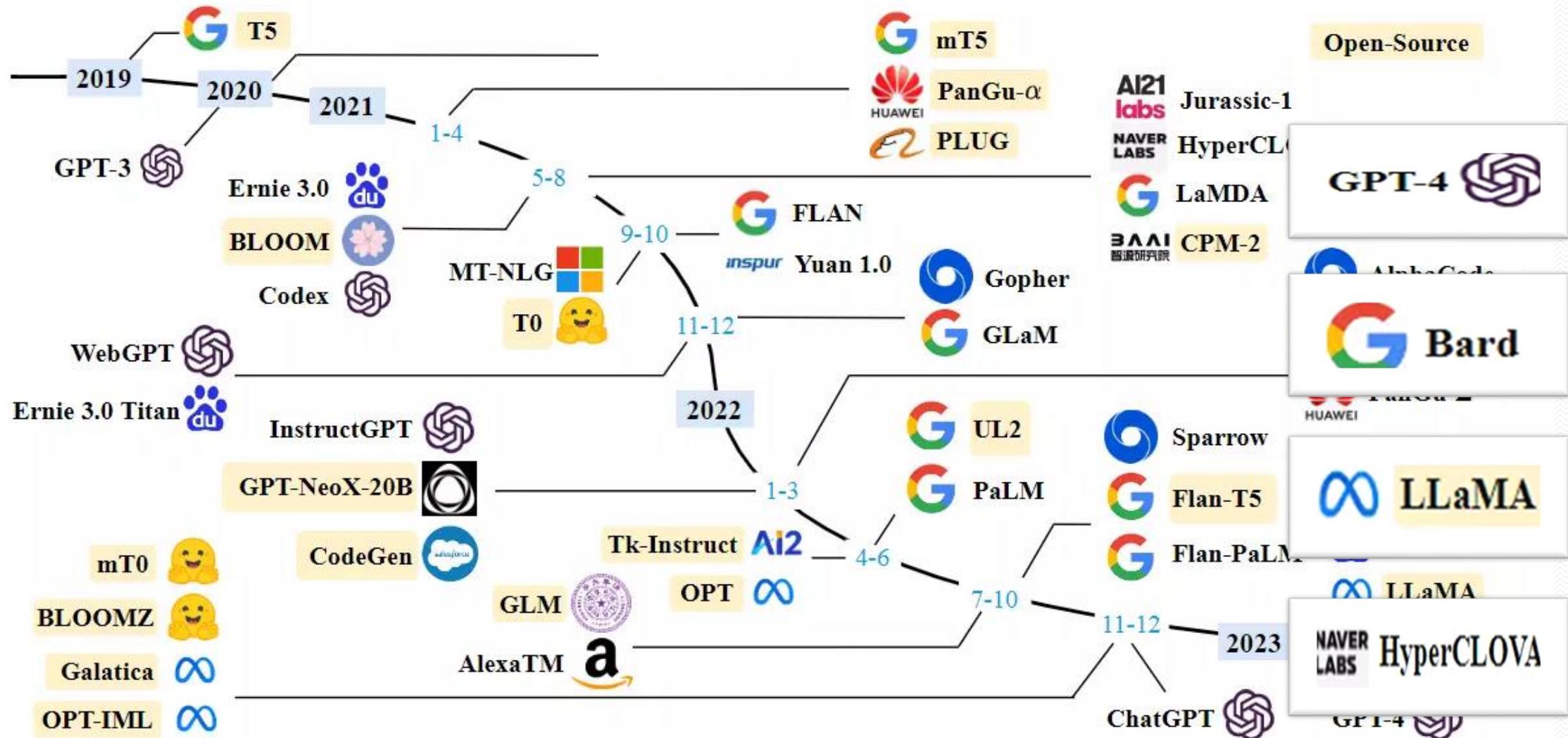
- 2021년 기준으로 최신 고성능 모델들은 Transformer 아키텍처를 기반으로 하고 있습니다.
 - GPT: Transformer의 디코더(Decoder) 아키텍처를 활용
 - BERT: Transformer의 인코더(Encoder) 아키텍처를 활용



Research 분야



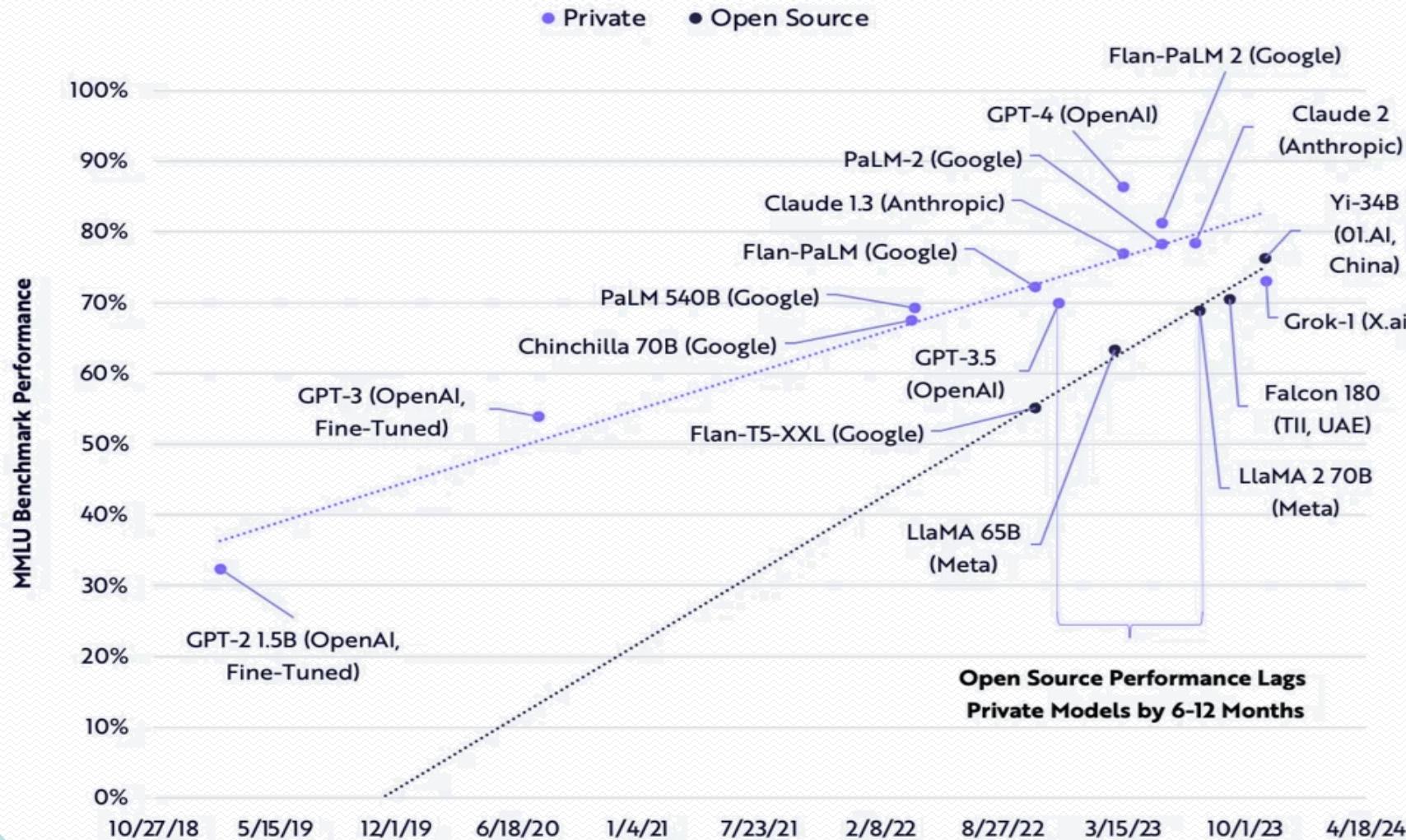
LLM Main models



A timeline of existing large language models (having a size larger than 10B) in recent years. We mark the open source LLMs in yellow color

오픈 소스와 프라이빗 모델의 비교

Open Source vs Private Models, 5-Shot MMLU Performance



GPT-4 : 86.5%
 Flan-PaLM- 구글의 의료 전문 모델 : 81.2%
 앤트로픽 '클로드 2' : 76.5%
 메타 '라마2 70B' : 68.9%
 UAE '팰컨 180B' : 70.4%
 일론 머스크 Xai '그록' : 73%
 중국 01.AI : 76.3%
 미스트랄 미쿠 70B(miku-1-70b), 'GPT-4' 성능에 근접

둘의 차이는 이제 불과 몇 개월 차이로 좁혀졌다

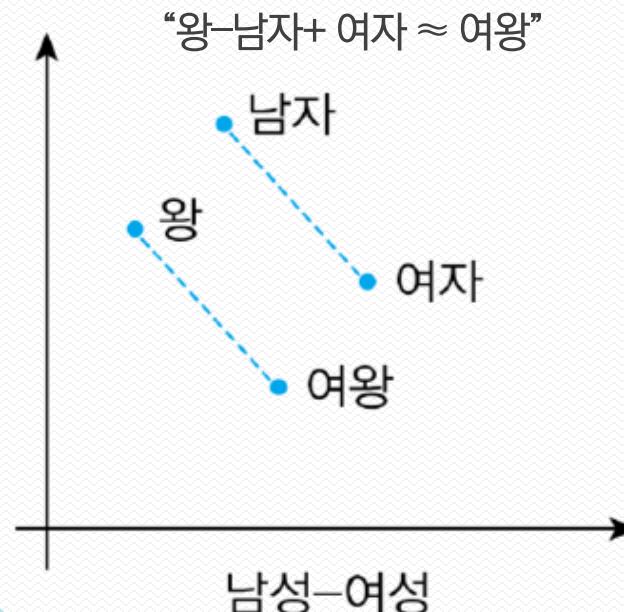
단어 임베딩

목표 : 컴퓨터가 단어의 의미를 이해하고 처리할 수 있도록 단어를 벡터로 변환.

해당 벡터는 다차원 공간에 위치하며, 단어 간의 의미적 관계를 반영함.

벡터 공간 상 단어 벡터들의 특징

- 학습된 단어 벡터들은 단어의 의미가 벡터 공간 상의 다양한 특성들(거리, 길이, 각도 등)로 표현됨
- 단어 벡터에는 문법에서 의존관계에 이르기까지 다양한 언어학적 지식들이 담겨져 있음.
- 단어 임베딩 모델에는 Word2Vec, GloVe, FastText 등이 있음



(예시) 5차원 벡터로 표현하면

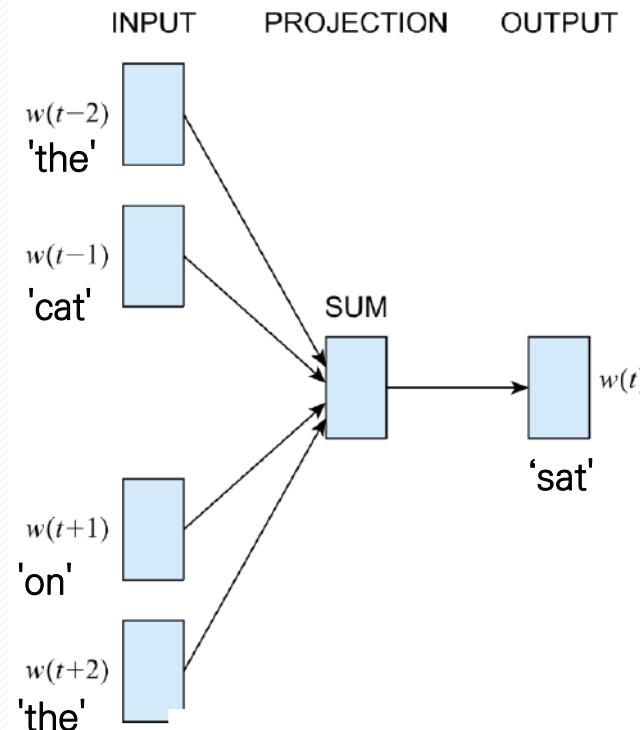
- 왕(king): [0.125, -0.765, 0.159, 0.256, -0.495]
- 여왕(queen): [0.145, -0.750, 0.140, 0.275, -0.482]
- 남자(man): [0.115, -0.855, 0.169, 0.333, -0.425]
- 여자(woman): [0.135, -0.840, 0.150, 0.352, -0.412]

Word2Vec

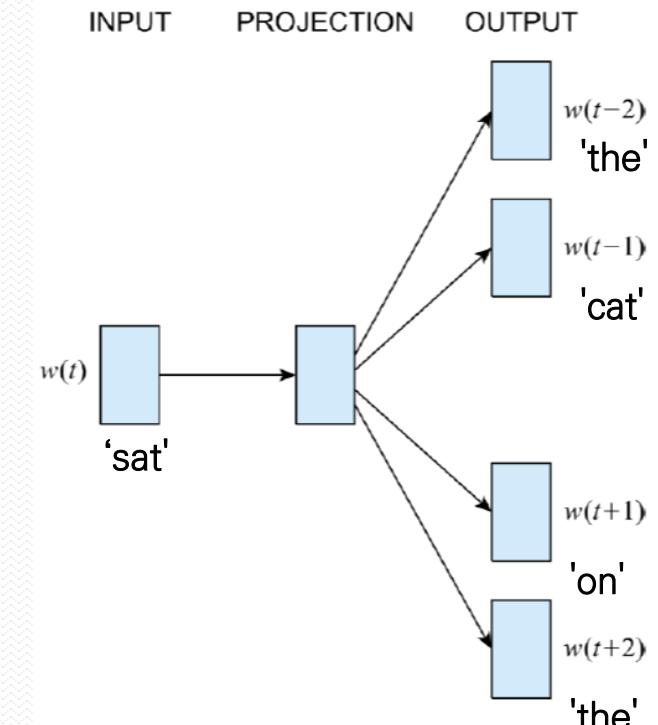
대표적인 두가지 알고리즘

- 두 방법 모두 확률적으로 더 먼 거리의 단어들을 선택함으로써 넓은 범위의 문맥을 단어 벡터 학습에 활용할 수 있도록 함

CBOW (Continuous Bag Of Words) 모델 :
특정 단어를 중심으로
이전 n 개의 단어와
이후 n 개의 단어가
주어졌을 때 , 중심
단어를 예측하는 것을
목표로 학습



Skip gram 모델 :
중심 단어가
주어졌을 때 이전 n
개의 단어와 이후 n
개의 단어를
예측하는 것을
목표로 학습



GloVe

GloVe : Global Vectors for Word Representation (Pennington et 의 2014 년 논문)

- 특정 단어가 문서 내에서 함께 등장하는 동시 발생 행렬 (co-occurrence matrix) 빈도를 기반으로 단어 벡터를 생성
- 단어 간 동시 발생 빈도의 비율이 벡터 공간상 차이로 나타나도록 단어 임베딩을 수행
- 특정 두 단어의 동시 발생 비율을 다른 단어 동시 발생 비율과 비교함으로써 단어 간 의미적 차이를 포착하는 학습

FastText

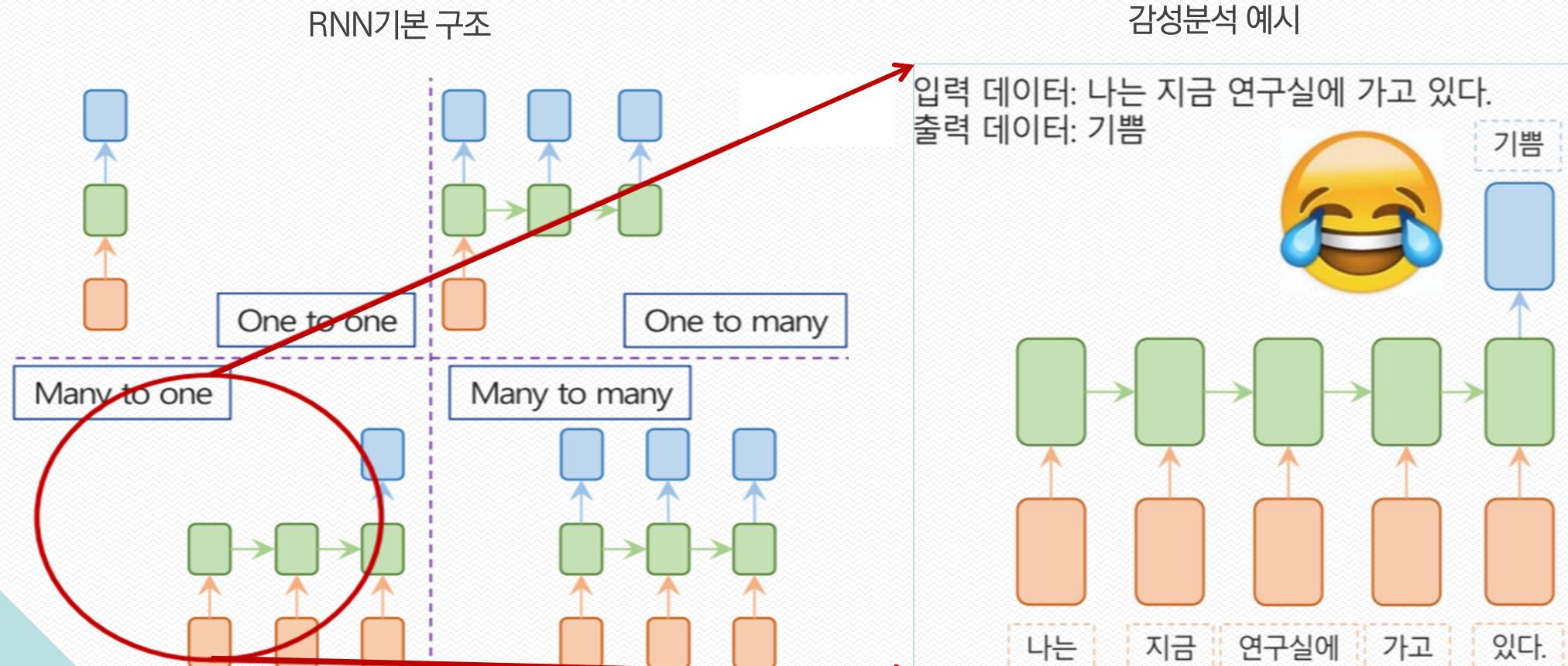
Facebook Research 에서 2016 년도에 공개

- 단어가 한국어의 형태소와 같이 더 작은 서브워드(subword) 단위로 나뉘어질 수 있다는 점에 착안
- 단어를 여러 n-gram 들의 집합으로 분리한 후 , 각 n-gram 에 대한 벡터를 학습
 - ✓ 2-gram 에서의 예 : “공부하였다” ⇒ ["<공”, ”공부”, ”부하”, ”하였”, ”였다”, ”다”]"
 - ✓ “공부하였다” 의 단어 벡터는 이를 구성하는 n gram 벡터들의 합으로 나타냄
- 모델이 단어의 형태학적인 특성을 학습, 복합 단어나 신조어, 심지어 철자가 틀린 단어까지도 그 의미를 더 잘 이해할 수 있게 함

순환 신경망(RNN)

시간 개념을 반영할 수 있는 인공 신경망

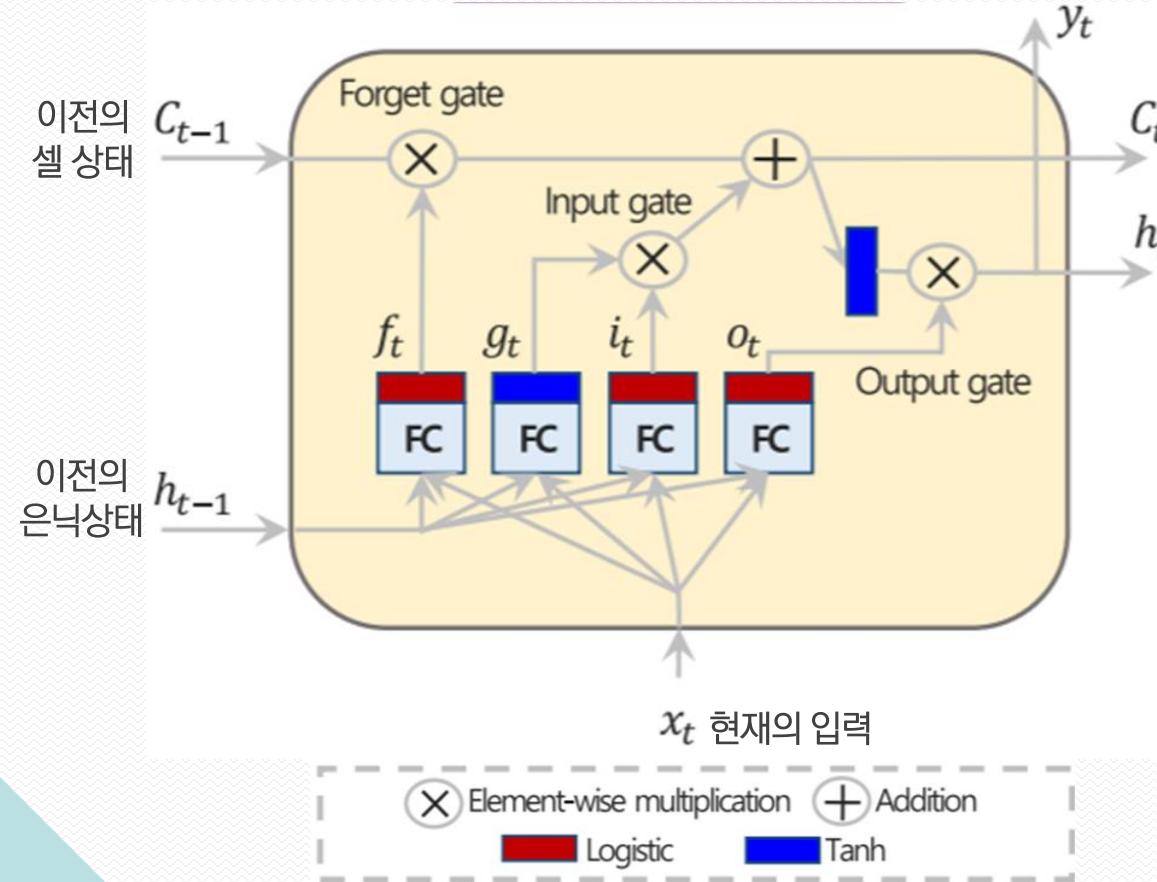
- RNN은 시퀀스 데이터(ex: 시계열데이터)를 처리하기 위해 설계된 신경망으로, 자연어 처리에서 특히 유용
- 시퀀스 데이터는 단어의 시퀀스인 문장, 시간에 따라 변화하는 센서데이터 등과 같이 순서에 의미가 있는 데이터



LSTM(Long Short Term Memory)

장기 의존성 문제를 완화한 RNN 개선 모델

- LSTM의 주요 특징은 '게이트(gates)'라는 구조를 도입하여 각 시점(time step)에서 어떤 정보를 기억하고, 삭제하며, 출력할지를 결정. 이 게이트들은 신경망이 필요한 정보를 장기간 기억하는 데 도움을 줌.



Forget gate : 과거 정보를 잊는 게이트

Input gate : 현재 정보를 기억하는 게이트

Output gate : 최종 결과 h_t 를 위한 게이트 & LSTM의 최종 결과

게이트 매커니즘을 통해 기울기 소실(Gradient vanishing)방지

수식 구현

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

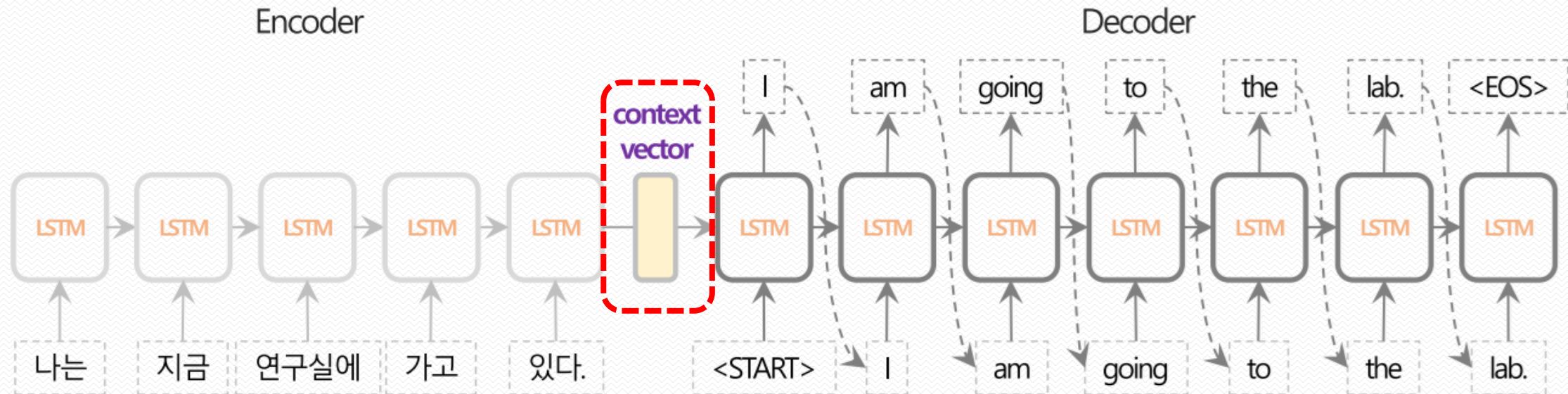
$$h_t = o_t * \tanh(c_t)$$

Seq2seq(Sequence-to-Sequence)

- Encoder–Decoder 구조로 RNN의 발전된 모델 아키텍쳐
- Encoder : 입력 데이터의 정보를 압축하는 역할
- Decoder : 인코더의 압축 정보와 출력 데이터를 입력 받아 문장 생성하는 역할
- LSTM, GRU 등의 RNN Cell을 길고 깊게 쌓아 더욱 복잡하고 방대한 시퀀스 데이터를 처리

입력 데이터: 나는 지금 연구실에 가고 있다.

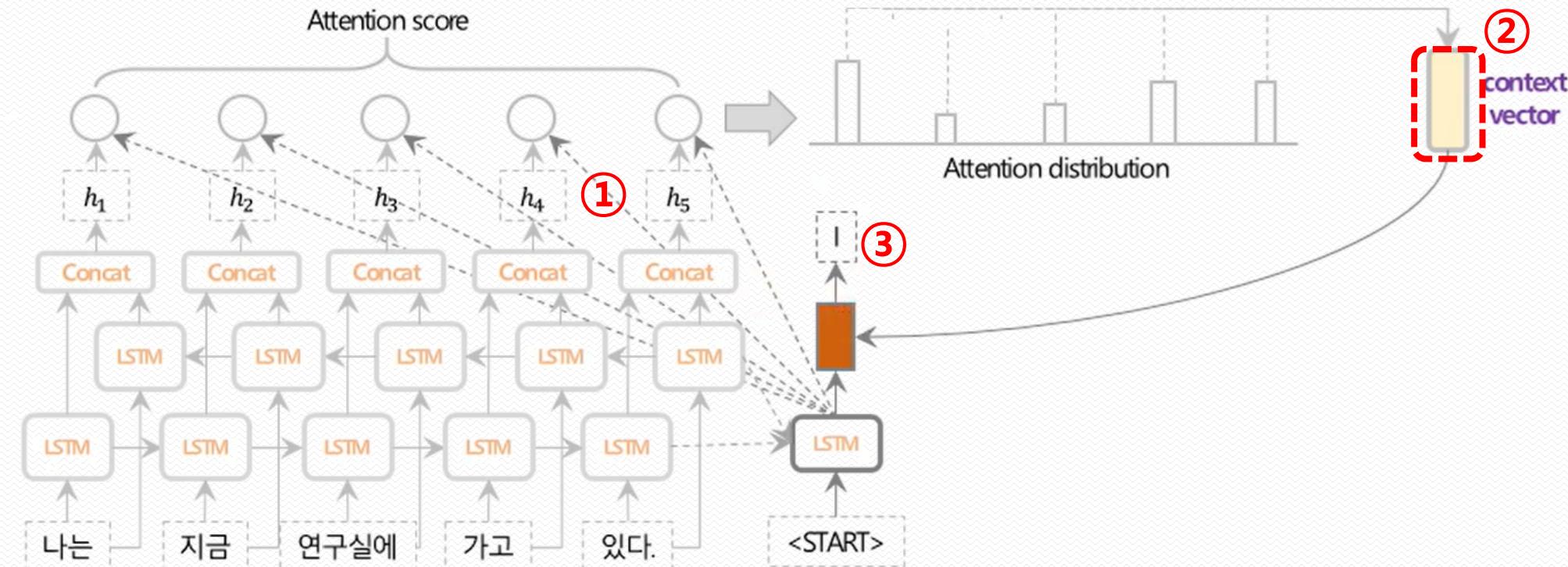
출력 데이터: I am going to the lab.



Attention

Attention Mechanism : Seq2Seq with Attention

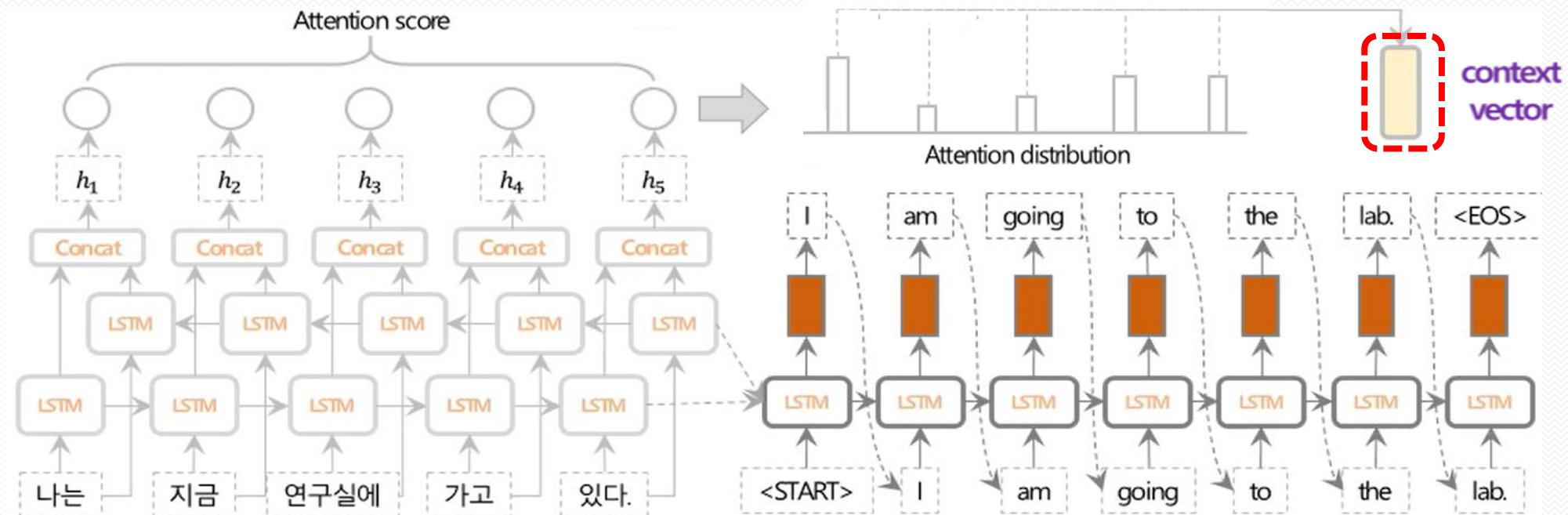
- Seq2Seq모델 Encoder에서 압축한 context vector는 전체 입력 시퀀스 데이터로만 효과적으로 표현할 수 있을까?
- **입력 시퀀스 데이터가 길어질 경우 여전히 문장 앞 부분에 대한 정보 손실 발생 (Long term dependency problem)**
- 입력 시퀀스 데이터의 길이가 긴 문장의 기억을 돋기 위한 방법
- Decoder에서 i번째 단어를 예측할 때 사용하는 이전 스텝의 Decoder정보와 인코더의 j번째 정보가 얼마나 유사한지 스코어 산출



Attention

Attention Mechanism : Seq2Seq with Attention

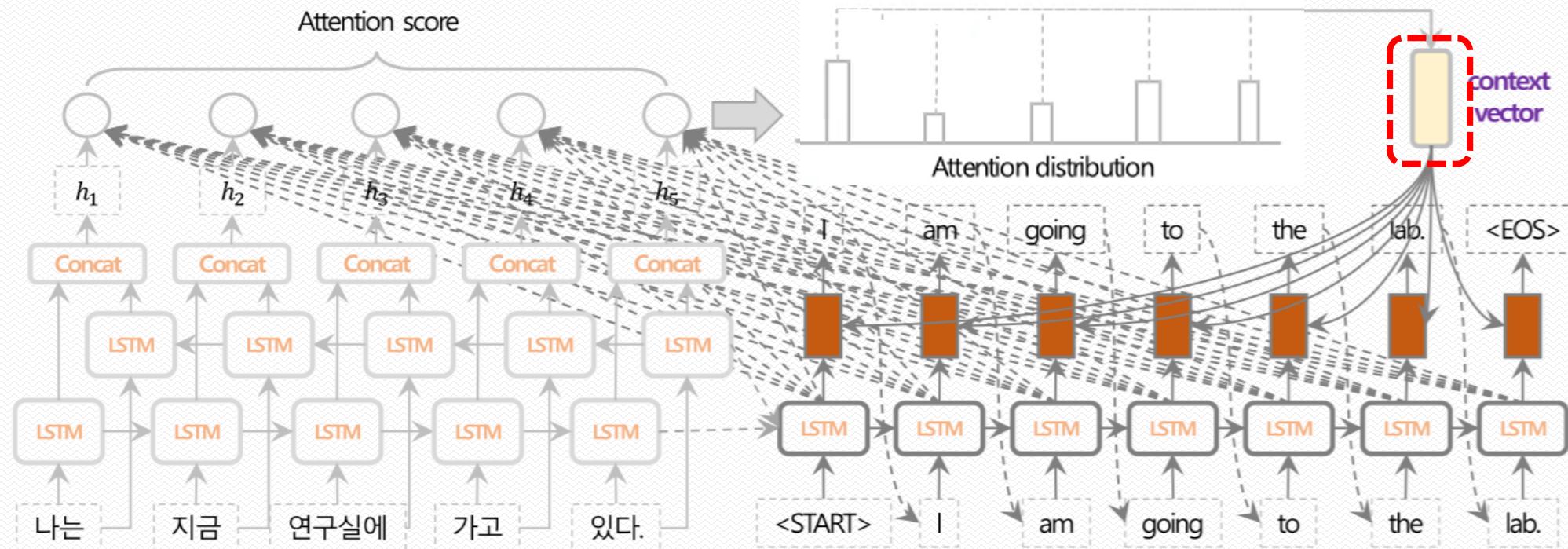
- Seq2Seq모델 Encoder에서 압축한 context vector는 전체 입력 시퀀스 데이터로만 효과적으로 표현할 수 있을까?
- **입력 시퀀스 데이터가 길어질 경우 문장 앞 부분에 대한 정보 손실 발생 (Long term dependency problem)**
- 입력 시퀀스 데이터의 길이가 긴 문장의 기억을 돋기 위한 방법
- Decoder에서 i번째 단어를 예측할 때 사용하는 이전 스텝의 Decoder정보와 인코더의 j번째 정보가 얼마나 유사한지 스코어 산출



Attention

Attention Mechanism : Seq2Seq with Attention

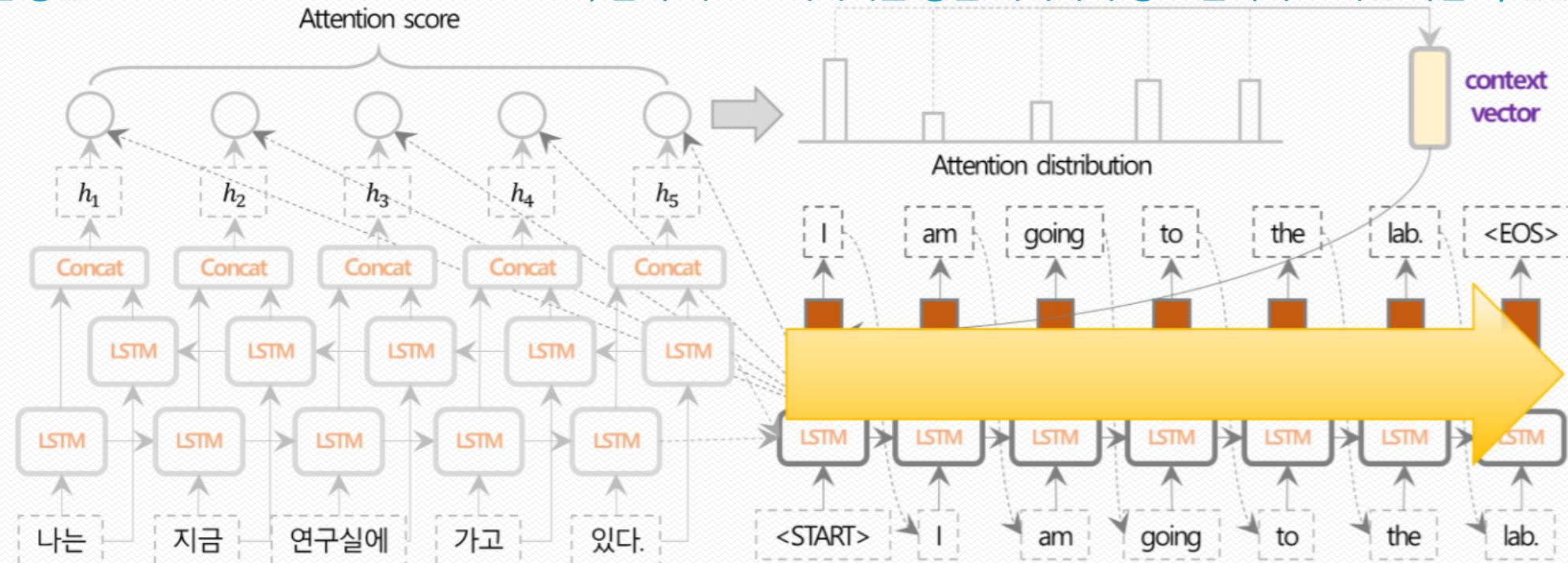
- Seq2Seq모델 Encoder에서 압축한 context vector는 전체 입력 시퀀스 데이터로만 효과적으로 표현할 수 있을까?
- **입력 시퀀스 데이터가 길어질 경우 문장 앞 부분에 대한 정보 손실 발생 (Long term dependency problem)**
- 입력 시퀀스 데이터의 길이가 긴 문장의 기억을 돋기 위한 방법
- Decoder에서 i번째 단어를 예측할 때 사용하는 이전 스텝의 Decoder정보와 인코더의 j번째 정보가 얼마나 유사한지 스코어 산출



문제점

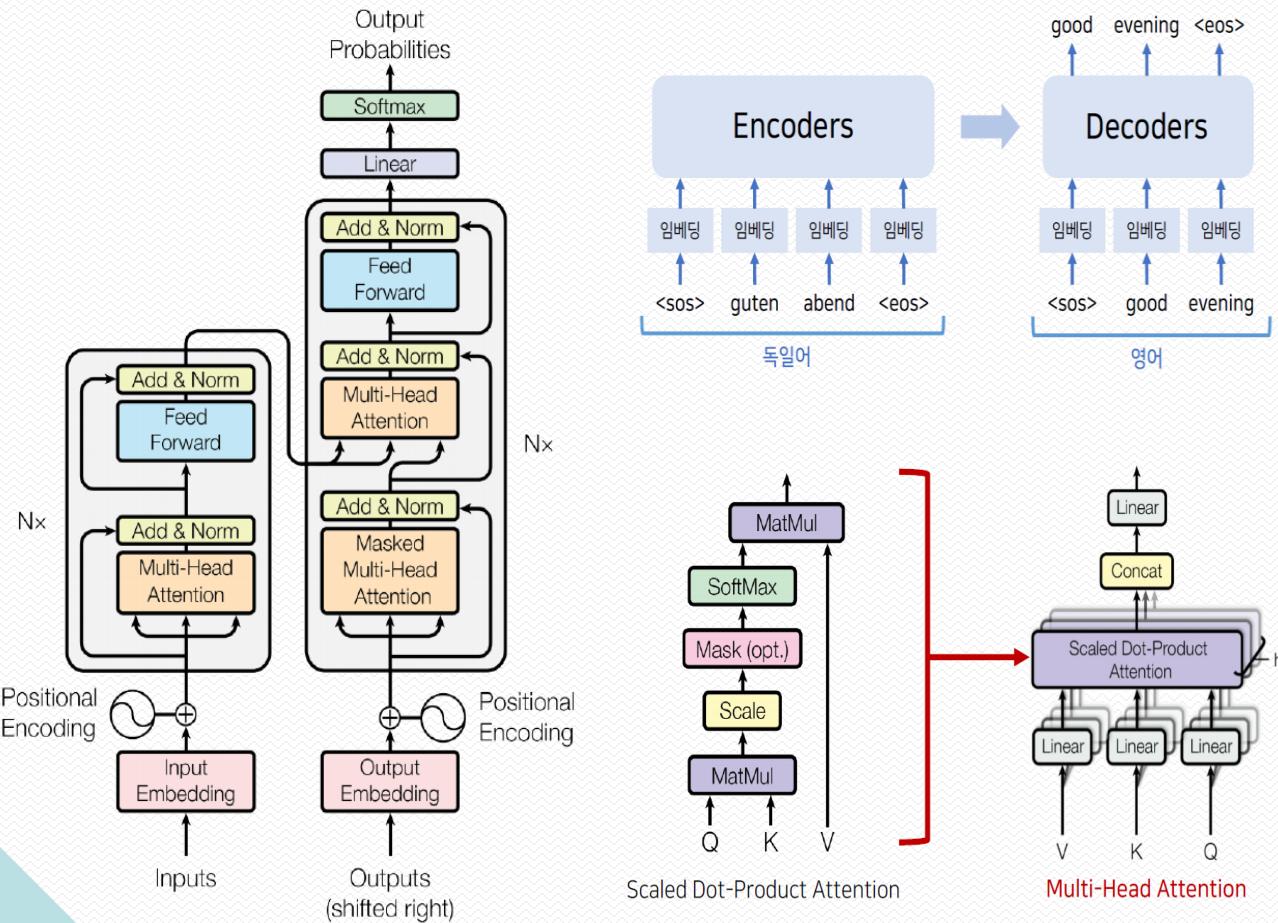
기존 Attention Mechanism의 문제점

- 입력 시퀀스 데이터를 순차적으로 처리하기 때문에 병렬 처리 불가능
- 계산 복잡도↑, 연산시간↑
- 해결 방안 : **Self-Attention mechanism**, 입력 시퀀스 데이터를 병렬 처리하여 정보 압축하고 계산 복잡도/연산시간을 줄이자 !



Attention Is All You Need

Attention Is All You Need (Google Brain, Google Research, University of Toronto)
 - 2017년 12월 Neural Information Processing Systems(Neural IPS)에서 발표
 - 2024년 2월 현재 106,941회 인용



Attention Is All You Need

Ashish Vaswani*	Noam Shazeer*	Niki Parmar*	Jakob Uszkoreit*
Google Brain	Google Brain	Google Research	Google Research
avaswani@google.com	noam@google.com	nikip@google.com	usz@google.com

Llion Jones*	Aidan N. Gomez* †	Lukasz Kaiser*
Google Research	University of Toronto	Google Brain
llion@google.com	aidan@cs.toronto.edu	lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

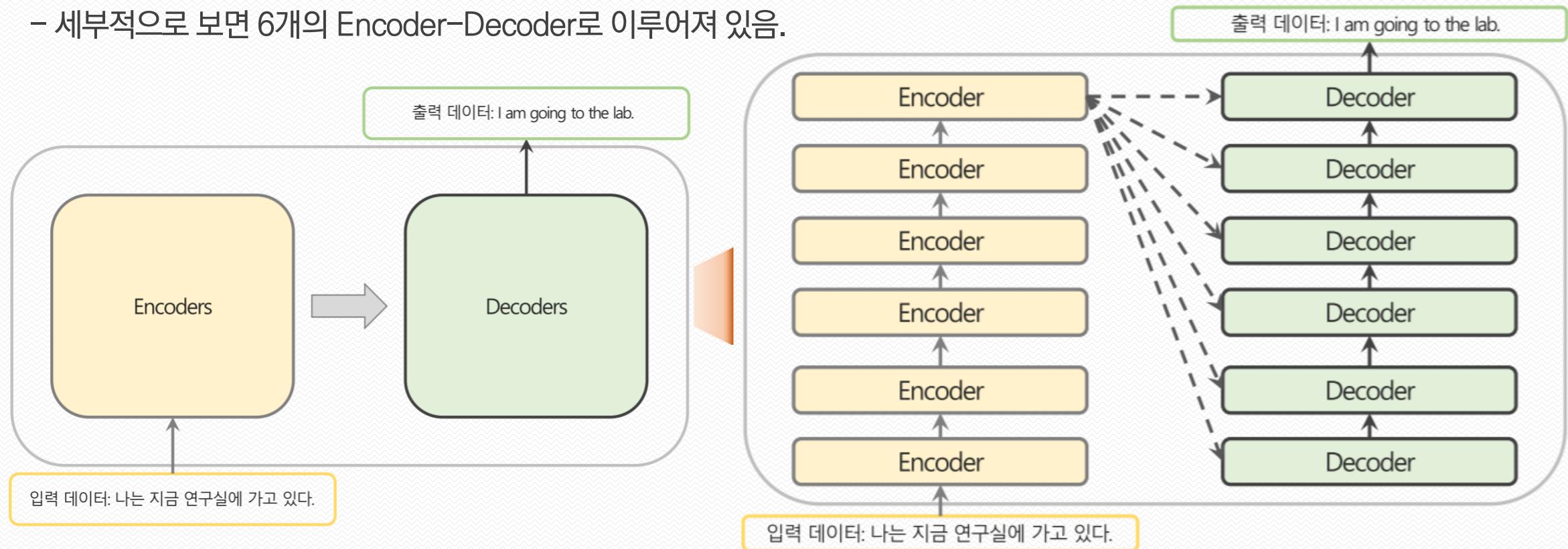
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the **Transformer**, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Transformer

Encoder–Decoder

- Transformer는 Encoder–Decoder 구조의 모델 아키텍쳐
- 세부적으로 보면 6개의 Encoder–Decoder로 이루어져 있음.

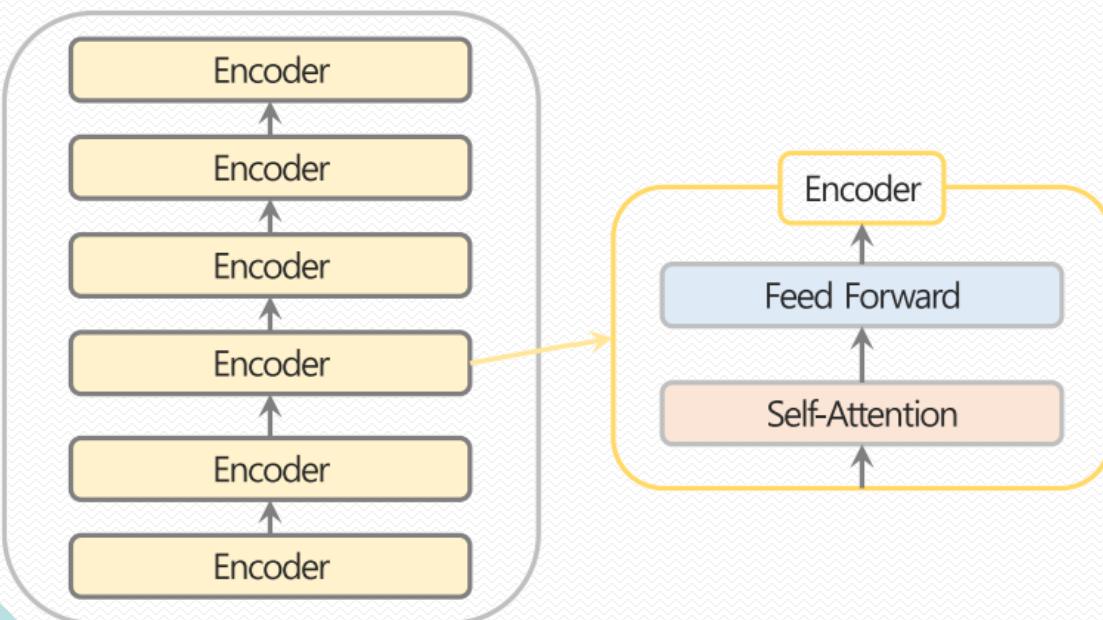


Transformer

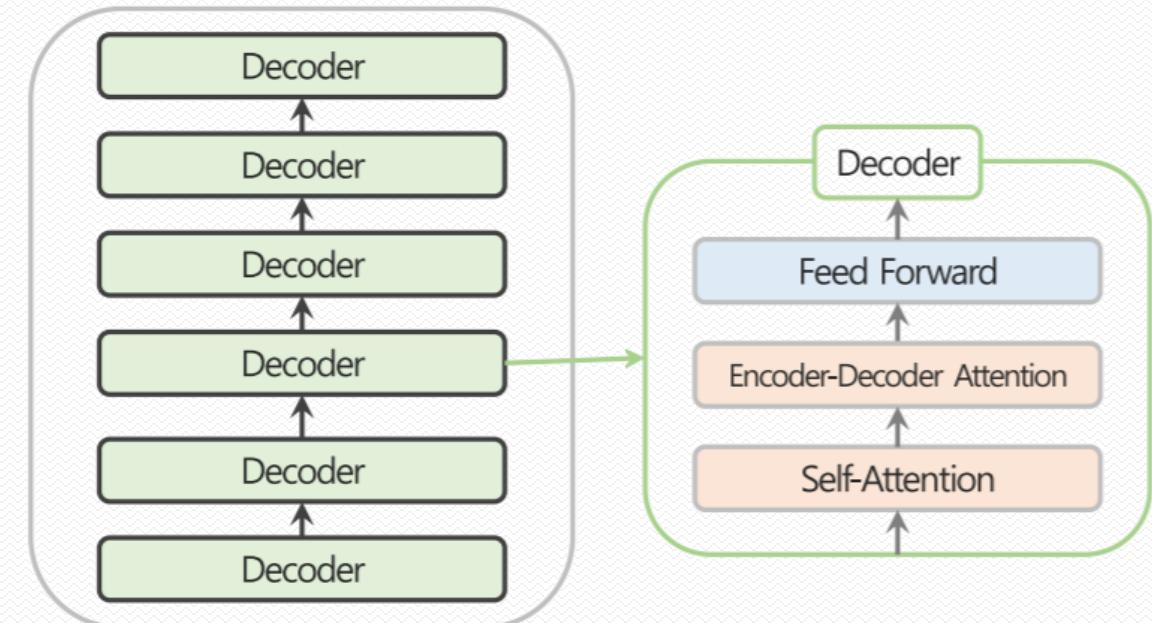
Encoder–Decoder

- Transformer는 Encoder–Decoder 구조의 모델 아키텍쳐
- 세부적으로 보면 6개의 Encoder–Decoder로 이루어져 있음.

Encoder의 네트워크 구조



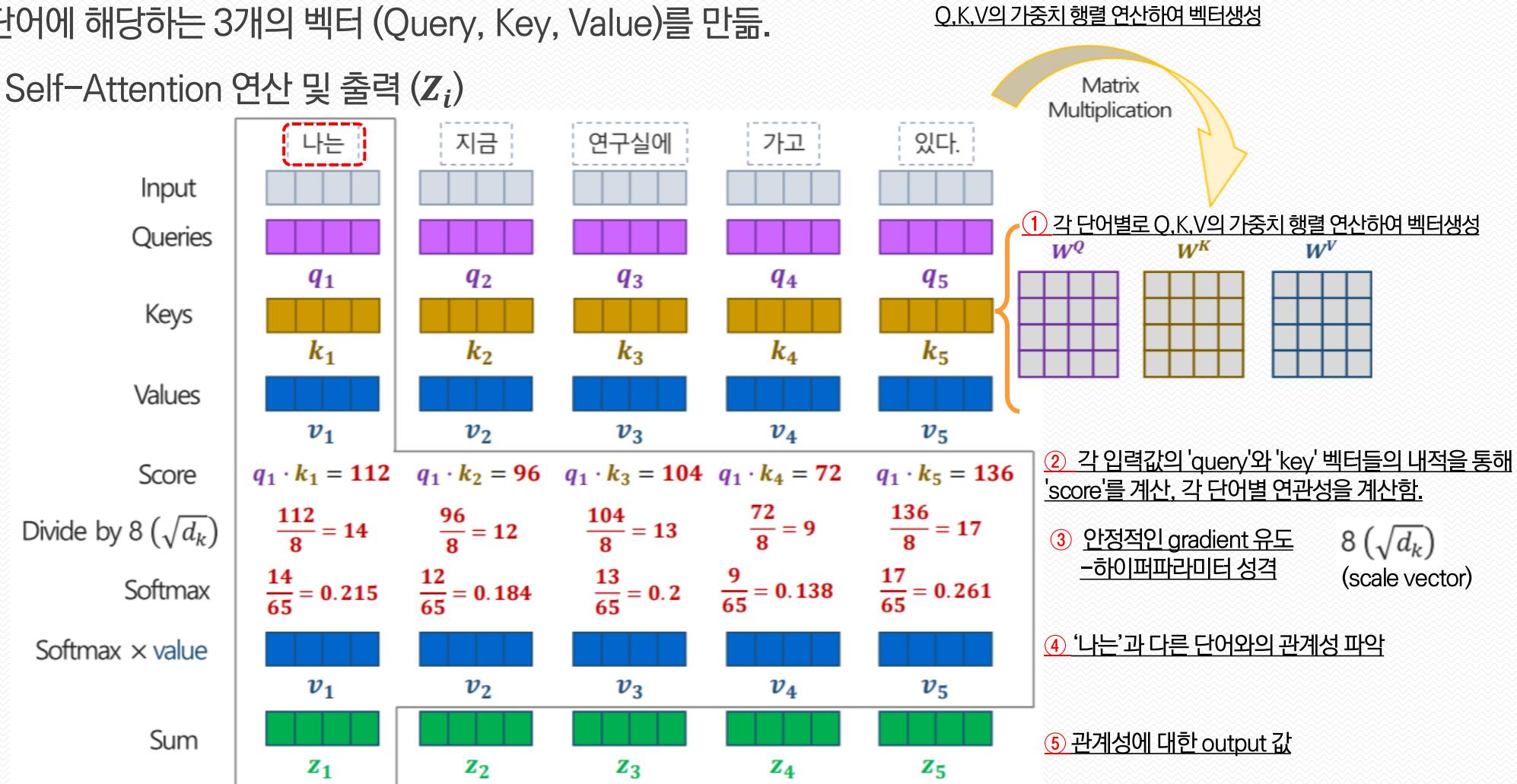
Decoder의 네트워크 구조



Transformer

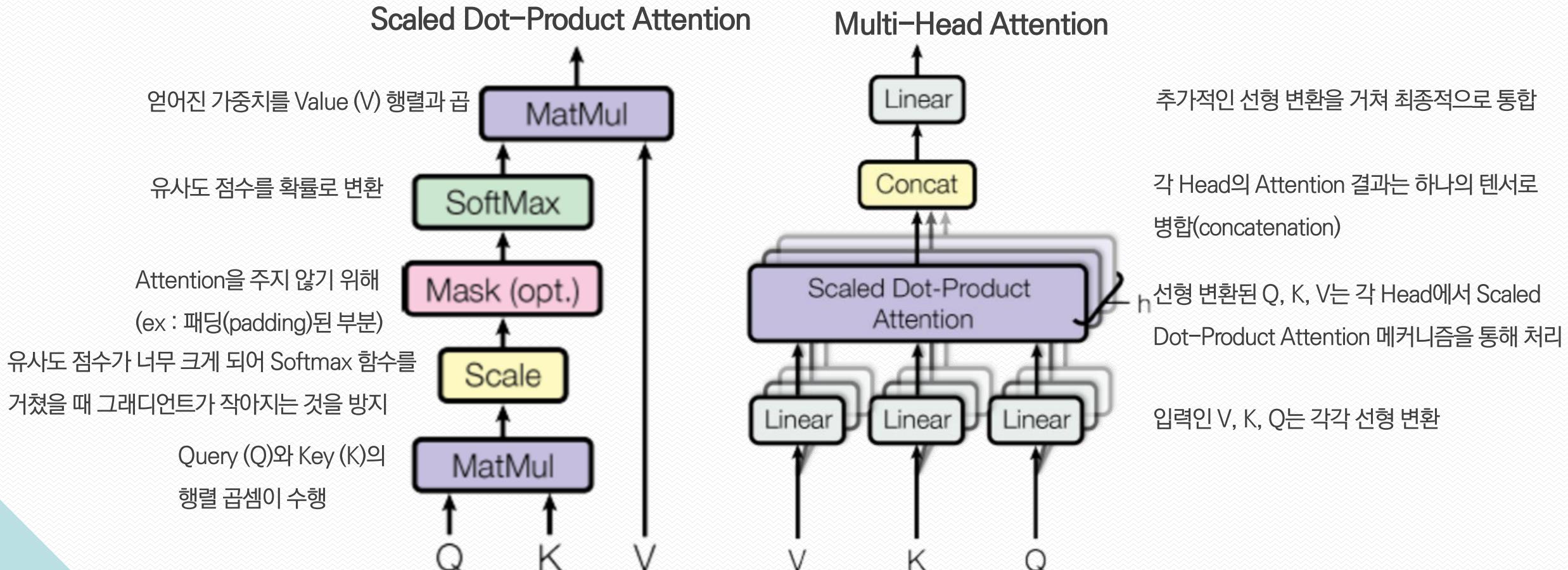
Self Attention

- Encoder의 각 단어에 해당하는 3개의 벡터 (Query, Key, Value)를 만듦.
- Encoder에서의 Self-Attention 연산 및 출력 (Z_i)



Transformer

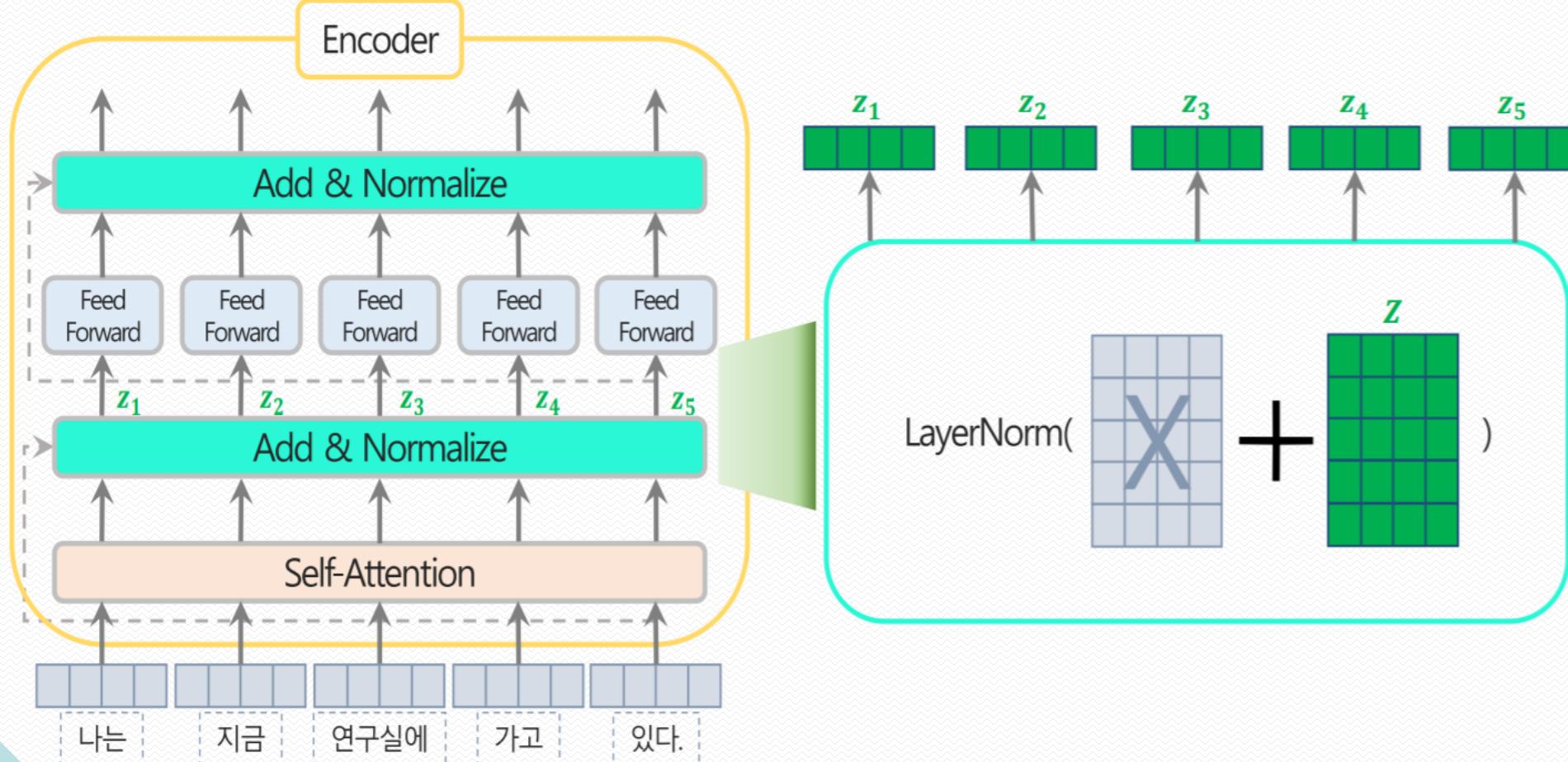
Self-Attention에 대한 Multi-Head Attention에 대한 도식화 (논문에서 아래 그림과 같이 간단하게 표현함)



Transformer

Encoder details

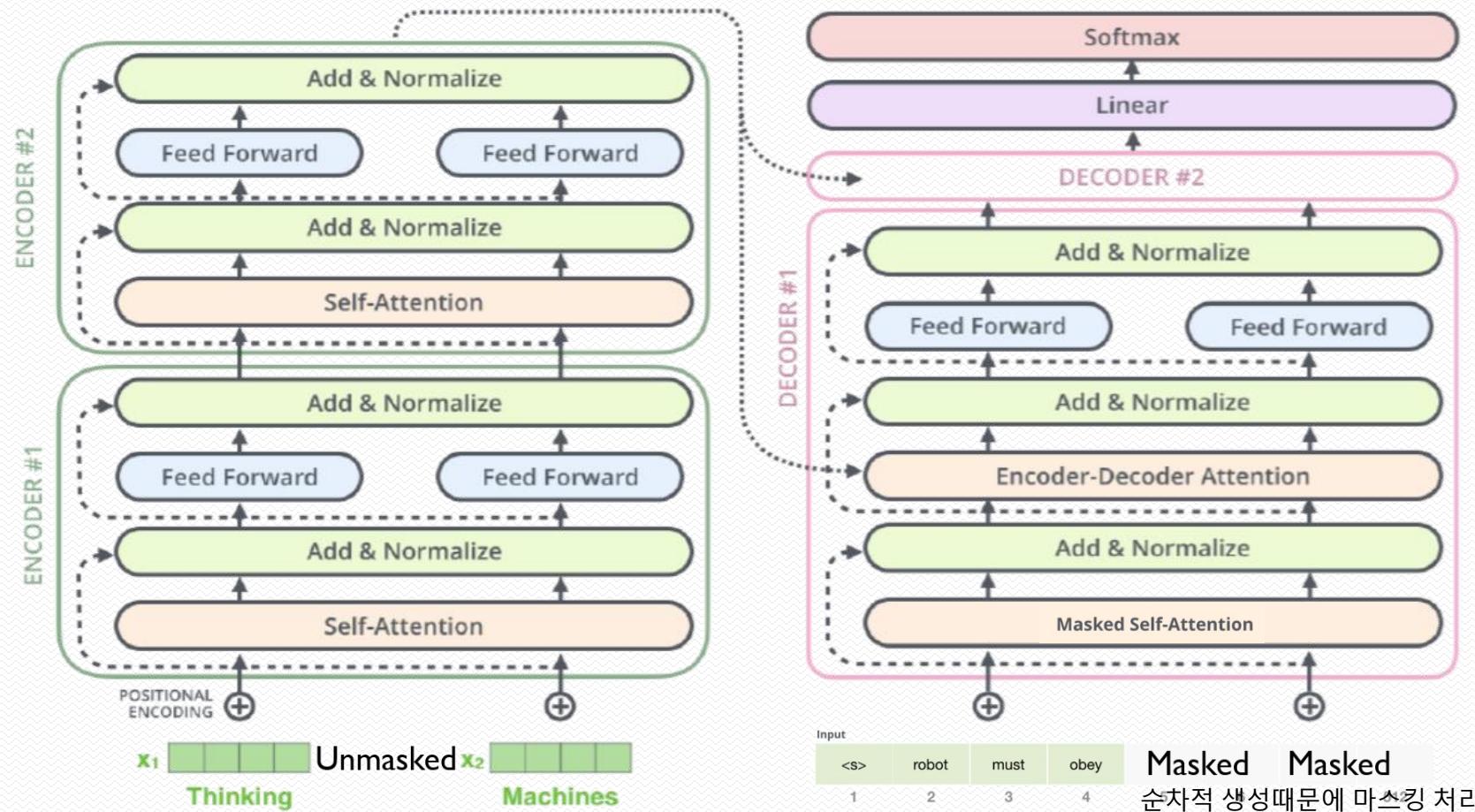
- Residual connection : 입력 데이터와 Self-Attention의 결과를 더함 (Add)
- Layer normalization : residual connection 결과 (입력 ‘x’ + context vector ‘z’)에 대한 정규화 진행 (Normalize)



Transformer

Encoder– Decoder의 차이점

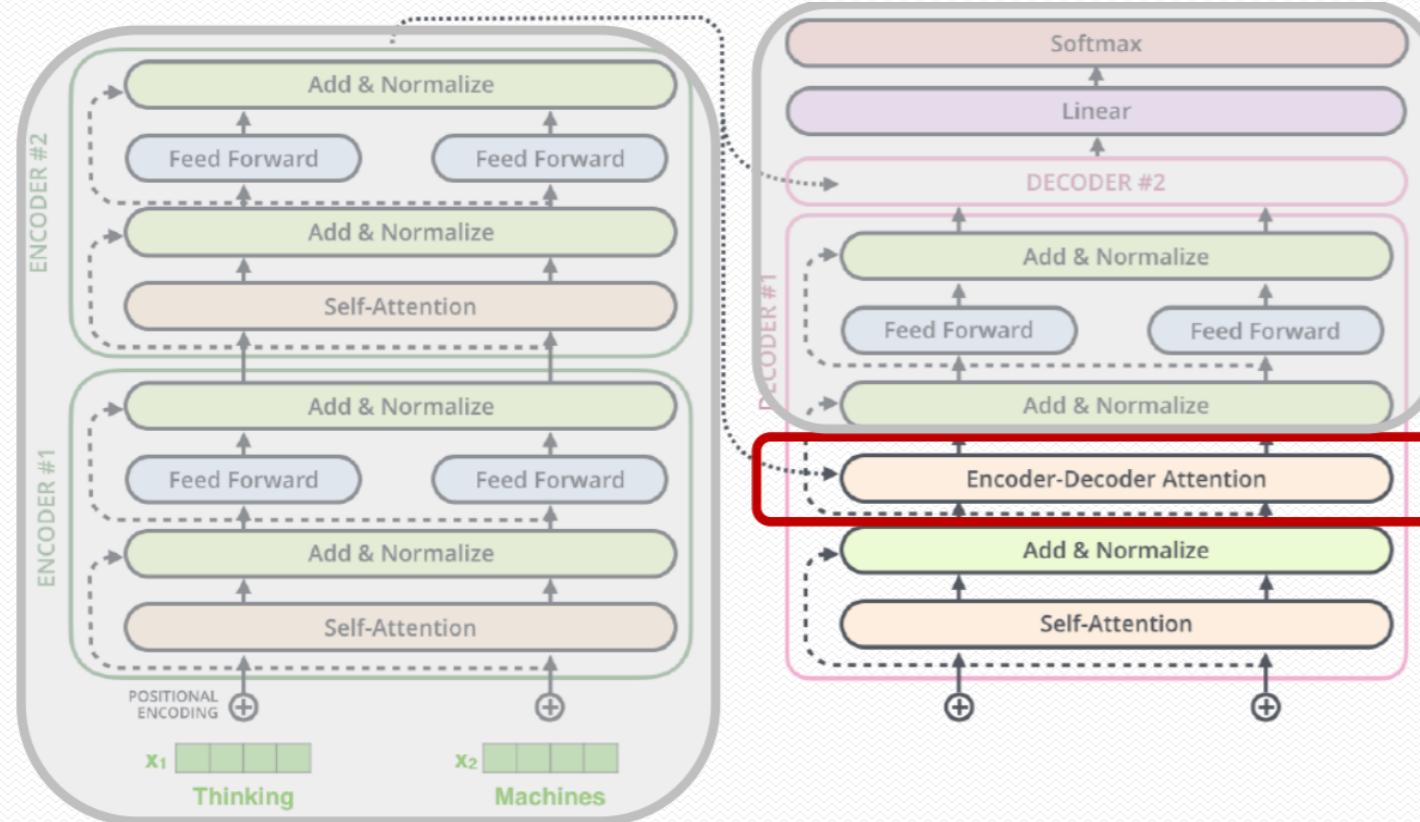
- 차이점1 : 디코더의 Self–Attention는 미래 위치로의 정보 흐름을 방지하기 위해 'Masking' 기법을 사용.
- 차이점2 : Decoder에서는 Encoder–Decoder Attention 레이어가 추가



Transformer

Encoder– Decoder의 차이점

- Decoder의 Encoder– Decoder Attention 레이어는 마지막 Encoder에서 출력한 key와 Value행렬을 사용하여 Self–Attention 연산 진행.
- Decoder가 입력 시퀀스 데이터의 적절한 위치에 집중하도록 함.



Transformer

Positional Encoding

- Self-Attention은 RNN과 달리 입력 시퀀스 데이터를 순차적으로 처리하지 않음.
- Positional Encoding : 입력 시퀀스에서 단어의 순서를 표현하기 위한 임베딩 방법(-1 ~ 1)

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

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

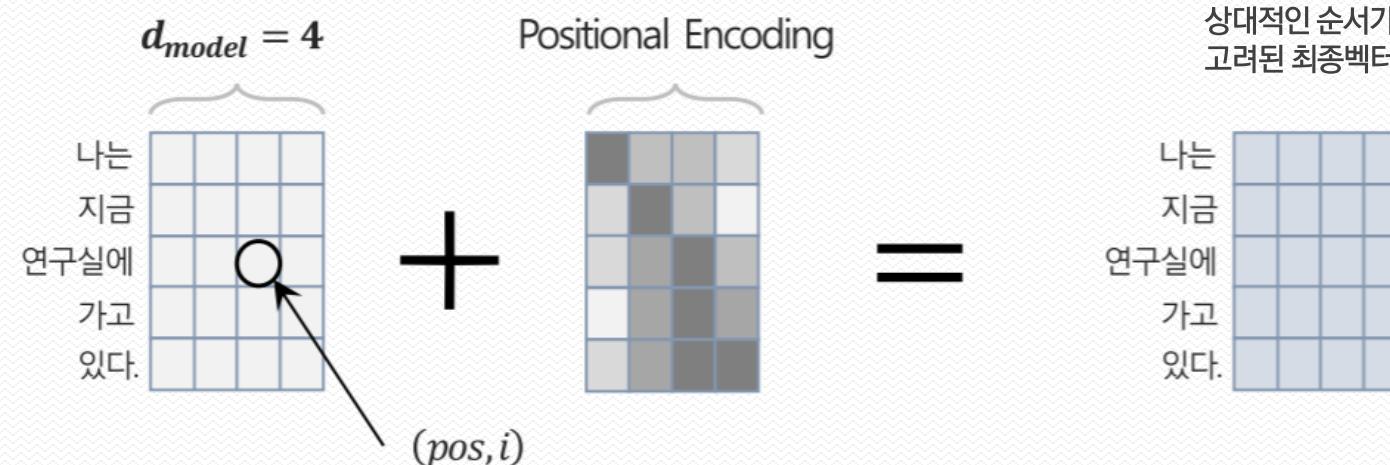
pos: 입력 시퀀스 데이터의 임베딩 벡터의 위치

i: 임베딩 벡터 내 차원의 인덱스

d_{model}: Transformer 모든 층의 출력 차원을 의미

sin함수: 임베딩 벡터의 차원 인덱스가 짝수

cos함수: 임베딩 벡터의 차원 인덱스가 홀수



Transformer

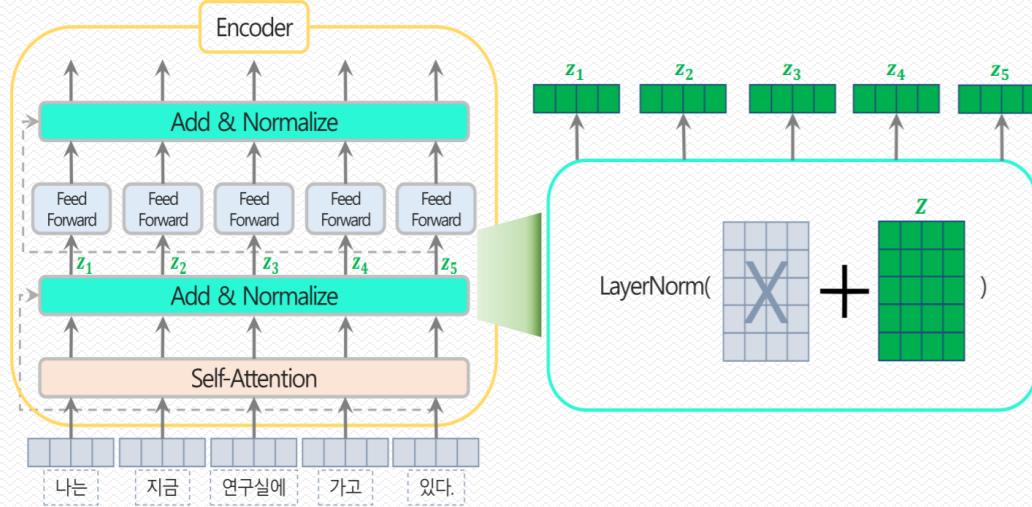
Positional Encoding

- A Simple Example ($n = 10$, dim = 10)
 - ✓ Distances between two positional encoding vectors

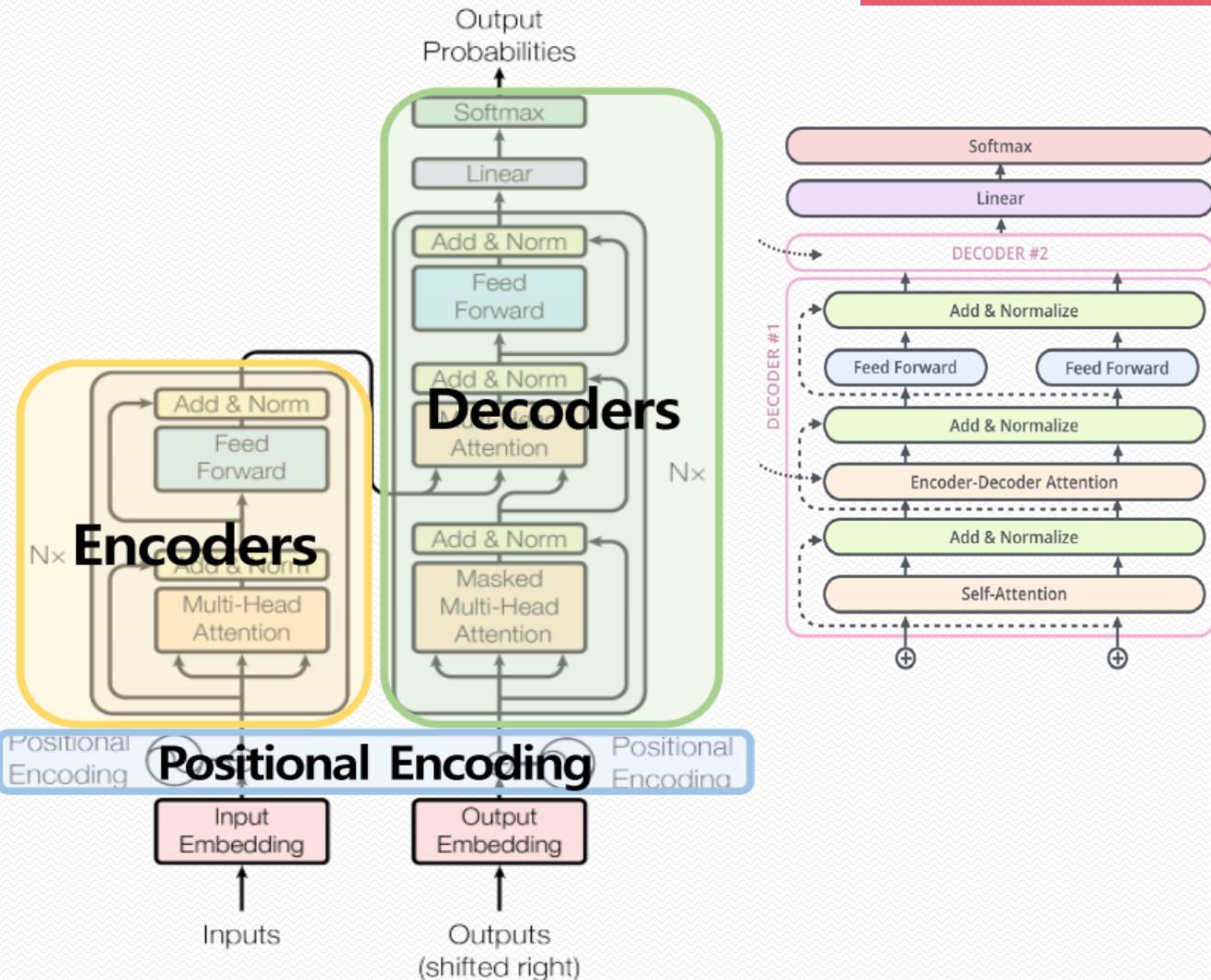
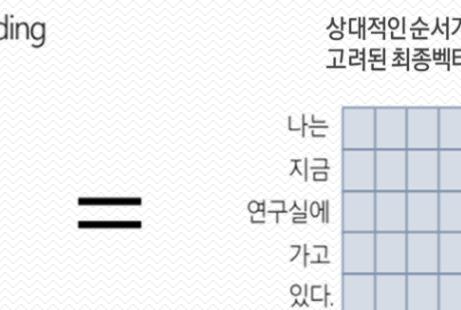
	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
X1	0.000	1.275	2.167	2.823	3.361	3.508	3.392	3.440	3.417	3.266
X2	1.275	0.000	1.104	2.195	3.135	3.511	3.452	3.442	3.387	3.308
X3	2.167	1.104	0.000	1.296	2.468	3.067	3.256	3.464	3.498	3.371
X4	2.823	2.195	1.296	0.000	1.275	2.110	2.746	3.399	3.624	3.399
X5	3.361	3.135	2.468	1.275	0.000	1.057	2.176	3.242	3.659	3.434
X6	3.508	3.511	3.067	2.110	1.057	0.000	1.333	2.601	3.169	3.118
X7	3.392	3.452	3.256	2.746	2.176	1.333	0.000	1.338	2.063	2.429
X8	3.440	3.442	3.464	3.399	3.242	2.601	1.338	0.000	0.912	1.891
X9	3.417	3.387	3.498	3.624	3.659	3.169	2.063	0.912	0.000	1.277
X10	3.266	3.308	3.371	3.399	3.434	3.118	2.429	1.891	1.277	0.000

Transformer

Architecture



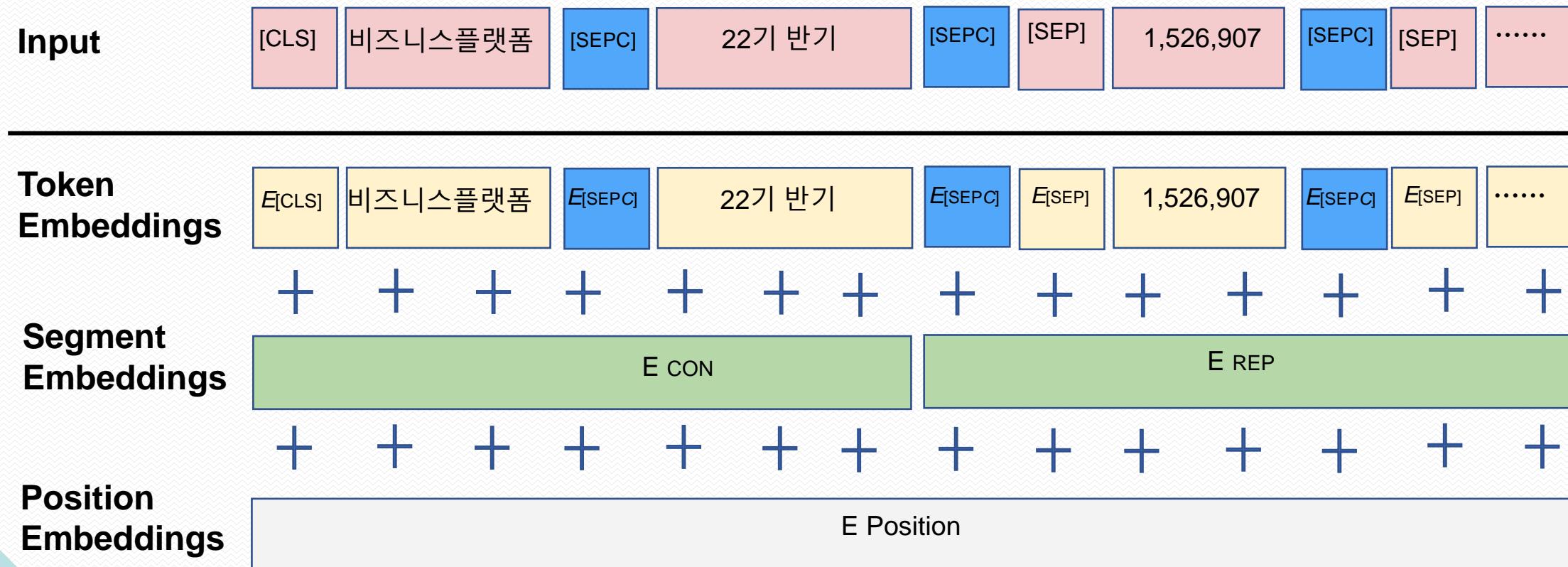
$$\text{Input Vector} + \text{Positional Encoding}_{(pos, i)} = \text{Encoded Vector}$$


[Open in Colab](#)
(코드실습) `Attention_is_All_You_Need_Tutorial_(German_English).ipynb`
[https://github.com/ndb796/Deep-Learning-Paper-Review-and-Practice/blob/master/code_practices/Attention_is_All_You_Need_Tutorial_\(German_English\).ipynb](https://github.com/ndb796/Deep-Learning-Paper-Review-and-Practice/blob/master/code_practices/Attention_is_All_You_Need_Tutorial_(German_English).ipynb)

BERT(Bidirectional Transformers)

BERT 입력 임베딩에 Special Token 추가.

문장이나 단어에 대한 어텐션이나 가중치를 부여하여 어휘사전에서 누락되지 않도록 확장된 임베딩 데이터를 생성

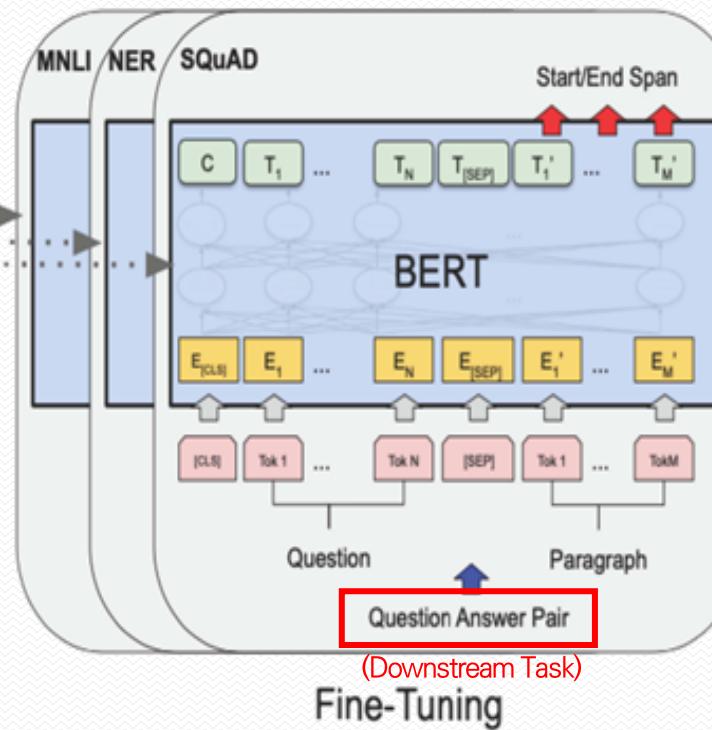
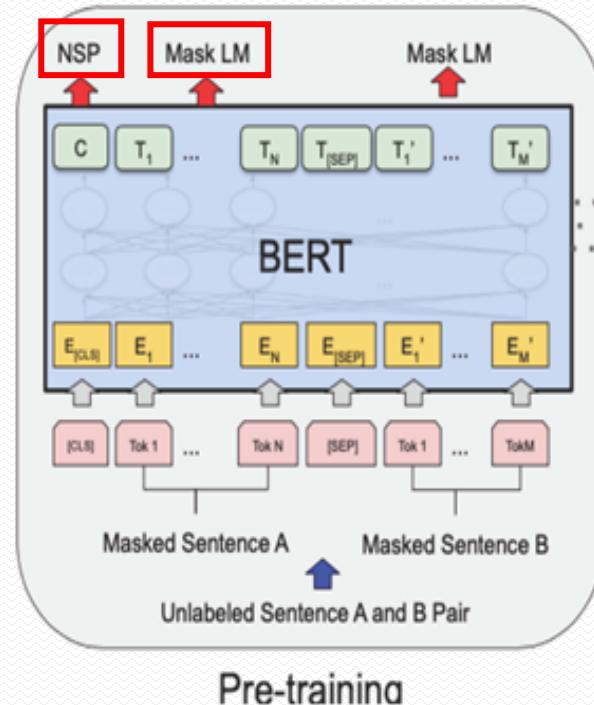


BERT(Bidirectional Transformers)

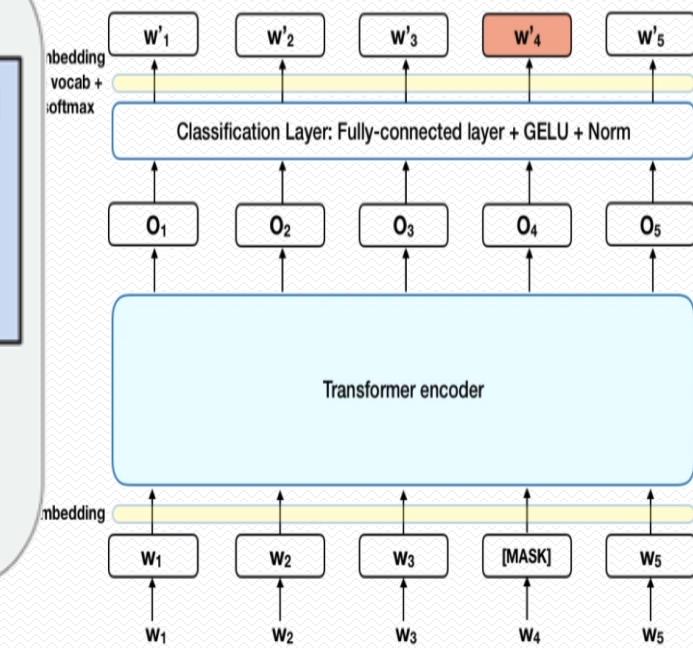
BERT에 대한 전반적인 Pre-training 및 Fine-tuning 절차를 지님.

원래 Transformer와 BERT의 가장 큰 차이점은 마스크언어모델, NSP(Next Sentence Prediction)과제를 수행하기 위한 마지막 예측 레이어의 존재 여부임.

BERT 구조



Masked Language Model



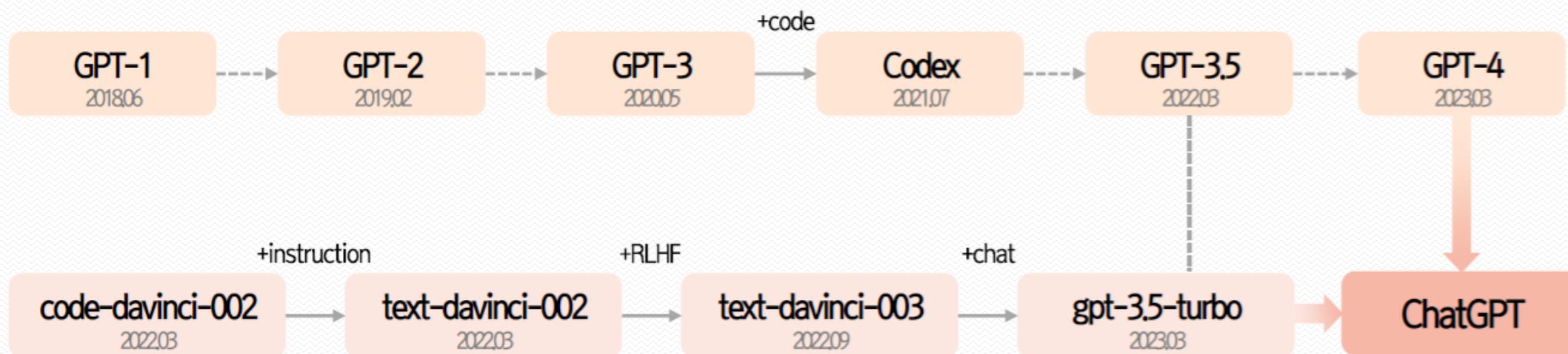
GPT(Generative Pre-trained Transformers)

GPT (Generative Pre-training Transformers)

- Transformer의 생성형 Decoder 구조 기반의 Autoregressive모델
- Autoregressive : 주어진 입력의 일부를 사용하여 이후에 오는 부분을 예측(다음 단어를 맞추는 방식)

Emergent Abilities for LLMs : 작은 모델에서는 나타나지 않지만, 거대한 모델에서 발현되는 능력

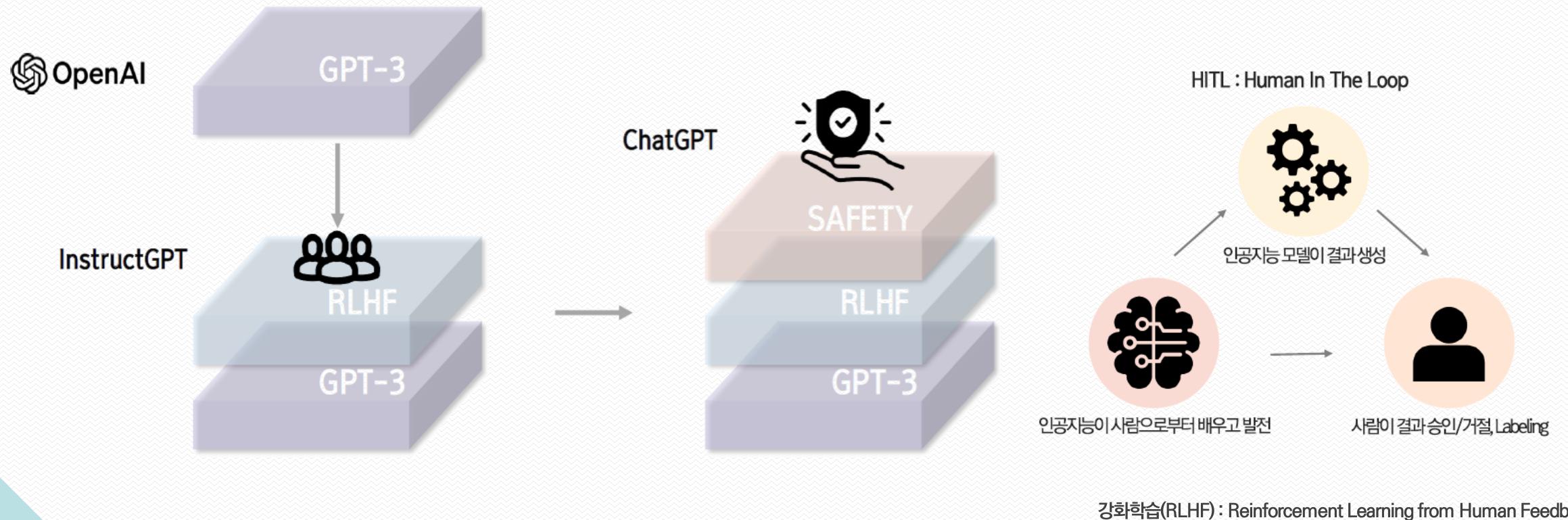
ChatGPT : 2022.11 OpenAI에서 출시한 대화형 인공지능 챗봇



GPT(Generative Pre-training Transformers)

ChatGPT

- Instruct GPT와 유사한 방식으로 학습되었지만, 대화에 최적화&안전강화 (환각 : Hallucination)
- 인간 피드백 기반 강화학습(*RLHF)을 기존 GPT-3에 도입하여 사용자의 지시를 잘 수행하도록 함.



챗GPT API 활용 사례

V. (실습)챗GPT API 활용

음성 비서 [\(Link\)](#)



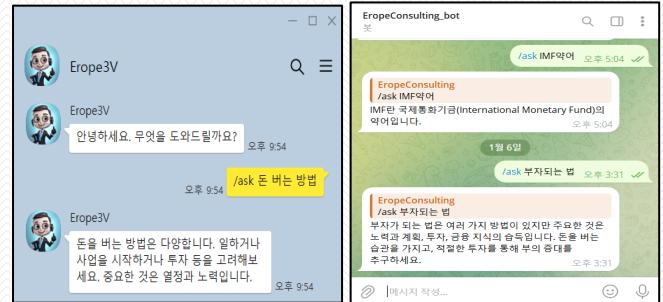
텍스트를 음성으로
변환하는 gTTS와 음성
파일을 텍스트로
변환해주는 OPENAI의
Whisper API를 통해 만든
음성비서를 만나보세요.



AI 챗봇 [\(Link\)](#)



아마존 클라우드 서비스를
통해 텔레그램과
카카오톡을 통해 언제라도
24시간 Chatbot을 만나
보실 수 있습니다.



회사 챗봇 [\(Link\)](#)



회사 회사, 또는 특정 정보를
주입해서 우리 회사만의 챗봇을
만들수 있습니다.



챗GPT API 실습 – 예제코드/환경

V. (실습)챗GPT API 활용



--EROPE-- 예제코드 제공 (https://github.com/EropeConsulting/INTEL_LLM)



OpenAI API KEY 발급받기 (<https://platform.openai.com/api-keys>)



AWS 계정 생성하기(<https://aws.amazon.com/ko/>)



스트림릿 가입 (<https://streamlit.io/cloud>) : 웹 애플리케이션을 간단하게 만들 수 있는 라이브러리



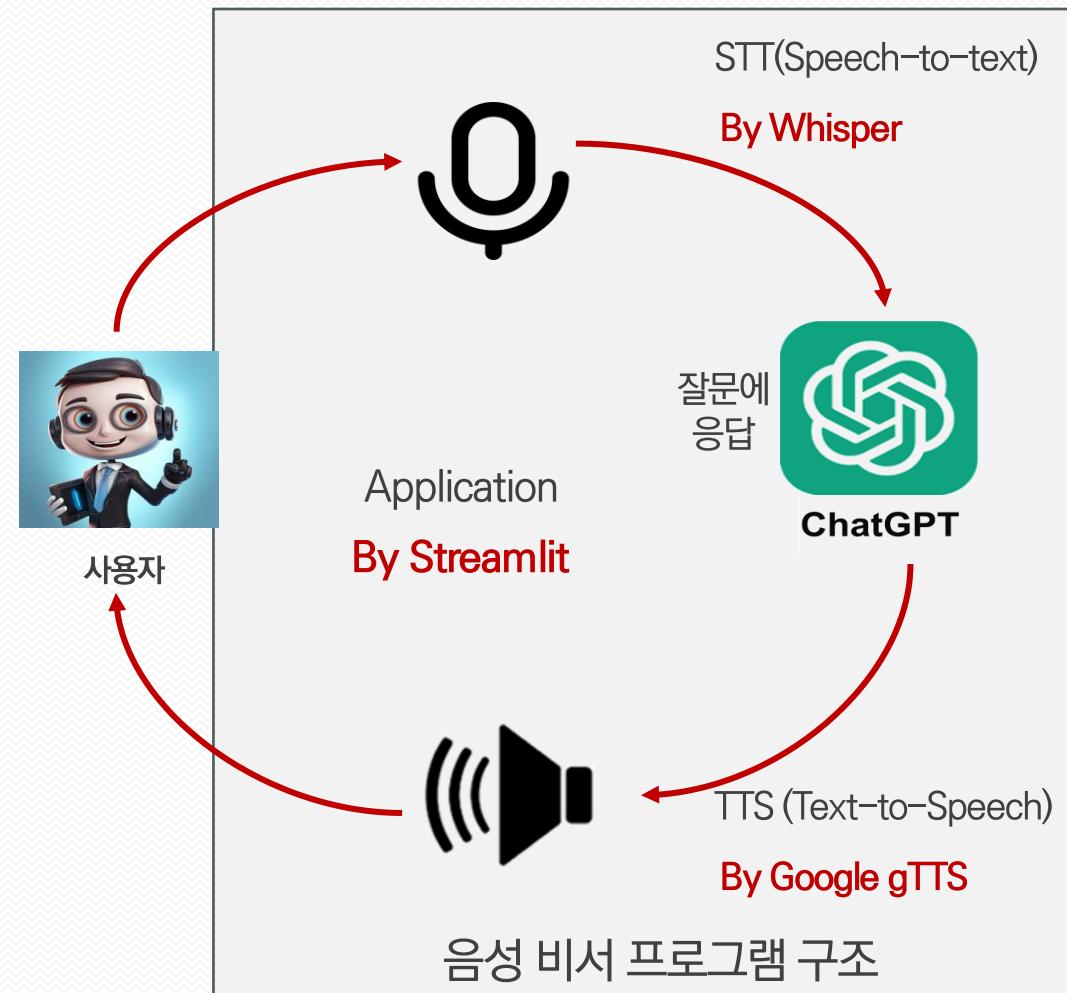
Visual Studio Code (<https://code.visualstudio.com/>)



깃허브 계정 생성 (<https://github.com/>)

챗GPT API 실습 - 나만의 음성 비서

사용자가 음성으로 질문하면 ChatGPT가 음성으로 답변하는 프로그램



챗GPT API 실습 - 나만의 음성 비서

V. (실습)챗GPT API 활용

음성 비서 프로그램 배포 완료 화면

The screenshot shows a web-based application for a speech-to-text program. On the left, there's a sidebar with an 'OPENAI API 키' input field containing placeholder text 'Enter Your API Key' and a visibility icon. Below it, 'GPT 모델' settings show 'gpt-4' selected (indicated by a red dot). At the bottom of the sidebar is a '초기화' button. The main content area has a title '음성 비서 프로그램'. To the right of the title is a 'Share' button and other icons. A section titled '음성비서 프로그램에 관하여' contains a list of bullet points about the program's technology stack. Below this is a '질문하기' section with a button labeled '클릭하여 녹음하기' and a '질문/답변' section.

OPENAI API 키
Enter Your API Key

GPT 모델
● gpt-4
○ gpt-3.5-turbo

초기화

음성비서 프로그램에 관하여

- 음성비서 프로그램의 UI는 스트림릿을 활용했습니다.
- STT(Speech-To-Text)는 OpenAI의 Whisper AI를 활용했습니다.
- 답변은 OpenAI의 GPT 모델을 활용했습니다.
- TTS(Text-To-Speech)는 구글의 Google Translate TTS를 활용했습니다.

질문하기

클릭하여 녹음하기

질문/답변

< Manage app

챗GPT API 실습 - 나만의 음성 비서

1. 기본 정보 입력

```
1 ##### 기본 정보 입력 #####
2 import streamlit as st
3 # audiorecorder 패키지 추가
4 from audiorecorder import audiorecorder
5 # OpenAI 패키지 추가
6 import openai
7 # 파일 삭제를 위한 패키지 추가
8 import os
9 # 시간 정보를 위한 패키지 추가
10 from datetime import datetime
11 # TTS 패키지 추가
12 from gtts import gTTS
13 # 음원 파일 재생을 위한 패키지 추가
14 import base64
```

챗GPT API 실습 - 나만의 음성 비서

1. 텍스트를 음성파일로 변환하는 gTTS 사용법 C:\voicebot> pip install gTTS

```
35 def TTS(response):
36     # gTTS 를 활용하여 음성 파일 생성
37     filename = "output.mp3"
38     tts = gTTS(text=response,lang="ko")
39     tts.save(filename)
40
41     # 음원 파일 자동 재생
42     with open(filename, "rb") as f:
43         data = f.read()
44         b64 = base64.b64encode(data).decode()
45         md = f"""
46             <audio autoplay="True">
47                 <source src="data:audio/mp3;base64,{b64}" type="audio/mp3">
48             </audio>
49             """
50         st.markdown(md,unsafe_allow_html=True,)
51     # 파일 삭제
52     os.remove(filename)
```

챗GPT API 실습 - 나만의 음성 비서

2. 음성파일을 텍스트로 변환하는 Whisper API 사용법 C:\voicebot> pip install openai==0.28.1

```
17 def STT(audio):
18     # 파일 저장
19     filename='input.mp3'
20     audio.export(filename, format="mp3")
21     # 음원 파일 열기
22     audio_file = open(filename, "rb")
23     # Whisper 모델을 활용해 텍스트 얻기
24     transcript = openai.Audio.transcribe("whisper-1", audio_file)
25     audio_file.close()
26     # 파일 삭제
27     os.remove(filename)
28     return transcript["text"]
```

챗GPT API 실습 - 나만의 음성 비서

3.1 스트림릿 기본 설정 영역 구현하기

```

56 ##### 메인 함수 #####
57 def main():
58     # 기본 설정
59     st.set_page_config(
60         page_title="음성 비서 프로그램",
61         layout="wide")
62

```

중략...

C:\> pip install streamlit

```

73     # 제목
74     st.header("음성 비서 프로그램")
75     # 구분선
76     st.markdown("---")
77
78     # 기본 설명
79     with st.expander("음성비서 프로그램에 관하여", expanded=True):
80         st.write(
81             """
82             - 음성비서 프로그램의 UI는 스트림릿을 활용했습니다.
83             - STT(Speech-To-Text)는 OpenAI의 Whisper AI를 활용했습니다.
84             - 답변은 OpenAI의 GPT 모델을 활용했습니다.
85             - TTS(Text-To-Speech)는 구글의 Google Translate TTS를 활용했
86             """
87         )
88
89     st.markdown("")

```

챗GPT API 실습 - 나만의 음성 비서

3. 2 스트림릿 옵션 선택 영역 구현하기

```
90     # 사이드바 생성
91     with st.sidebar:
92
93         # Open AI API 키 입력받기
94         openai.api_key = st.text_input(label="OPENAI API 키", placeholder="Enter Your API Key", value="", type="password")
95
96         st.markdown("---")
97
98         # GPT 모델을 선택하기 위한 라디오 버튼 생성
99         model = st.radio(label="GPT 모델", options=["gpt-4", "gpt-3.5-turbo"])
100
101        st.markdown("---")
102
103        # 리셋 버튼 생성
104        if st.button(label="초기화"):
105            # 리셋 코드
106            st.session_state["chat"] = []
107            st.session_state["messages"] = [{"role": "system", "content": "You are a thoughtful assistant.\n| | | | | | | | | | | | | | | | | | Respond to all input in 25 words and answer in korea"}]
108            st.session_state["check_reset"] = True
109
```

챗GPT API 실습 - 나만의 음성 비서

3. 3 스트림릿 상태를 저장하기 위한 session_state 함수 구현하기

st.session_state는 스트림릿에서 사용하는 저장공간으로 session_state를 이용하면 프로그램을 재실행하더라도 정보가 유지됨. 이전 질문과 답변 모두 차례로 누적하여 저장함

```
--  
63     # session state 초기화  
64     if "chat" not in st.session_state:  
65         st.session_state["chat"] = []  
66  
67     if "messages" not in st.session_state:  
68         st.session_state["messages"] = [{"role": "system", "content": "You are a thoughtful assistant. \  
69             | | | | | | | | Respond to all input in 25 words and answer in korea"}]  
70  
71     if "check_reset" not in st.session_state:  
72         st.session_state["check_reset"] = False
```

챗GPT API 실습 - 나만의 음성 비서

3.4 스트림릿 오디오 레코더를 활용하여 음성 녹음 구현하기 | C:\voicebot>pip install streamlit-audierecorder==0.0.2

```
112     # 기능 구현 공간
113     col1, col2 = st.columns(2)
114     with col1:
115         # 왼쪽 영역 작성
116         st.subheader("질문하기")
117         # 음성 녹음 아이콘 추가
118         audio = audierecorder("클릭하여 녹음하기", "녹음중...")
119         if (audio.duration_seconds > 0) and (st.session_state["check_reset"]==False):
120             # 음성 재생
121             st.audio(audio.export().read())
122             # 음원 파일에서 텍스트 추출
123             question = STT(audio)
124
125             # 채팅을 시작화하기 위해 질문 내용 저장
126             now = datetime.now().strftime("%H:%M")
127             st.session_state["chat"] = st.session_state["chat"]+ [{"user":now, question}]
128             # GPT 모델에 넣을 프롬프트를 위해 질문 내용 저장
129             st.session_state["messages"] = st.session_state["messages"]+ [{"role": "user", "content": question}]
```

챗GPT API 실습 - 나만의 음성 비서

3.5 ChatGPT API로 질문하고 답변 구하기

```
30     def ask_gpt(prompt, model):
31         response = openai.ChatCompletion.create(model=model, messages=prompt)
32         system_message = response["choices"][0]["message"]
33         return system_message["content"]
```

중략...

```
131     with col2:
132         # 오른쪽 영역 작성
133         st.subheader("질문/답변")
134         if (audio.duration_seconds > 0) and (st.session_state["check_reset"]==False):
135             # ChatGPT에게 답변 얻기
136             response = ask_gpt(st.session_state["messages"], model)
137
138             # GPT 모델에 넣을 프롬프트를 위해 답변 내용 저장
139             st.session_state["messages"] = st.session_state["messages"]+ [{"role": "system", "content": response}]
140
141             # 채팅 시작화를 위한 답변 내용 저장
142             now = datetime.now().strftime("%H:%M")
143             st.session_state["chat"] = st.session_state["chat"]+ [("bot",now, response)]
```

챗GPT API 실습 - 나만의 음성 비서

3.6 대화 내용을 채팅 형식으로 시각화하기

```
145     # 채팅 형식으로 시각화 하기
146     for sender, time, message in st.session_state["chat"]:
147         if sender == "user":
148             st.write(f'<div style="display:flex;align-items:center;"> \
149                     <div style="background-color:#007AFF;color:white;border-radius:12px;padding:8px 12px;margin-right:8px;">
150                         {message}</div><div style="font-size:0.8rem;color:gray;">{time}</div></div>', unsafe_allow_html=True)
151             st.write("")
152         else:
153             st.write(f'<div style="display:flex;align-items:center;justify-content:flex-end;"> \
154                     <div style="background-color:lightgray;border-radius:12px;padding:8px 12px;margin-left:8px;">
155                         {message}</div><div style="font-size:0.8rem;color:gray;">{time}</div></div>', unsafe_allow_html=True)
156             st.write("")
```

챗GPT API 실습 - 나만의 음성 비서

5. 깃허브에 코드 및 패키지 정보 업로드하기

5.1 패키지 생성 C:\voicebot> pip freeze > requirements.txt

5.2 깃허브에 코드 및 패키지 정보 업로드하기

The screenshot shows a GitHub repository page for 'EropeConsulting / INTEL_LLM'. The 'Code' tab is selected. In the file list, several files are listed under the 'voicebot' directory. Two files are highlighted with red boxes: 'ch03_voicebot_student.py' and 'requirements.txt'. The table below provides details for these files.

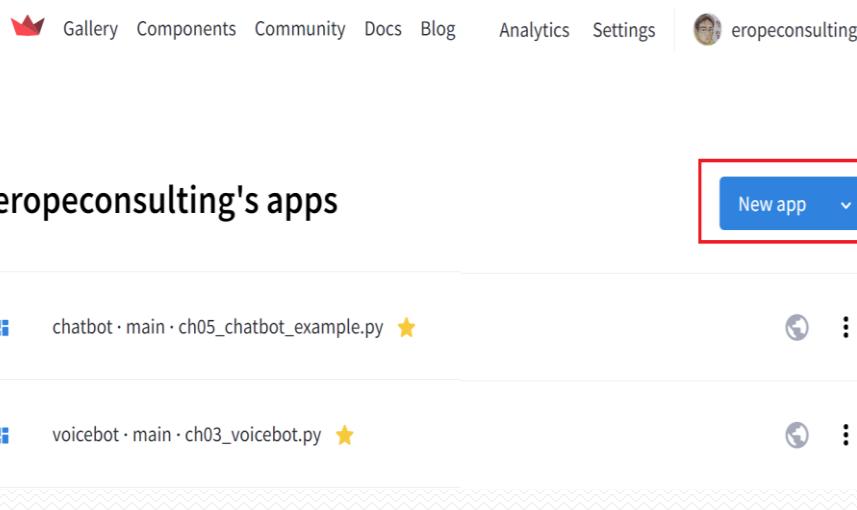
Name	Last commit message	Last commit date
..	Add files via upload	2230363 · 1 minute ago
ch03_voicebot_student.py	Add files via upload	1 minute ago
output.mp3	Add files via upload	last week
packages.txt	Add files via upload	last week
requirements.txt	Add files via upload	last week

챗GPT API 실습 - 나만의 음성 비서

V. (실습)챗GPT API 활용

6. 스트림릿 사이트와 깃허브 연동하기

6.1 스트림릿 가입 <https://streamlit.io/cloud>



6.2 스트림릿 사이트와 깃허브 연동하기

Deploy an app

Repository

EropeConsulting/lecture_LLM

Paste GitHub URL

Branch

main

Main file path

ch03_voicebot.py

App URL (Optional)

lectureLLM-jous6yyn7mcpaxnappazr2

.streamlit.app

Domain is available

Advanced settings...

Deploy!

챗GPT API 실습 - 나만의 음성 비서

V. (실습)챗GPT API 활용

음성 비서 프로그램 배포 완료 화면

The screenshot shows a web-based application for a speech-to-text program. On the left, there's a sidebar with an 'OPENAI API 키' input field containing placeholder text 'Enter Your API Key' and a visibility icon. Below it, 'GPT 모델' settings show 'gpt-4' selected (indicated by a red dot). At the bottom of the sidebar is a '초기화' button. The main content area has a title '음성 비서 프로그램'. To the right of the title is a 'Share' button and other icons. A section titled '음성비서 프로그램에 관하여' contains a list of bullet points about the program's technology stack. Below this is a '질문하기' section with a button labeled '클릭하여 녹음하기' and a '질문/답변' section.

OPENAI API 키
Enter Your API Key

GPT 모델
● gpt-4
○ gpt-3.5-turbo

초기화

음성비서 프로그램에 관하여

- 음성비서 프로그램의 UI는 스트림릿을 활용했습니다.
- STT(Speech-To-Text)는 OpenAI의 Whisper AI를 활용했습니다.
- 답변은 OpenAI의 GPT 모델을 활용했습니다.
- TTS(Text-To-Speech)는 구글의 Google Translate TTS를 활용했습니다.

질문하기

클릭하여 녹음하기

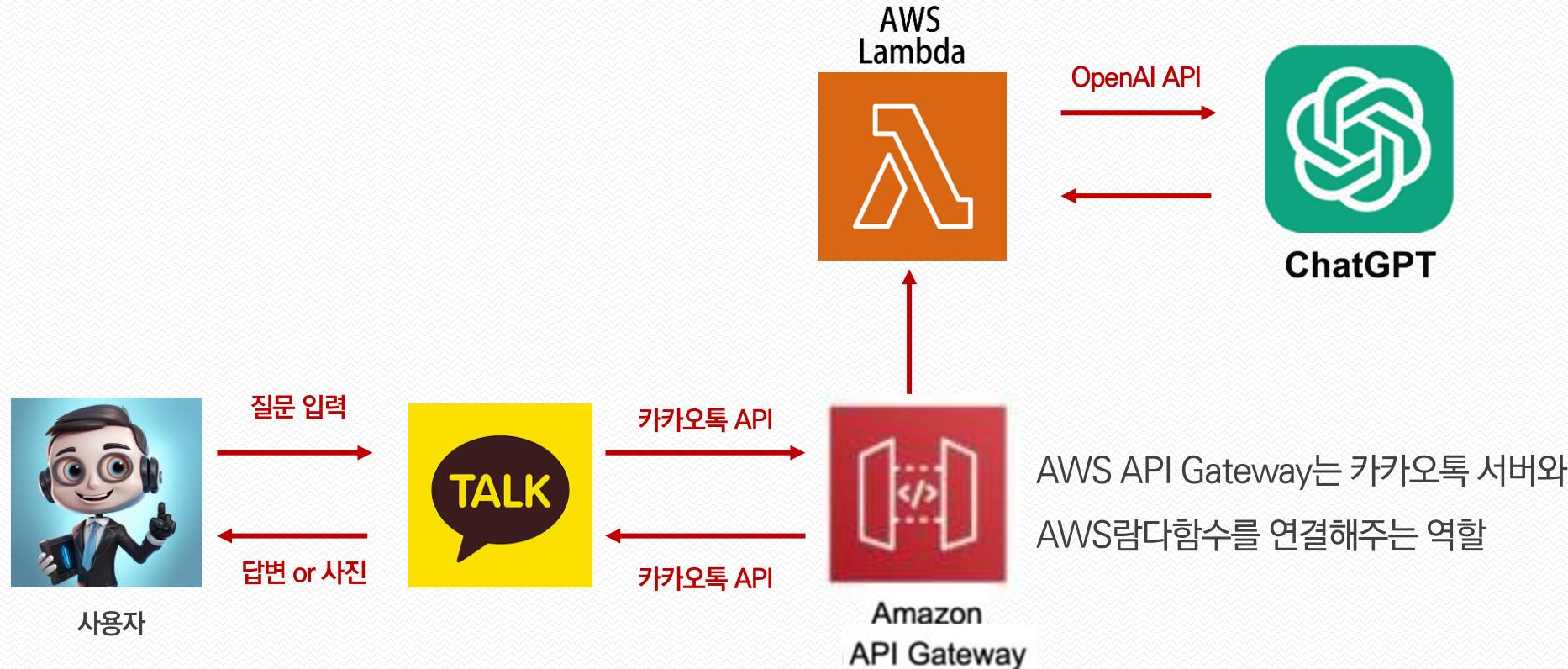
질문/답변

< Manage app

챗GPT API 실습 - 카카오톡 챗봇

V. (실습)챗GPT API 활용

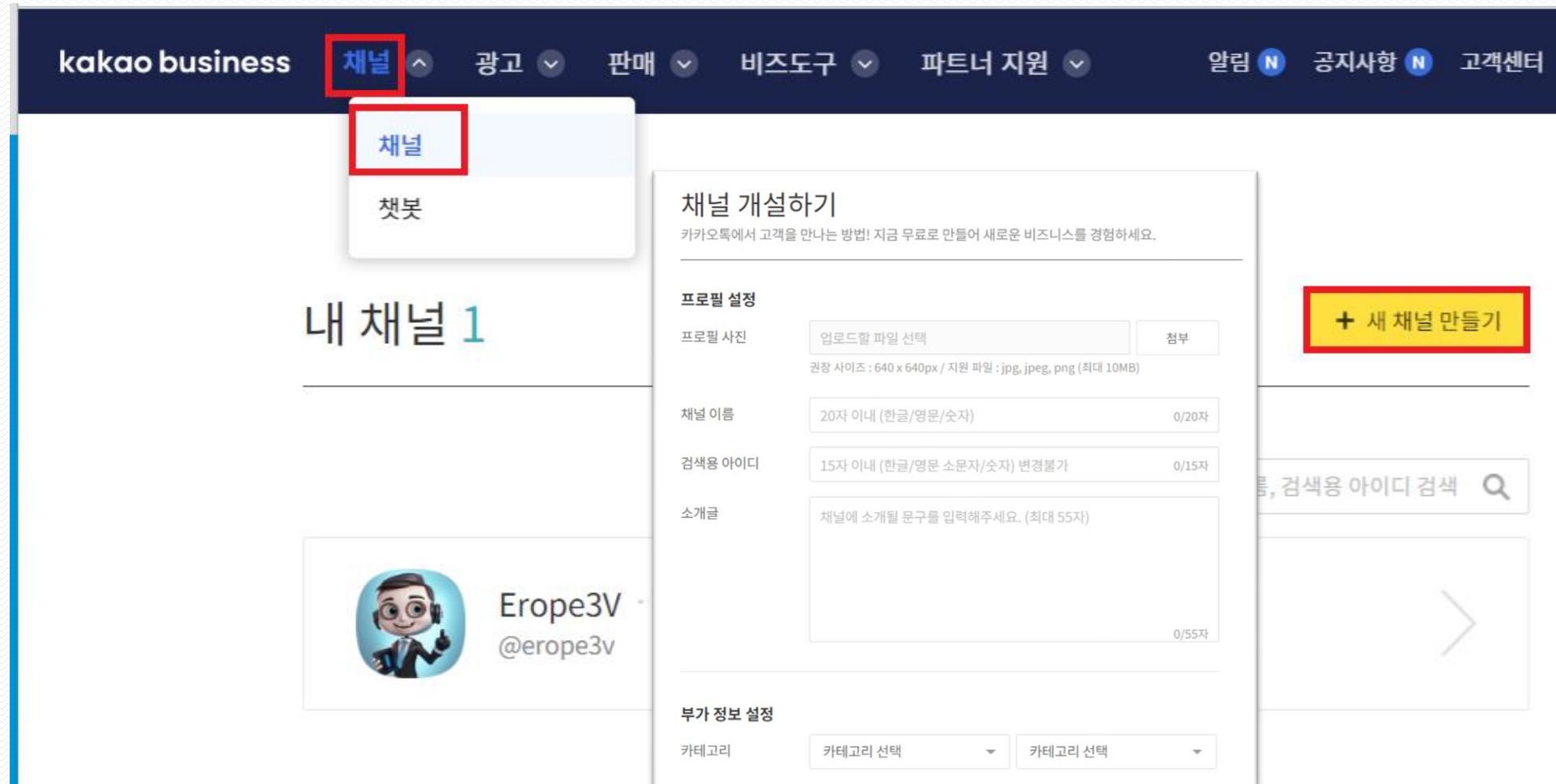
1. AWS를 활용한 카카오 챗봇의 구조



챗GPT API 실습 - 카카오톡 챗봇

1. 카카오톡 챗봇 생성하기

- 카카오 비즈니스 채널 만들기 <https://business.kakao.com/>



챗GPT API 실습 - 카카오톡 챗봇

1. 카카오톡 챗봇 생성하기

- 챗봇 관리자 센터에서 챗봇 제작하기 <https://business.kakao.com/>

The image shows two screenshots of the Kakao Business Chatbot Manager Center.

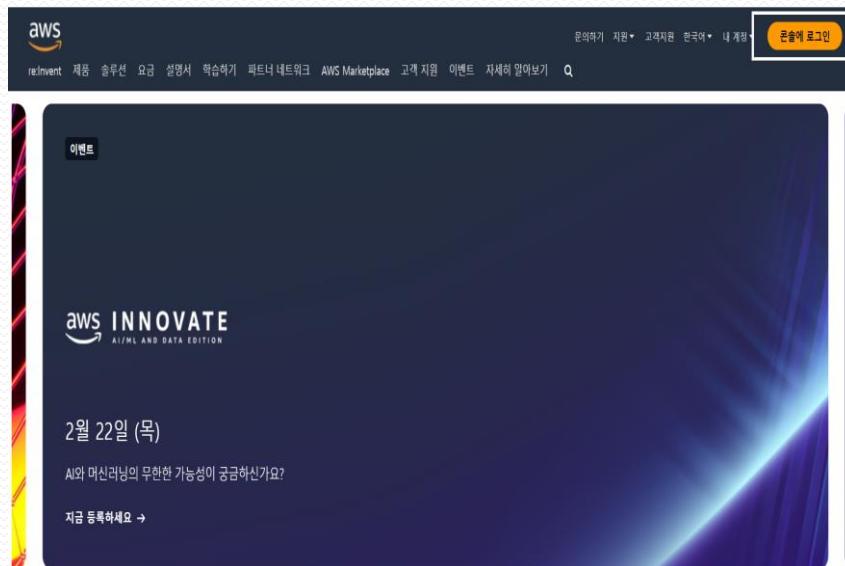
Left Screenshot: Shows the main interface of the Chatbot Manager Center. A red box highlights the '채널' (Channel) tab in the top navigation bar. Another red box highlights the '챗봇' (Chatbot) section under the channel list, which shows a bot named 'Erope3V' connected to a 'KakaoTalk' channel. A third red box highlights the '챗봇' (Chatbot) button in the sidebar menu. At the bottom, there are two buttons: '챗봇 관리자센터 바로가기' (Go to Chatbot Manager Center) and '챗봇 제작사례 확인하기' (Check Chatbot Case Studies).

Right Screenshot: Shows the '내 챗봇' (My Chatbots) page. It lists existing chatbots: 'Erope3V' (selected, highlighted with a red box), '보이스봇 beta' (highlighted with a red box), and '카카오톡 챗봇' (highlighted with a red box). A fourth red box highlights the '카카오톡 챗봇 생성' (Create KakaoTalk Chatbot) button in a modal window. The modal window contains fields for '카카오톡 챗봇 이름 설정' (Set KakaoTalk Chatbot Name) and 'GPTAPI마스터' (Master GPTAPI). Buttons at the bottom right of the modal are '취소' (Cancel) and '확인' (Confirm).

챗GPT API 실습 - 카카오톡 챗봇

2. AWS 계정 생성하기

- <https://aws.amazon.com/ko/>



A screenshot of the AWS Console Home page. The top navigation bar shows 'aws' and various service icons. The main area is titled '콘솔 홈' (Console Home). It features a section for recently visited services: 'Lambda' and 'API Gateway' are highlighted with a red box. To the right, there's a section for '애플리케이션 (0) 정보' (Application (0) Information) with a 'Create Application' button. At the bottom, there are links for 'AWS 시작' (AWS Start), 'AWS Health 정보' (AWS Health Information), and '비용 및 사용량 정보' (Cost and Usage Information).

챗GPT API 실습 - 카카오톡 챗봇

3. AWS람다 함수 생성하기

The image shows two screenshots of the AWS Lambda console.

Left Screenshot: Function Creation

- The top navigation bar shows the AWS logo, services menu, search bar, and navigation icons.
- The main navigation bar has 'Lambda' selected (highlighted with a red box).
- The breadcrumb path is 'Lambda > 함수 > 함수 생성'.
- The title '함수 생성' is displayed with a '정보' link.
- A note says '다음 옵션 중 하나를 선택하여 함수를 생성합니다.'.
- Three options are shown:
 - 새로 작성** (selected, highlighted with a red box): '간단한 Hello World 예제는 시작하십시오.'
 - 블루프린트 사용**: '샘플 코드 및 구축 Lambda 애플리케이션을 위한 구성 사전 설정을 일반적인 사용 사례를 살펴봅니다.'
 - 컨테이너 이미지**: '함수에 대해 배포할 컨테이너 이미지를 선택합니다.'
- 기본 정보** section:
 - 함수 이름**: Input field containing 'myFunctionName'.
 - Description: '함수의 용도를 설명하는 이름을 입력합니다.'
 - 런타임 정보**: 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.'

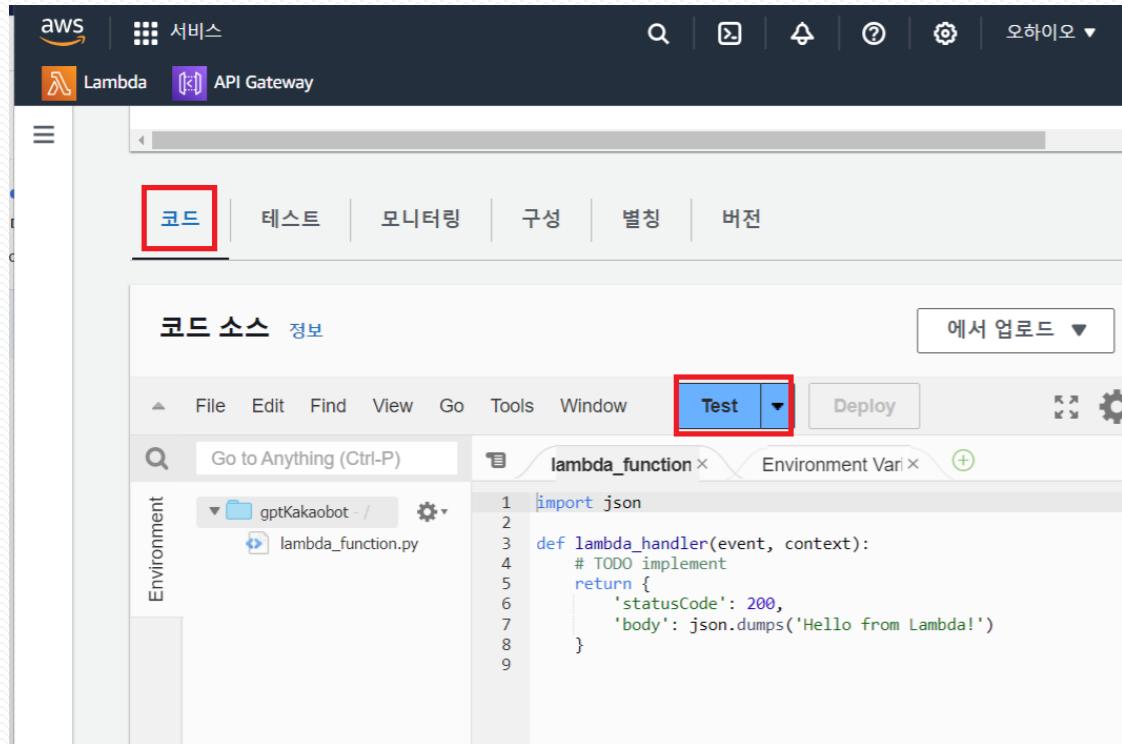
Right Screenshot: Function Details

- The top navigation bar shows the AWS logo, services menu, search bar, and navigation icons.
- The main navigation bar has 'Lambda' selected.
- The breadcrumb path is 'Lambda > 함수 > gptKakaobot'.
- The title 'gptKakaobot' is displayed.
- 함수 개요** section:
 - 설명**: '-'
 - 마지막 수정**: '1분 전'
 - 함수 ARN**: 'arn:aws:lambda:us-east-2:53267 6954588:function:gptKakaobot'
 - 함수 URL**: '정보'
- 함수 디자인** tab is selected.
- 함수 콘솔** tab is available.
- Layers** section: 'Layers (0)'.
- Triggers** and **Targets** sections are present with '+ 추가' buttons.

챗GPT API 실습 - 카카오톡 챗봇

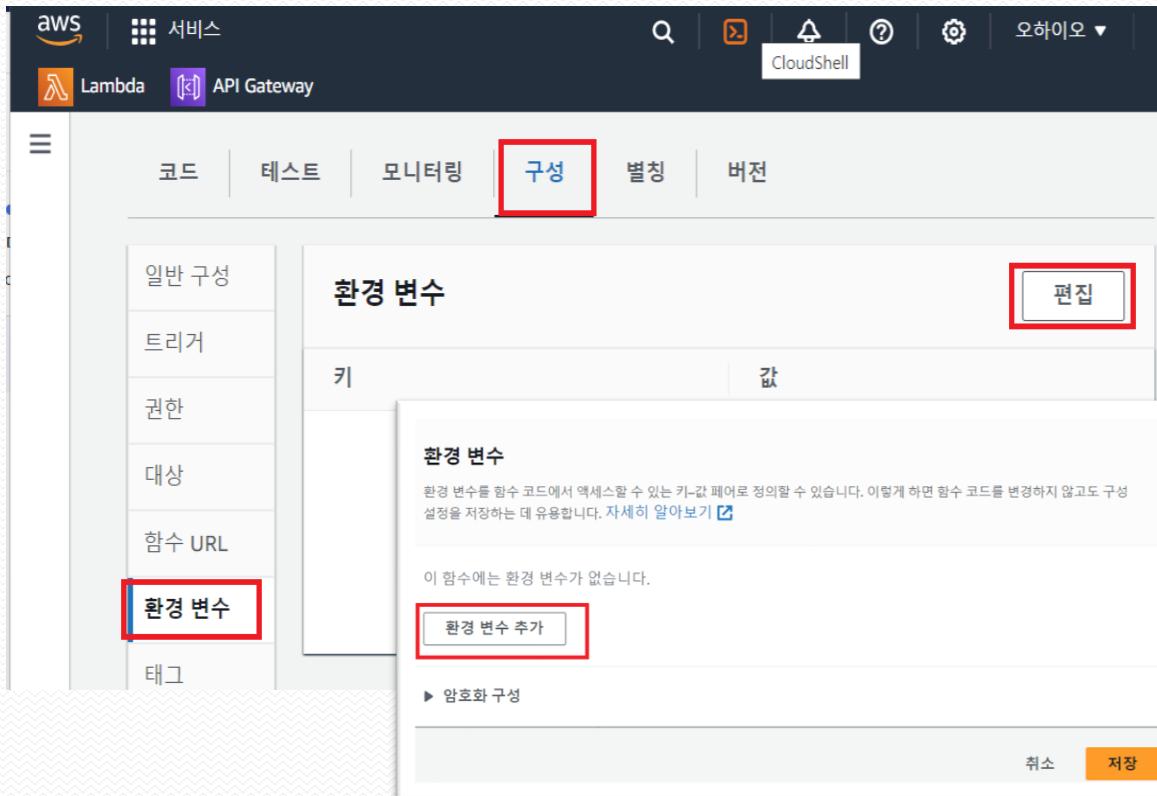
V. (실습)챗GPT API 활용

3. AWS람다 함수 생성하기



챗GPT API 실습 - 카카오톡 챗봇

4. 람다함수에 환경변수 생성(OpenAI API 키 입력하기)



챗GPT API 실습 - 카카오톡 챗봇

V. (실습)챗GPT API 활용

5. lambda_function 코드 작성하기

The image shows two side-by-side screenshots of the AWS Lambda function editor and its execution results.

Left Screenshot (Editor):

- The tab bar at the top has "코드" (Code) highlighted with a red box.
- The code editor window shows a Python file named "lambda_function.py". The file contains code for a Lambda function that handles KakaoTalk messages. It imports json, openai, threading, time, queue, and os. It sets an OpenAI API key from the environment variable OPENAI_API. It defines a lambda_handler function that reads a message from the event body, writes it to a log file, and then starts a thread to handle the response using the OpenAI API.
- The toolbar at the bottom has "Test" and "Deploy" buttons, both highlighted with red boxes.

Right Screenshot (Execution Results):

- The title bar says "코드 소스 정보".
- The toolbar at the top has "Test" and "Deploy" buttons, with "Deploy" highlighted with a red box.
- The main area shows the "Execution results" section. It displays the "Test Event Name" as "MyEventName".
- The "Response" section shows a JSON object with an errorMessage ("body"), an errorType ("KeyError"), a requestId ("afe865ef-0d75-4733-9465-168bdc8977e8"), and a stackTrace.
- The "Function Logs" section shows the full log output, including the start request, an error message about a missing 'body' parameter, the traceback, and the end request.

챗GPT API 실습 - 카카오톡 챗봇

5.1 lambda_function 코드 작성 : 기본 정보 설정 단계

```
1 ##### 기본 정보 설정 단계 #####
2 import json
3 import openai
4 import threading
5 import time
6 import queue as q
7 import os
```

챗GPT API 실습 - 카카오톡 챗봇

5.2 lambda_function 코드 작성 : 기능함수 구현단계 – 메시지 전송 및 사진 전송

```
103 ##### 기능 구현 단계 #####
104
105 # 메세지 전송
106 def textResponseFormat(bot_response):
107     response = {'version': '2.0', 'template': {
108         'outputs': [{"simpleText": {"text": bot_response}}]}, 'quickReplies': []}
109     return response
110
111 # 사진 전송
112 def imageResponseFormat(bot_response,prompt):
113     output_text = prompt+"내용에 관한 이미지입니다"
114     response = {'version': '2.0', 'template': {
115         'outputs': [{"simpleImage": {"imageUrl": bot_response,"altText":output_text}}]}, 'quickReplies': []}
116     return response
```

챗GPT API 실습 - 카카오톡 챗봇

5.3 lambda_function 코드 작성 : ChatGPT에게 질문/답변 받기, DALL.E2에게 질문/그림 URL받기

```
136 # ChatGPT에게 질문/답변 받기
137 def getTextFromGPT(messages):
138     messages_prompt = [{"role": "system", "content": 'You are a thoughtful assistant.\\
139                         Respond to all input in 25 words and answer in korea'}]
140     messages_prompt += [{"role": "user", "content": messages}]
141     response = client.chat.completions.create(model="gpt-3.5-turbo", messages=messages_prompt)
142     message = response.choices[0].message.content
143     return message
144
145 # DALLE에게 질문/그림 URL 받기
146 def getImageURLFromDALLE(messages):
147     response = client.images.generate(
148         model="dall-e-2",
149         prompt=messages,
150         size="512x512",
151         quality="standard",
152         n=1)
153     image_url = response.data[0].url
154     return image_url
```

챗GPT API 실습 - 카카오톡 챗봇

5.4 lambda_function 코드 작성 : 메인 함수 – lambda_handler함수로 람다가 호출되면 작동하는 핵심동작 함수 (1/3)

```
14 # 메인 함수
15 def lambda_handler(event, context):
16
17     run_flag = False
18     start_time = time.time()
19     # 카카오 정보 저장
20     kakaorequest = json.loads(event['body'])
21     # 응답 결과를 저장하기 위한 텍스트 파일 생성
22
23     filename = "/tmp/botlog.txt"
24     if not os.path.exists(filename):
25         with open(filename, "w") as f:
26             f.write("")
27     else:
28         print("File Exists")
```

챗GPT API 실습 - 카카오톡 챗봇

5.4 lambda_function 코드 작성 : 메인 함수 – lambda_handler함수로 람다가 호출되면 작동하는 핵심동작 함수 (2/3)

```
30     # 답변 생성 함수 실행
31     response_queue = q.Queue()
32     request_respond = threading.Thread(target=responseOpenAI,
33                                         args=(kakaorequest, response_queue,filename))
34     request_respond.start()
35
36     # 답변 생성 시간 체크
37     while (time.time() - start_time < 3.5):
38         if not response_queue.empty():
39             # 3.5초 안에 답변이 완성되면 바로 값 리턴
40             response = response_queue.get()
41             run_flag= True
42             break
43         # 안정적인 구동을 위한 딜레이 타임 설정
44         time.sleep(0.01)
```

챗GPT API 실습 - 카카오톡 챗봇

5.4 lambda_function 코드 작성 : 메인 함수 – lambda_handler함수로 람다가 호출되면 작동하는 핵심동작 함수 (3/3)

```
50     return{
51         'statusCode':200,
52         'body': json.dumps(response),
53         'headers': {
54             'Access-Control-Allow-Origin': '*',
55         }
56     }
```

챗GPT API 실습 - 카카오톡 챗봇

5.5 lambda_function 코드 작성 : responseOpenAI()함수 – 사용자의 채팅을 분석하여 ChatGPT에게 답변을 받거나 DALL.E2에게 그림을 받는 기능 수행 (1/3)

```
58 # 답변/사진 요청 및 응답 확인 함수
59 def responseOpenAI(request,response_queue,filename):
60     # 사용자다 버튼을 클릭하여 답변 완성 여부를 다시 봤을 시
61     if '생각 다 끝났나요?' in request["userRequest"]["utterance"]:
62         # 텍스트 파일 열기
63         with open(filename) as f:
64             last_update = f.read()
65         # 텍스트 파일 내 저장된 정보가 있을 경우
66         if len(last_update.split())>1:
67             kind = last_update.split()[0]
68             if kind == "img":
69                 bot_res, prompt = last_update.split()[1],last_update.split()[2]
70                 response_queue.put(imageResponseFormat(bot_res,prompt))
71             else:
72                 bot_res = last_update[4:]
73                 response_queue.put(textResponseFormat(bot_res))
74             dbReset(filename)
```

챗GPT API 실습 - 카카오톡 챗봇

5.5 lambda_function 코드 작성 : responseOpenAI()함수 – 사용자의 채팅을 분석하여 ChatGPT에게 답변을 받거나 DALL.E2에게 그림을 받는 기능 수행 (2/3)

```
76      # 이미지 생성을 요청한 경우
77      elif '/img' in request["userRequest"]["utterance"]:
78          dbReset(filename)
79          prompt = request["userRequest"]["utterance"].replace("/img", "")
80          bot_res = getImageURLFromDALLE(prompt)
81          response_queue.put(imageResponseFormat(bot_res,prompt))
82          save_log = "img" + " " + str(bot_res) + " " + str(prompt)
83          with open(filename, 'w') as f:
84              f.write(save_log)
```

챗GPT API 실습 - 카카오톡 챗봇

5.5 lambda_function 코드 작성 : responseOpenAI()함수 – 사용자의 채팅을 분석하여 ChatGPT에게 답변을 받거나 DALL.E2에게 그림을 받는 기능 수행 (3/3)

```
86     # ChatGPT 답변을 요청한 경우
87     elif '/ask' in request["userRequest"]["utterance"]:
88         dbReset(filename)
89         prompt = request["userRequest"]["utterance"].replace("/ask", "")
90         bot_res = getTextFromGPT(prompt)
91         response_queue.put(textResponseFormat(bot_res))
92
93         save_log = "ask" + " " + str(bot_res)
94         with open(filename, 'w') as f:
95             f.write(save_log)
96
97     # 아무 답변 요청이 없는 채팅일 경우
98     else:
99         # 기본 response 값
100        base_response = {'version': '2.0', 'template': {'outputs': [], 'quickReplies': []}}
101        response_queue.put(base_response)
102
```

챗GPT API 실습 - 카카오톡 챗봇

6. 계층을 생성하여 OpenAI 패키지를 레이어로 등록하기

The screenshot shows two side-by-side views of the AWS Lambda console.

Left View (AWS Lambda Service):

- Header: AWS Lambda, 서비스, Lambda, API Gateway
- Left sidebar: 대시보드, 애플리케이션, 함수, 추가 리소스 (highlighted), 코드 서명 구성, 계층 (highlighted), 복제본.
- Middle area: Lambda > 계층
- Table: 계층 (1)

이름	버전	호환 런타임	호환 아키텍처
openai310	1	python3.10	x86_64

Right View (Lambda Layer Creation):

- Header: Lambda, API Gateway
- Title: 계층 생성
- Form fields:
 - 이름: openai310
 - 설명 - 선택 사항: 설명
 - 선택 사항:
 - .zip 파일 업로드
 - Amazon S3에서 파일 업로드
 - 업로드 (highlighted)
- File browser: 열기 (highlighted)
 - 선택 사항: x86_64, arm64
 - 설명: 최대 15개의 런타임을 선택합니다.
 - 선택 사항: 실행 시간
 - 선택 사항: 라이선스
- File list:
 - 구성 새 폴더: ch03
 - 내 PC: ch04_urllib3_example2 (highlighted), python (highlighted)
- Buttons: 취소, 생성 (highlighted)

챗GPT API 실습 - 카카오톡 챗봇

6. 계층을 생성하여 OpenAI 패키지를 레이어로 등록하기

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Lambda' is selected. The main area shows a function named 'gptKakaobot'. Under the 'Layers' section, there is a button labeled 'Layers (0)'. Below this, there is a 'Add a layer' button. The entire 'Layers' section is highlighted with a red box.

The screenshot shows the 'Layer Add' dialog box. It has two main sections: '함수 런타임 설정' (Function Runtime Configuration) and '계층 선택' (Layer Selection). In the runtime configuration, 'Python 3.10' is selected. In the layer selection section, 'AWS 계층' (AWS Layer) is selected, indicated by a checked radio button. A red box highlights this selection. Below it, there is a description: 'AWS에서 제공하는 계층 목록에서 계층을 선택합니다.' (Select a layer from the AWS-provided layer list). The 'Version' dropdown is set to '1'. At the bottom right, there is a large orange '추가' (Add) button, which is also highlighted with a red box.

챗GPT API 실습 - 카카오톡 챗봇

7. 람다의 제한시간 늘리기

The screenshot shows the AWS Lambda function configuration interface. On the left, there's a sidebar with tabs: 코드 (Code), 테스트 (Test), 모니터링 (Monitoring), 구성 (Configuration), 별칭 (Alias), and 버전 (Version). The '구성' (Configuration) tab is selected and highlighted with a red box.

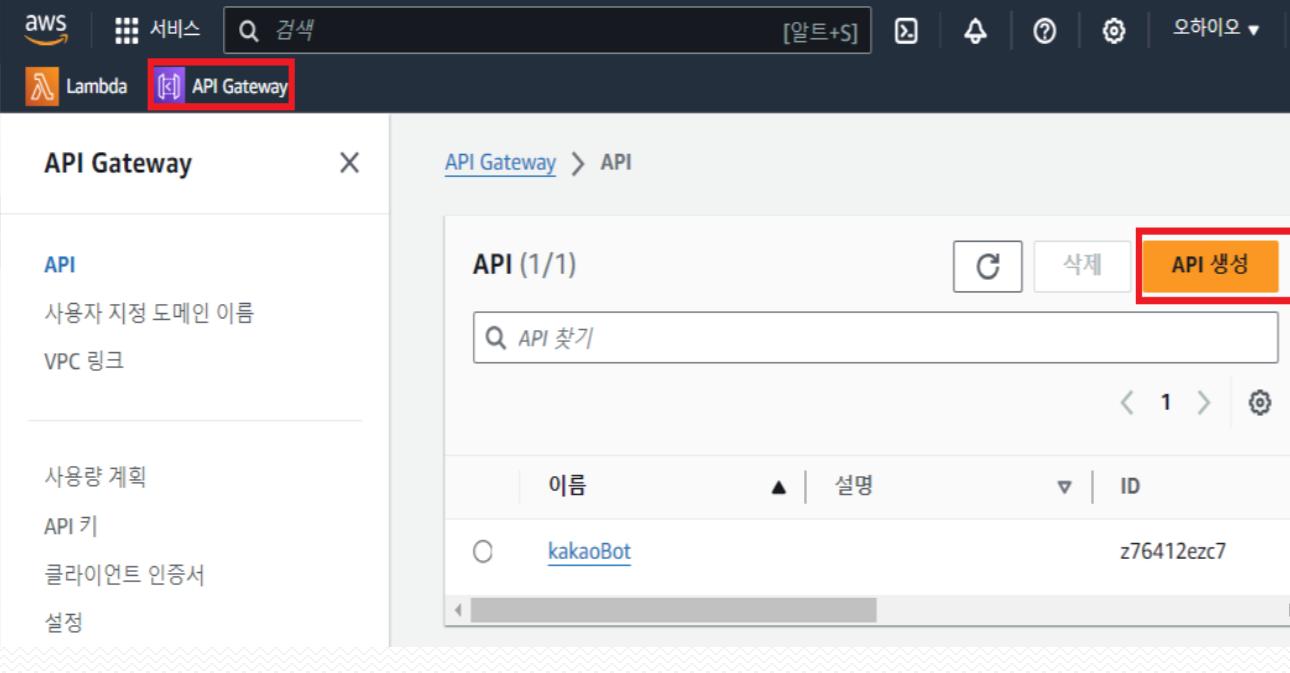
In the main area, under the 'General Configuration' tab, there's a 'General Configuration' section with a 'View details' link. It shows the following settings:

- 설명 (Description):** - (empty)
- 메모리 (Memory):** 128 MB
- 임시 스토리지 (Temporary Storage):** 512 MB
- 제한 시간 (Timeout):** 1 분 0 초 (highlighted with a red box)
- SnapStart (SnapStart):** None

To the right, under 'Basic Settings', there's a 'Memory' section with a 'Configure' button, a 'Temporary storage' section with a 'Configure' button, and a 'SnapStart' section with a dropdown menu set to 'None'. At the bottom, there's a 'Save' button highlighted with a red box.

챗GPT API 실습 - 카카오톡 챗봇

8. AWS API 게이트웨이 생성하기



The screenshot shows the AWS API Gateway console. On the left sidebar, under the 'API' section, there is a table with one row: '이름' (Name) is 'kakaoBot' and 'ID' is 'z76412ezc7'. At the top right of the main area, there is a yellow button labeled 'API 생성' (Create API), which is highlighted with a red box.

API 유형 선택

HTTP API

OIDC 및 OAuth2와 같은 기능과 기본 CORS 지원이 내장된, 지역 시간이 짧고 비용 효율적인 REST API를 구축합니다.

다음과 호환됩니다.
Lambda, HTTP 백엔드

구축

API 생성

통합 생성 및 구성

API가 통신할 백엔드 서비스를 지정하십시오. 이것을 통합이라고 합니다. Lambda 통합의 경우, API Gateway가 Lambda 함수를 호출하고 이 함수의 응답으로 응답합니다. HTTP 통합의 경우, API Gateway가 지정된 URL로 요청을 보내서 URL을 응답을 반환합니다.

통합 (1) 정보

Lambda

AWS 리전 Lambda 함수 Version Learn
us-e... arn:aws:lambda:us-east-2:532676954! more.
2.0

통합 추가

API 이름
HTTP API에는 이름이 있어야 합니다. 이 이름은 표시용이며, 고유할 필요는 없습니다. API의 ID(나중에 생성됨)를 사용하여 이 API를 프로그래밍 방식으로 참조합니다.

kakaobot2

다음

챗GPT API 실습 - 카카오톡 챗봇

8. AWS API 게이트웨이 생성하기

경로 구성 - 선택 사항

경로 구성 정보

API Gateway는 경로를 사용하여 API 사용자에게 통합을 표시합니다. HTTP API에 대한 경로는 HTTP 메서드와 리소스 경로 (예: GET /pets)의 두 부분으로 구성됩니다. 통합에 대한 특정 HTTP 메서드(GET, POST, PUT, PATCH, HEAD, OPTIONS 및 DELETE)를 정의하거나 ANY 메서드를 사용하여 지정된 리소스에서 정의하지 않은 모든 메서드를 일치시킬 수 있습니다.

메서드	리소스 경로	통합 대상
POST	/kakaoBot	kakaoBot

경로 추가

취소 검토 및 생성 이전 다음

스테이지 정의 - 선택 사항

스테이지 구성 정보

스테이지는 API를 배포할 수 있는 독립적으로 구성 가능한 환경입니다. 스테이지에 자동 배포가 구성된 경우를 제외하고, API 구성 변경을 사용하려면 스테이지에 API를 배포해야 합니다. 기본적으로 큰 술을 통해 생성된 모든 HTTP API에는 \$default라는 이름의 기본 스테이지가 있습니다. API에서 변경한 모든 내용은 해당 스테이지로 자동 배포됩니다. '개발' 또는 '프로덕션'과 같은 환경을 나타내는 스테이지를 추가할 수 있습니다.

스테이지 이름	자동 배포
\$default	<input checked="" type="radio"/>
스테이지 추가	제거

취소 이전 다음

스테이지

스테이지

- \$default (Auto-deploy: enabled)

취소 이전 생성

챗GPT API 실습 - 카카오톡 챗봇

8. AWS API 게이트웨이 생성하기

The screenshot shows the AWS Lambda function configuration interface for a function named 'kakaoBot'. The 'API Gateway' tab is selected under the 'Triggers' section. A red box highlights the 'API endpoint' field, which contains the URL: <https://z76412ezc7.execute-api.us-east-2.amazonaws.com/kakaoBot>.

Function Overview:

- 함수 개요**: 정보
- Application Composer로 내보내기**, **다운로드**
- 라이언스**: kakaoBot
- Layers**: (1)
- API Gateway** (highlighted with a red box)
- + 대상 추가**
- + 트리거 추가**

Configuration Tabs:

- 코드
- 테스트
- 모니터링
- 구성** (selected)
- 별칭
- 버전

Trigger Configuration:

- 트리거 (1) 정보**
- 트리거 찾기**
- 트리거**
- API Gateway: kakaoBot**
- API endpoint: https://z76412ezc7.execute-api.us-east-2.amazonaws.com/kakaoBot** (highlighted with a red box)

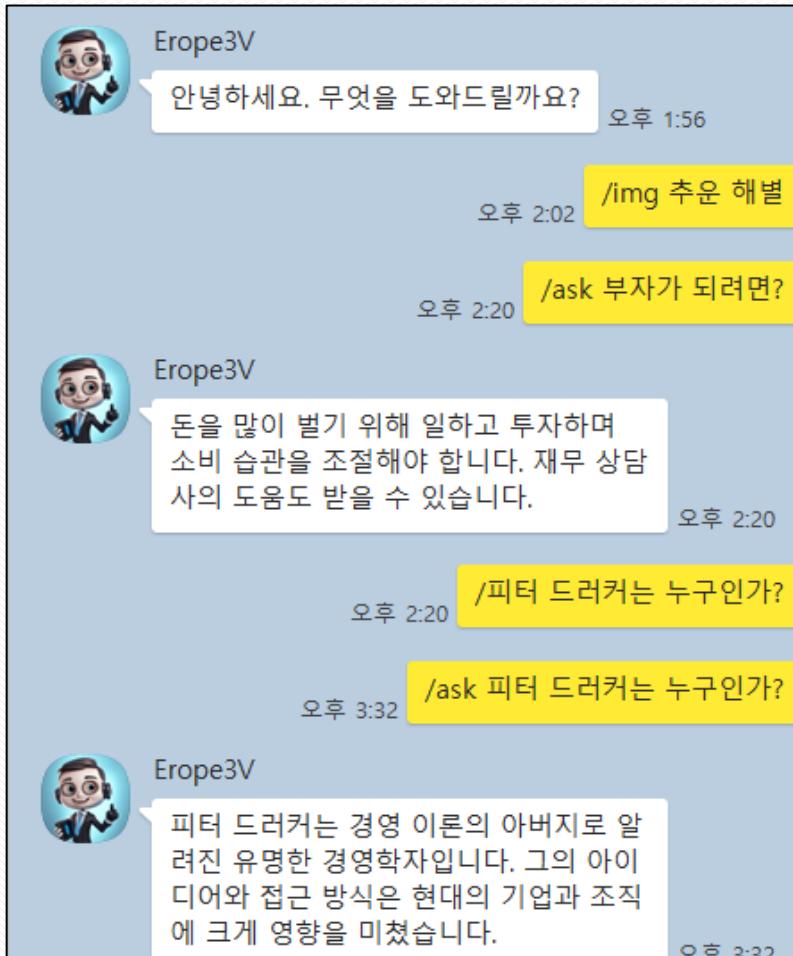
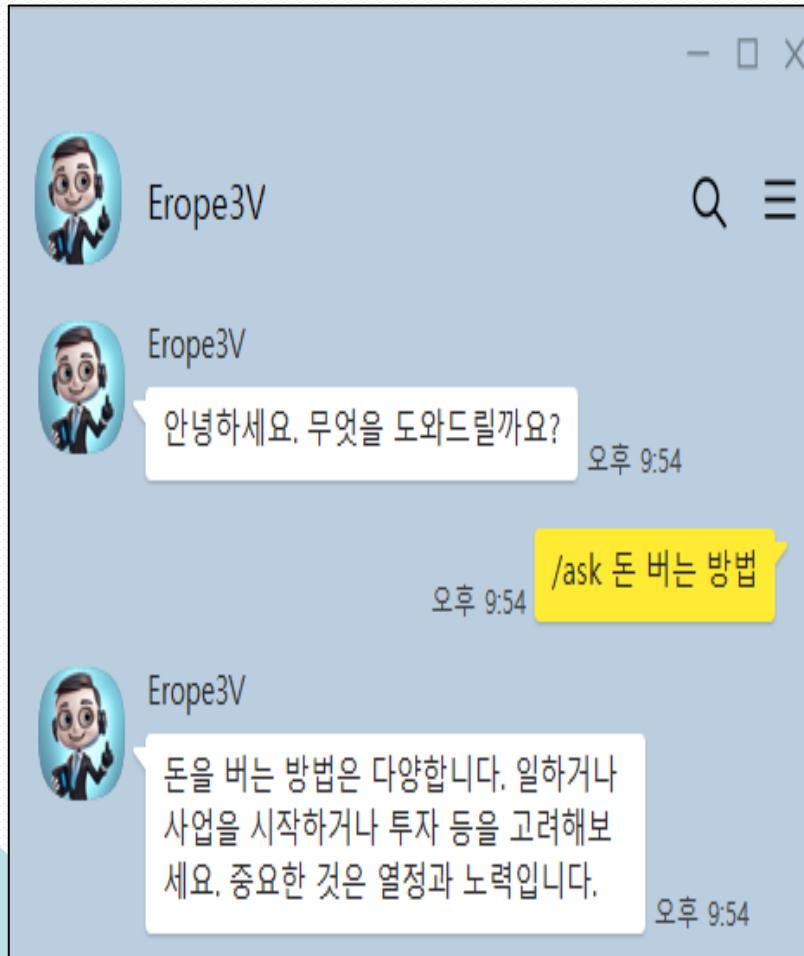
챗GPT API 실습 - 카카오톡 챗봇

9. AWS API 게이트웨이의 주소와 카카오톡 서버 연결하기

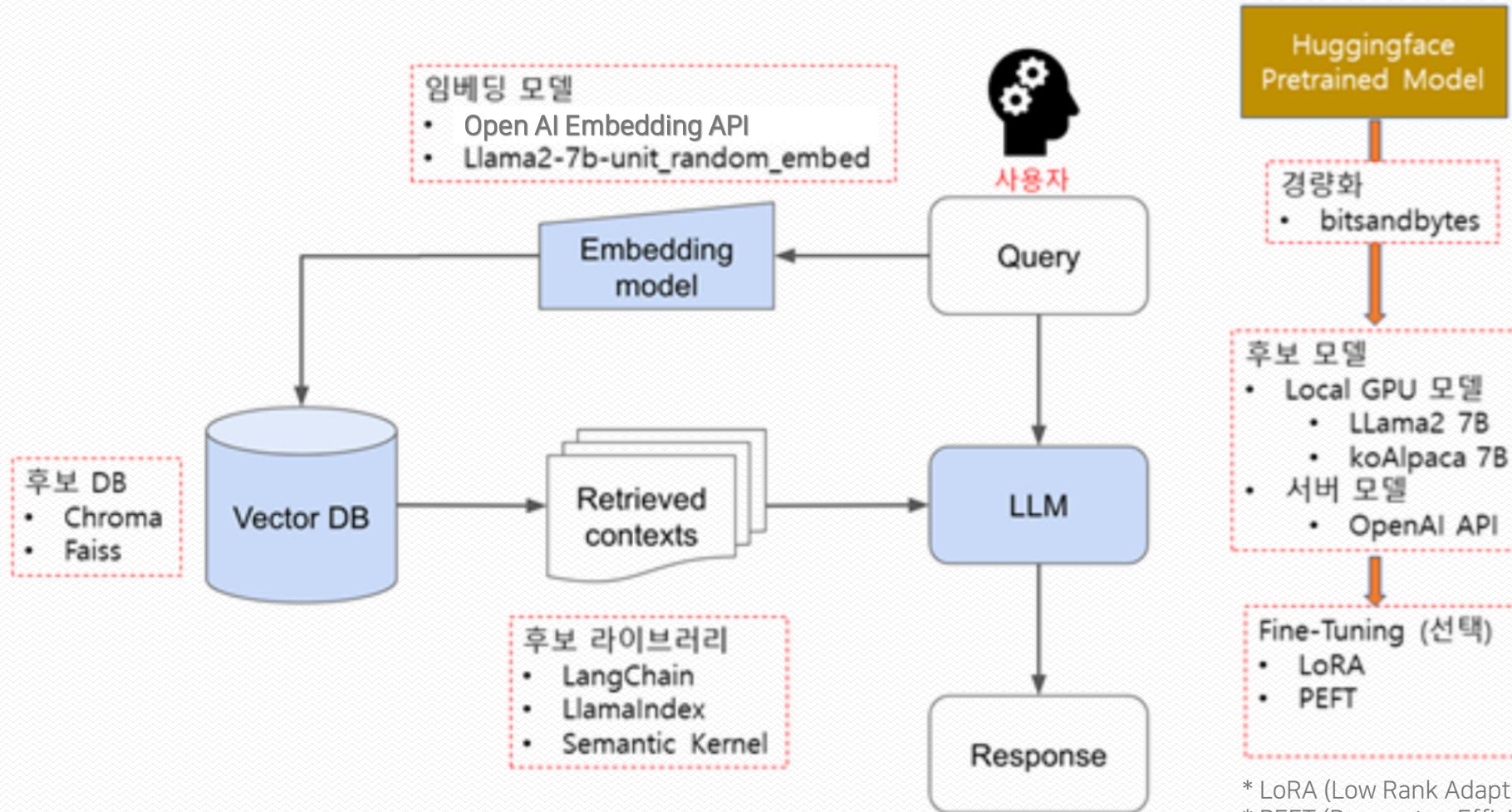
The screenshot shows the AWS Lambda function configuration for the 'kakaoBot' function. The 'API Gateway' trigger is selected and highlighted with a red box. The 'API endpoint' field contains the URL <https://z76412ezc7.execute-api.us-east-2.amazonaws.com/kakaoBot>, which is also highlighted with a red box.

The screenshot shows the Kakao Business Chatbot Management Platform. A skill named 'kakaoBot' is being registered. The 'Skill Name' field contains 'kakaoBot'. The 'Skill Type' dropdown is set to '전체'. The 'Skill Description' section shows the ARN: arn:aws:lambda:us-east-2:532676954588:function:kakaoBot. The 'Skill Status' is set to '미사용' (Not Enabled). The 'Skill Version' dropdown is set to '1'. The 'Skill Author' is listed as 'kubbugy@gmail.com'. The 'Skill Update Date' is '2024.01.28 15:04'. The 'Skill Registration' button is highlighted with a red box.

챗GPT API 실습 - 카카오톡 챗봇



랭체인(LangChain)



* LoRA (Low Rank Adaptation)

* PEFT (Parameter-Efficient Fine-Tuning)

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

‘신혼부부 임차보증금 지원 정책이 궁금해’



벡터 : [0.54, 0.84, 0.28, 0.59]

유사도 계산

유사도 Top -3

챗봇 답변

11번 문서
21번 문서
56번 문서

유사도가 가장 높은 문서 3개
신혼부부 임차보증금 지원은…
희망두배 청년통장에 대해서…
역세권청년주택 주거비 지원…

57개 문서의 임베딩 벡터
[0.27, 0.58, 0.69, 0.34]
[0.27, 0.58, 0.69, 0.34]

[0.27, 0.58, 0.69, 0.34]
[0.27, 0.58, 0.69, 0.34]

1번 문서
2번 문서
중략
56번 문서
57번 문서



서울시에서는 신혼부부를 위해
신혼부부 임차보증금 지원 정책을
시행하고 있으며, 지원
대상은.....

랭체인실습 - 우리 회사 챗봇

1. 스트림릿 및 OpenAI 설치

```
C:\chatbot> pip install streamlit
```

```
C:\chatbot> pip install streamlit-chat
```

```
C:\chatbot> pip install openai==0.28.1 (이미 신규버전 설치되어 있으면 pip uninstall openai)
```

```
C:\chatbot> pip install matplotlib scipy plotly scikit-learn
```

2. 스트림릿 파일 실행 : C:\chatbot> streamlit run ch05_chatbot_example_student.py



랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

- ChatGPT를 이용한 데이터 전처리 – 서울시 청년 동땅 정보통 (<https://youth.seoul.go.kr>)

(#참조) ChatGPT 웹사이트에 복사한 해당 내용을 평문으로 재작성하도록 요청하여 간단하게 데이터 전처리를 수행함.

(프롬프트 예시) – 아래 내용은 정리되지 않은 표 형태의 데이터입니다. 당신은 아래 내용을 평문으로 작성해야 합니다.

〈내용〉

-----청년정책 본문 내용 중략 -----

〈내용끝〉

- 위 내용 전부를 읽기 쉽도록 평문으로 재작성해주세요. 임의로 정리하거나 소제목을 붙이지 마세요. 끊어서 작성하지 말고 반드시 하나의 평문이어야 합니다.
- 웹사이트 주소나 URL 주소가 있다면 해당 주소를 단축하거나 애매모호하게 줄이지 마세요. URL주소는 반드시 원문 그대로 작성해야 합니다.

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

2. 정제된 데이터 (https://github.com/EropeConsulting/INTEL_LLM/tree/main/chatbot/data)

The screenshot shows the Seoul City Policy website interface. On the left, there's a sidebar with various menu items like 'Chungnam Policy', 'Public Participation', 'Chungnam Policy Analysis', 'Seoul City Policy' (which is highlighted with a red box), 'Central Chungnam Policy', 'Policy Database', 'Chungnam Policy Information', 'Seoul Youth Policy', 'Chungnam Research', 'Chungnam Policy Analysis', and 'Chungnam Policy and Issues'. The main content area has a search bar with placeholder text '검색어를 입력해주세요.' and a search button. Below the search bar are several filters: '정책유형' (Policy Type), '취업상태' (Employment Status), '모집현황' (Recruitment Status), '연령대' (Age Group), and '담당기관' (Responsible Agency). The search results are displayed in a grid format. A red box highlights the first result, '서울시 정책 (총 64건)', which has three sub-categories: '교육' (Education), '문화' (Culture), and '복지' (Welfare). Each category has a brief description and a link to the full policy details.

This screenshot shows a detailed view of a policy document. At the top, it says '아래 내용은 정리되지 않은 표 형태의 데이터입니다. 당신은 아래 내용을 평문으로 작성해야 합니다.' (The following content is unstructured table data. You must rewrite it in plain text). The document contains the following information:
- <내용>
- 청년예술지원
- 사업개요
- 정책 유형 복지, 문화 주관 기관 서울시청 문화예술과
- 정책 소개 활동 경력이 부족한 청년예술인 대상 첫 작품 발표 지원
- 지원 내용 - 창작지원금: 최대 1,000만원 이내
- 지원 대상: 전문가 멘토링, 네트워킹 프로그램 제공
- 사업 운영 기간: 2023.1.1. ~ 2023.12.31. 사업 신청 기간 참고: 2023.1.1. ~ 2023.12.31.
- 신청 기간의 경우: 2022.10.4. ~ 2022.10.28. (신청 완료)
- 지원 규모: 40명 (단체) 관련 사이트
- 신청 자격
- 연령: 19세 ~ 39세
- 학력: 제한 없음 전공 요건 제한 없음
- 취업 상태: 제한 없음 특화 분야 요건 제한 없음
- 추가 단서 사항: -
- 참여 제한 대상: -
- 신청 방법
- 신청 절차: 서울문화예술지원시스템(SCAS)을 통한 온라인 신청
- 신청 시기 및 발표: 2023.1.1. ~ 2023.12.31.
- 제출 서류: 지원 신청서
- 신청 사이트: <https://www.scas.kr/scas/bizinfo/detail/198>
- 기타: 기타 사항 문의: 02-362-9745
- 운영 기관: 서울문화재단
- 참고 사이트: I <https://www.scas.kr/scas/bizinfo/detail/198>
- 참고 사이트 II: -
- <내용 끝>
At the bottom, there are two numbered instructions:
1. 위 내용 전부를 읽기 쉽도록 평문으로 재작성 하시오.
2. 웹 사이트 주소나 URL 주소가 있다면 원문 그대로 작성해야 합니다.

This screenshot shows a text file with the following content:
1.txt

정책 이름: 청년예술지원 서울시청 문화예술과에서는 청년 예술인들을 위한 프로그램을 진행합니다.
이 프로그램은 활동 경력이 부족한 청년 예술인들을 대상으로 하며, 최대 1,000만원의 창작 지원금과 전문가 멘토링 및 네트워킹 프로그램을 제공합니다. 이 프로그램은 2023년 1월 1일부터 2023년 12월 31일까지 운영됩니다. 신청자는 19세에서 39세까지의 나이 조건을 충족하며, 서울에서 첫 작품 발표(오프라인)를 준비 중인 예술인(단체)어야 합니다. 학력, 전공, 취업 상태, 특화 분야에 대한 요건은 없습니다. 신청은 온라인으로 서울문화예술지원시스템(SCAS)을 통해 진행되며, 신청 기간은 2023년 10월 4일부터 10월 28일까지입니다. 심사 및 발표는 2023년 1월 경에 이루어집니다. 프로그램에 대한 자세한 내용은 <https://www.scas.kr/scas/bizinfo/detail/198>에서 확인할 수 있으며, 추가 문의는 02-362-9745로 하실 수 있습니다.

2.txt

정책 이름: 서울청년문화패스 사회에 처음 발을 내딛는 19세 청년들을 위해 연극, 뮤지컬, 무용, 클래식, 전통예술 등 다양한 문화예술 공연 관람 기회를 제공하고 문화예술 분야를 활성화하기 위해 서울시청 문화정책과에서 문화이용권을 지원하는 사업을 진행합니다. 이 프로그램은 2023년 1월 1일부터 2023년 12월 31일까지 운영되며, 신청 기간은 2023년 4월 19일부터 5월 31일까지입니다. 참여 자격은 만 19세 이하(2004년생 기준), 서울에 거주하고 중위소득 150% 이하인 자입니다. 총 28,000명이 지원을 받을 수 있으며, 신청은 청년동반정보통(<https://youth.seoul.go.kr>)에서 온라인으로 진행됩니다. 프로그램에 대한 자세한 내용은 <https://mediahub.seoul.go.kr/archives/2006883>에서 확인할 수 있습니다. 이외의 제출 서류는 필요하지 않으며, 외국인 및 재외동포의 경우 국내 거소 사실을 증명하는 서류를 첨부해야 합니다. 이 프로그램은 (재)서울문화재단에서 운영됩니다.

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.1 패키지 불러오기

```
1 import openai
2 import streamlit as st
3 import os, tenacity
4 import pandas as pd
5 import numpy as np
6 from numpy import dot
7 from numpy.linalg import norm
8 import ast
9 from openai.embeddings_utils import get_embedding
10 from streamlit_chat import message
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.2 embedding.csv 생성 및 데이터프레임 로드 (1/2)

```
20 # if : embedding.csv 파일이 존재하면, 파일을 읽어서 데이터프레임 df를 로드한다.
21 if os.path.isfile(file_path):
22     print(f"{file_path} 파일이 존재합니다.")
23     df = pd.read_csv(file_path)
24     df['embedding'] = df['embedding'].apply(ast.literal_eval)
25
26 # 그렇지 않다면 text열과 embedding열이 존재하는 df를 신규 생성해야 한다.
27 else:
28     # 57개의 서울 청년 정책 txt 파일명을 txt_files에 저장한다.
29     folder_path = './data' # data 폴더 경로
30     txt_files = [file for file in os.listdir(folder_path) if file.endswith('.txt')] # txt 파일 목록
31
32     data = []
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.2 embedding.csv 생성 및 데이터프레임 로드 (2/2)

```
34      # txt_files로부터 57개의 청년 정책 데이터를 로드하여 df를 신규 생성한다.
35      for file in txt_files:
36          txt_file_path = os.path.join(folder_path, file)
37          with open(txt_file_path, 'r', encoding='utf-8') as f:
38              text = f.read() # 파일 내용 읽기
39              data.append(text)
40
41      df = pd.DataFrame(data, columns=['text'])
42
43      # 데이터프레임의 text 열에 대해서 embedding을 추출
44      df['embedding'] = df.apply(lambda row: get_embedding(
45          row.text,
46          engine="text-embedding-ada-002"
47      ), axis=1)
48
49      # 추후 사용을 위해 df를 embedding.csv 파일로 저장
50      # 이렇게 저장해두면 추후 실행했을 때 df를 만드는 과정을 생략할 수 있다.
51      df.to_csv(file_path, index=False, encoding='utf-8-sig')
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.3 코사인 유사도 검색시스템 코드 (1/2)

```
53 # 주어진 질의로부터 유사한 문서를 반환하는 검색 시스템
54 # 함수 return_answer_candidate 내부에서 유사도 계산을 위해 cos_sim을 호출한다.
55 def cos_sim(A, B):
56     return dot(A, B)/(norm(A)*norm(B))
57
58 def return_answer_candidate(df, query):
59     query_embedding = get_embedding(
60         query,
61         engine="text-embedding-ada-002"
62     )
63     df["similarity"] = df.embedding.apply(lambda x: cos_sim(np.array(x), np.array(query_embedding)))
64     top_three_doc = df.sort_values("similarity", ascending=False).head(3)
65     return top_three_doc
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.3 코사인 유사도 검색시스템 코드 (2/2)

```
67 # 챗봇의 답변을 만들기 위해 사용될 프롬프트를 만드는 함수
68 def create_prompt(df, query):
69     result = return_answer_candidate(df, query)
70     system_role = f"""You are an artificial intelligence language model named "정채기" that specializes in summarizing \
71     and answering documents about Seoul's youth policy, developed by developers 사용자1 and 사용자2.
72     You need to take a given document and return a very detailed summary of the document in the query language.
73     Here are the document:
74         doc 1 :"""+ str(result.iloc[0]['text']) + """
75         doc 2 :"""+ str(result.iloc[1]['text']) + """
76         doc 3 :"""+ str(result.iloc[2]['text']) + """
77     You must return in Korean. Return a accurate answer based on the document.
78     """
79     user_content = f"""User question: "{str(query)}". """
80
81     messages = [
82         {"role": "system", "content": system_role},
83         {"role": "user", "content": user_content}
84     ]
85
86     return messages
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.4 챗봇의 답변 생성 (1/2)

```
87 # 위의 create_prompt 함수가 생성한 프롬프트로부터 챗봇의 답변을 만드는 함수
88 def generate_response(messages):
89     result = openai.ChatCompletion.create(
90         model="gpt-3.5-turbo",
91         messages=messages,
92         temperature=0.4,
93         max_tokens=500)
94     return result['choices'][0]['message']['content']
95
96 # 챗봇의 마스코트 이미지 삽입
97 st.image('images/ask_me_chatbot.png')
98
99 # 화면에 보여주기 위해 챗봇의 답변을 저장할 공간 할당
100 if 'generated' not in st.session_state:
101     st.session_state['generated'] = []
102
103 # 화면에 보여주기 위해 사용자의 답변을 저장할 공간 할당
104 if 'past' not in st.session_state:
105     st.session_state['past'] = []
```

랭체인실습 - 우리 회사 챗봇

VI. (실습) 랭체인

3.4 챗봇의 답변 생성 (2/2)

```
107 # 사용자의 입력이 들어오면 user_input에 저장하고 Send 버튼을 클릭하면 submitted의 값이 True로 변환
108 with st.form('form', clear_on_submit=True):
109     user_input = st.text_input('정책을 물어보세요!', '', key='input')
110     submitted = st.form_submit_button('Send')
111
112 # submitted의 값이 True면 챗봇이 답변을 하기 시작
113 if submitted and user_input:
114     # 프롬프트 생성
115     prompt = create_prompt(df, user_input)
116     # 생성 한 프롬프트를 기반으로 챗봇의 답변을 생성
117     chatbot_response = generate_response(prompt)
118     # 화면에 보여주기 위해 사용자의 질문과 챗봇의 답변을 각각 저장
119     st.session_state['past'].append(user_input)
120     st.session_state["generated"].append(chatbot_response)
121
122 # 사용자의 질문과 챗봇의 답변을 순차적으로 화면에 출력
123 if st.session_state['generated']:
124     for i in reversed(range(len(st.session_state['generated']))):
125         message(st.session_state['past'][i], is_user=True, key=str(i) + '_user')
126         message(st.session_state["generated"][i], key=str(i))
```

오픈소스 AI 챗봇 '허깅챗'

(<https://huggingface.co/chat/>)

The screenshot illustrates the process of creating and configuring an AI assistant on the HuggingChat platform.

Left Panel (Dashboard):

- HuggingChat v0.7.0**: The main header.
- Making the community's best AI chat models available to everyone.**: A tagline.
- Current Model**: **mistralai/Mixtral-8x7B-Instruct-v0.1**.
- Settings** section:
 - Models** list: Shows various AI models, with **mistralai/Mixtral-8x7B-Instruct...** marked as **Active**. This list is highlighted with a red box.
 - Assistants** section: Shows an existing assistant named **Erope3V** and a link to **Browse Assistants**.
- Navigation** menu on the left:
 - EropeConsulting
 - Theme
 - Assistants
 - Settings
 - About & Privacy

Center Panel (Assistant Configuration):

- Erope3V**: Assistant profile picture and name.
- AI management strategy consultant, Peter Drucker**: Description.
- Created by EropeConsulting**: Creator information.
- Start chatting**: A large button highlighted with a red box.
- Direct URL**: Shareable link: <https://hf.co/chat/assistant/65bf3138a17b8fd4bd1c9caa>. A "Copy" button is next to it, also highlighted with a red box.
- System Instructions**: Placeholder text: "AI management strategy consultant, Peter Drucker".

Right Panel (Chat Interface):

- Erope3V**: Assistant profile picture.
- Assistant**: Assistant name.
- Erope3V**: Assistant description.
- Created by EropeConsulting**: Creator information.
- Erope3V : V-Shaped Recovery & Victory Start 새로운 도전과 성공 스토리를 위한 혁신과 기여 (To len...**: Chat message preview.
- Ask anything**: Input field for user questions.
- Model: meta-llama/Llama-2-70b-chat-hf - Generated content may be inaccurate or false.



THANK YOU