

나의 AI편친

자연어 처리 모델을 활용한 개인화 AI휴먼

Team 언어유희

Content

0 | 팀원 소개

1 | 프로젝트 진행 상황 리뷰

2 | 테크니컬 리뷰

3 | 진행 예정 프로세스 소개

4 | 시연 예시 / Q & A

5 | 진행 예상 방향

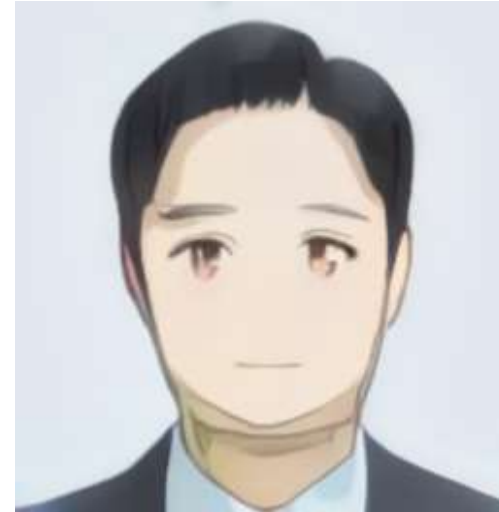
팀원 소개



팀장
장세종



이소담

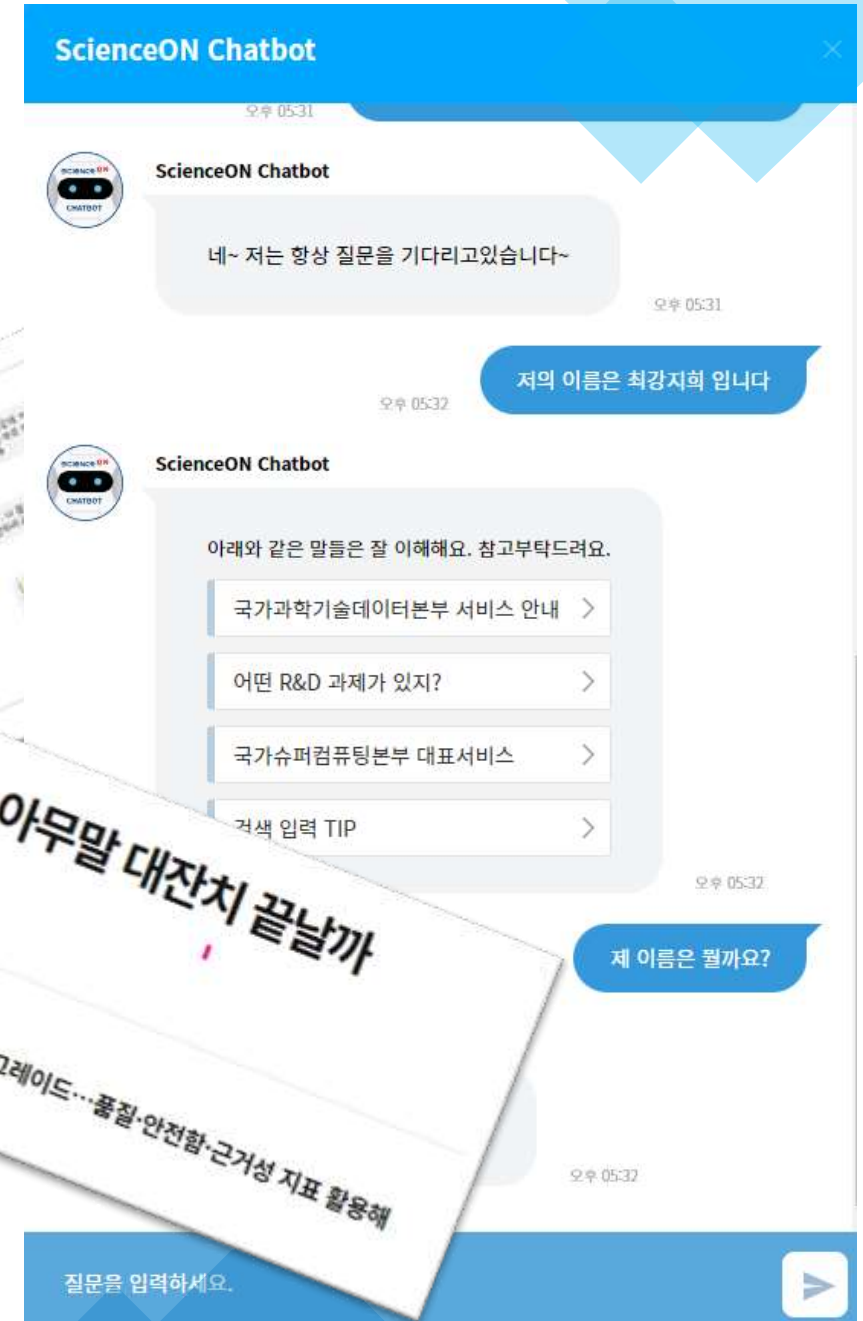
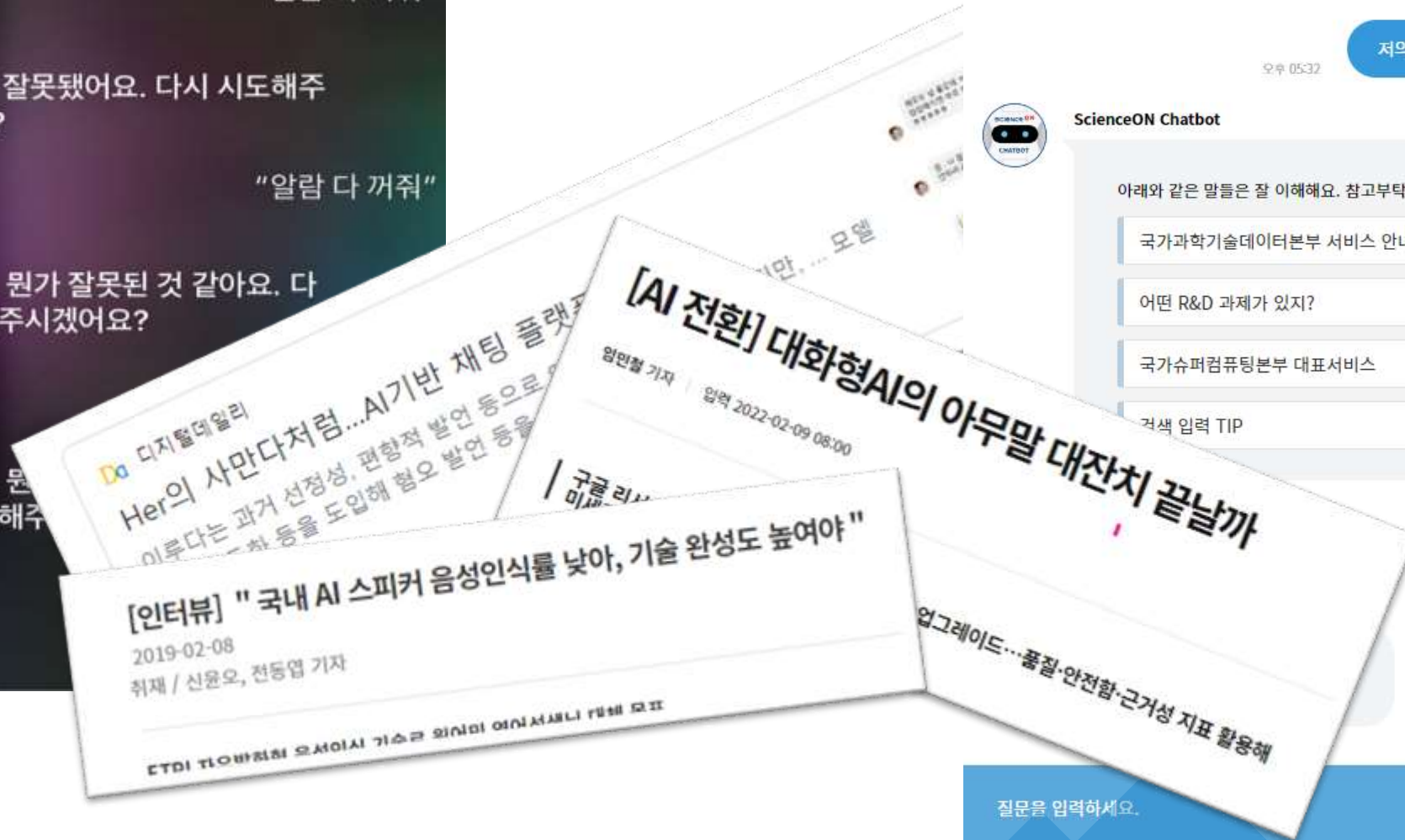
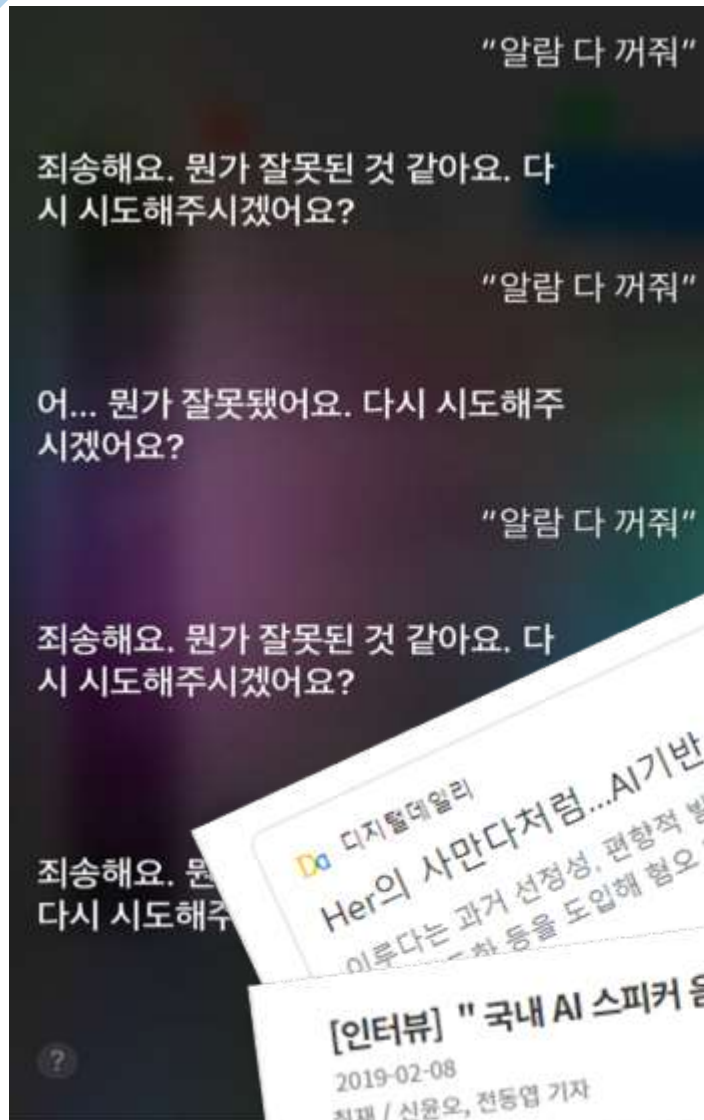


이 한



남궁지희

AI 챗봇 한계



나보다 나를 더 잘 아는

당신을 이해하고
당신에게 귀 기울이는 존재



“안녕 🖐️”

난 너의 AI핀친
유희지희야”

프로젝트 진행 상황



데이터 수집



..	
폴더 웰니스 대화 스크립트 데이터셋	Add files via upload
폴더 한국어 감정 정보가 포함된 연속적 대화 데이터셋	Add files via upload
폴더 한국어 대화 데이터셋	Add files via upload
파일 ChatBotData_지희.csv	03/20
파일 dataset1.xlsx	3/22일까지 합친 내용
파일 kakaotalk.csv	800
파일 트위터_대화시나리오DB_2000Set.xlsx	Add files via upload
파일 한국어_연속적_대화_데이터셋.xlsx	Add files via upload

```
In [10]: data.iloc[:50]
```

```
Out[10]:
```

	dialog #	발화	감정
0	5	아 진짜! 사무실에서 피지 말라니깐! 간접흡연이 얼마나 안좋은데!	분노
1	NaN	그럼 직접흡연하는 난 얼마나 안좋은지? 안그래? 보면 꼭 지 생각만 하고	혐오
2	NaN	친구도 담배 피지?	중립
3	NaN	친구? 누구?	중립
4	NaN	몰라니 친구래	중립
5	NaN	내 친구? 친구 누구?	중립
6	NaN	그걸 내가 어떻게 알아!	분노
7	5	그래서 무슨 일 해?	중립
8	NaN	그냥 엄마나 조금	중립
9	NaN	엄마나? 엄마나 뭐?	중립
10	NaN	회사에서 코딩도 하고 있어 그런 거	중립

```
In [13]: df=pd.DataFrame(columns=['Q','A'])
```

```
In [14]: df
```

```
Out[14]:
```

	Q	A
--	---	---

```
In [17]: df.loc[0]=[['아 진짜! 사무실에서 피지 말라니깐! 간접흡연이 얼마나 안좋은데!',  
'그럼 직접흡연하는 난 얼마나 안좋은지? 안그래? 보면 꼭 지 생각만 하고.',  
'친구도 담배 피지?'],  
'친구? 누구?']
```

```
In [18]: df
```

```
Out[18]:
```

	Q	A
0	[아 진짜! 사무실에서 피지 말라니깐! 간접흡연이 얼마나 안좋은데! 그럼 직접흡연... 친구? 누구?	

데이터 가공 (감정분류)

7개 감정분류 원본 데이터

행 번호	중립	슬픔	공포	혐오	분노	놀람
170						

29	S					만 내켜 혐오
30	NaN					왜? 중립
31	NaN					뭔가 알리는 기분이야. 불감해 공포
32	NaN					뭐가? 중립
33	NaN					생각해봐, 친구랍시고 갑자기 불쑥 나타나서는 안아상해? 공포
34	NaN					뭐가 이상해? 우정만 돈는구만 뭐 중립
35	NaN					우정 좋아하네. 싸가지 못했어? 지가 뭐라고 혐오
36	NaN					절친이라며 중립
37	NaN					나이에 갑자기 연락하는 것도 이상하고 혐오
38	NaN					복 받은 넘인가 보지 뭐 중립
39	NaN					너 오늘 유난히 대답 잘 한다? 라임 맞춰가면서 혐오
40	NaN					그니까. 쓸데없는 걱정 말고 도전해봐 중립
41	S					집 짓는 이유가 그렇게 중 해? 왜, 내가 사기라도 칠까봐? 혐오
42	NaN					또. 함악하게. 그런 게 아니라 왜 집을 짓는지 알아야 어떤 집이 될 한지를 알지. 중립
43	NaN					아하~ 그러니까 날 잘 알고 싶다 이거지? 내가 궁금하시다? 뭐가 그렇게 궁금할까?.. 중립

긍정, 부정, 중립 라벨링 작업

	id	document	label
404	4221289	NaN	0
412	9509970	NaN	1
470	10147571	NaN	1
584	7117896	NaN	0
593	6478189	NaN	0

화자(Q) / 청자(A) 대화쌍 구성

```
import pandas as pd
data=pd.read_csv('data.csv',header=None)
df=pd.DataFrame(columns=['Q','A'])

__speaker='__친구'
this_text=''
box=[]

for a in data[0]:
    if '[O I T E]' in a:
        this_speaker='__친구'
    else:
        this_speaker='__지회'
        w=a[::-1].find(' ')
        text=a[-w+1:]

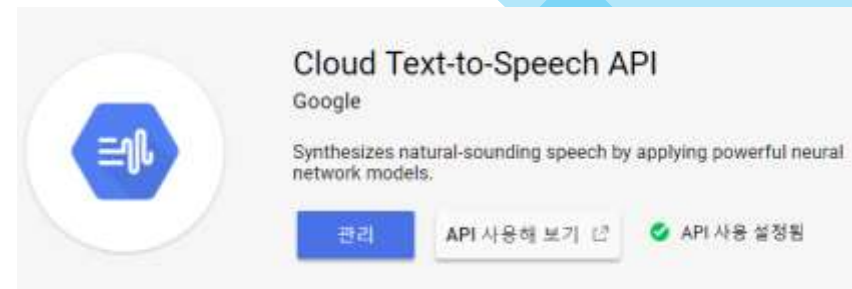
    # print(text)
    if this_speaker==__speaker:
        this_text+= ' '+text
    else:
        box.append([__speaker,this_text])
        if __speaker=='__친구':
            __speaker='__지회'
        else:
            __speaker='__친구'
        this_text=text

box.append([this_speaker,this_text])

q=[]
a=[]
for idx,i in enumerate(box):
    if i[0]=='__친구':
        q.append(i[1])
    else:
        a.append(i[1])

df['Q']=q
df['A']=a
df
```

STT - TTS



API 및 서비스

대시보드

API 및 서비스 사용 설정

사용 설정된 API 및 서비스

일부 API 및 서비스는 자동으로 사용 설정됩니다.

지난 1시간 동안의 활동

트래픽	요류
이 기간에는 트래픽이 없습니다.	이 기간에는 요류가 없!

API

요류

요류 비율

Cloud Translation API

Google Drive API

YouTube Data API v3

프로젝트 생

Google APIs auto post fb with google sheet

API 및 서비스 Google Sheets API

사용자 인증 정보 + 사용자 인증 정보 만들기

이 API와 호환되는 사용자 인증 정보

모든 사용자 인증 정보를 보거나 새 사용자 인증 정보를 만들려면 [인증 정보](#)로 이동하세요.

```
{
  "type": "service_account",
  "project id": "sd-invest",
}
```

```
{  
  "type": "service_account",  
  "project_id": "sd-invest",  
  "private_key_id": "68fe5544da05b88a789885f28c6ec2730397287d",  
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIIEvgIBADANBgkqhkiG9w0BAQEFAASCAmEgggQKAEEgaAoCCgGCAGQGAQQBgjcKAgEAAQEA\n-----",  
  "client_email": "firebase-adminsdk-xtw7a@sd-invest.iam.gserviceaccount.com",  
  "client_id": "103601898378166573587",  
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",  
  "token_uri": "https://oauth2.googleapis.com/token",  
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",  
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-xtw7a@sd-invest.iam.gserviceaccount.com"\br/>}
```

구글 API 인증



aws 계정 생성, 데이터 베이스 인스턴스

Amazon RDS

대시보드
데이터베이스
백리 편집기
성능 개선 도구
스냅샷
자동 백업
예약 인스턴스
프록시

서브넷 그룹
파라미터 그룹
옵션 그룹
사용자 지정 엔진 버전

이벤트
이벤트 목록

권장 사항 1
인용서 업데이트

요약

DB 식별자
db-sd2022

역할
인스턴스

CPU
6.27%

현재 활동
0 연결

상태
사용 가능

엔진
MySQL Community

클래스
db.t2.micro

리전 및 AZ
us-east-1b

연결 & 보안

모니터링

로그 및 이벤트

구성

유지 관리 및 백업

태그

CloudWatch (30)

범례: db-sd2022

Q

비교할 인스턴스 추가

모니터링

지난 한 시간

< 1 2 3 4 5 >

CPU 사용률 (%)

DB 연결 (개)

여유 스토리지 공간 (MB/초)

사용 가능 메모리 (MB)

MySQL Workbench -> aws 연결

Manage Server Connections

MySQL Connections

- Local instance wampstac...
- AWS_Con

Connection Name: AWS_Con

Connection Remote Management System Profile

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: db-sd2022.cbmzxbmxkhi4.us Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: SD_admin Name of the user to connect with.

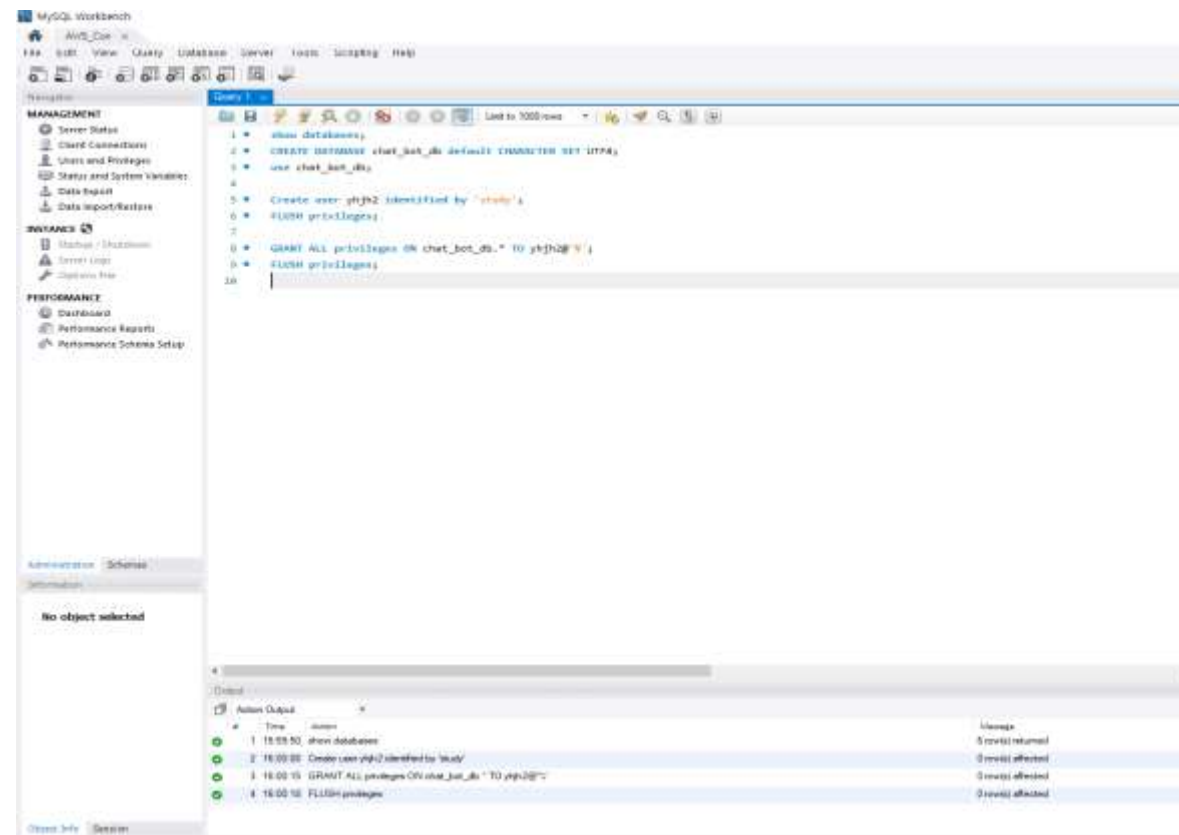
Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

New Delete Duplicate Move Up Move Down Test Connection Close



```
1 • show databases;
2 • CREATE DATABASE chat_bot_db default CHARACTER SET UTF8;
3 • use chat_bot_db;
4
5 • Create user yjh2 identified by 'study';
6 • FLUSH privileges;
7
8 • GRANT ALL privileges ON chat_bot_db.* TO yjh2@'%';
9 • FLUSH privileges;
10
```



MySQL Workbench aws 연결

- DataBase 생성
- User 생성
- 권한 부여

RoBERTa 논문 정리(논문 리뷰) - A Robustly Optimized BERT Pretraining Approach (RoBERTa)

포스팅 개요 이번 포스팅은 자연어처리(NLP) 논문 중 A Robustly Optimized BERT Pretraining Approach라는 논문을 리뷰하는 포스팅입니다. 해당 자연어처리 논문은 흔히 RoBERTa라고 많이 언급되는 논문인데요. 앞서 GPT-1, GPT-2, BERT 논문 리뷰에 이어서 자연어처리 논문 시리즈 네 번째 포스팅입니다. 추가로 해당 포스팅의 내용은 제가 진행하는 사내 자연어 처리 스터디에서 발표한 자료를 블로그로 정리한 자료임을 알려드립니다. 자연어 처리 논문...

deep learning(딥러닝) | 2021. 10. 25. 09:24



생성적 적대적 신경망을 이용한 생성기반 멀티턴 챗봇

김진태[○], 김학수, 권오욱[◆], 김영길[◆]
강원대학교 IT대학 컴퓨터정보통신공학전공[○], 한국전자통신연구원[◆]
{lsxo1119, nipdrkim}@kangwon.ac.kr, {ohwoog, kimyk}@etri.re.kr

Generative Multi-Turn Chatbot Using Generative Adversarial Network

Jintae Kim[○], Harksoo Kim, Oh-Woog Kwon[◆], Young-Gil Kim[◆]
Kangwon National University Computer and Communication Engineering[○]
Electronics and Telecommunications Research Institute[◆]

음성인식을 이용한 자막 자동생성 시스템

손원섭^{*} · 김웅곤^{**}

Subtitle Automatic Generation System using Speech to Text

Won-Seob Son^{*} · Eung-Kon Kim^{**}

▲ 테크니컬 리뷰



언어 모델

$$P(w \mid context)$$

단어 시퀀스(문장)에 확률을 부여하는 모델

어제

어제

어제

어제

어제

어제

카페

카페

카페

카페

카페

갔었어

갔었어

갔었어

갔었어

거기

거기

거기

사람

사람

많더라

순방향(LEFT-TO-RIGHT) 언어모델

많더라

사람

많더라

거기

사람

많더라

갔었어

거기

사람

많더라

카페

카페

어제

갔었어

거기

사람

많더라

역방향(RIGHT-TO-LEFT) 언어모델

$P(w_1, w_2, w_3, w_4, \dots, w_n) = \prod_{i=1}^n P(w_i \mid w_1, \dots, w_{i-1})$
이전 단어들이 주어졌을 때 다음 단어가 등장할 확률의 연쇄

언어 모델

어제	카페	갔었어	거기	사람	많더라
어제	<input type="text"/>	갔었어	거기	사람	많더라
어제	카페	<input type="text"/>	거기	사람	많더라
어제	카페	갔었어	<input type="text"/>	사람	많더라
어제	카페	갔었어	거기	<input type="text"/>	많더라
어제	카페	갔었어	거기	사람	<input type="text"/>

마스크 언어모델 (Masked Language Model)

문장 전체를 읽고
가려진 단어를 맞추는 모델

어제	카페	<input type="text"/>	거기	사람	많더라
어제	<input type="text"/>	갔었어	<input type="text"/>	사람	많더라

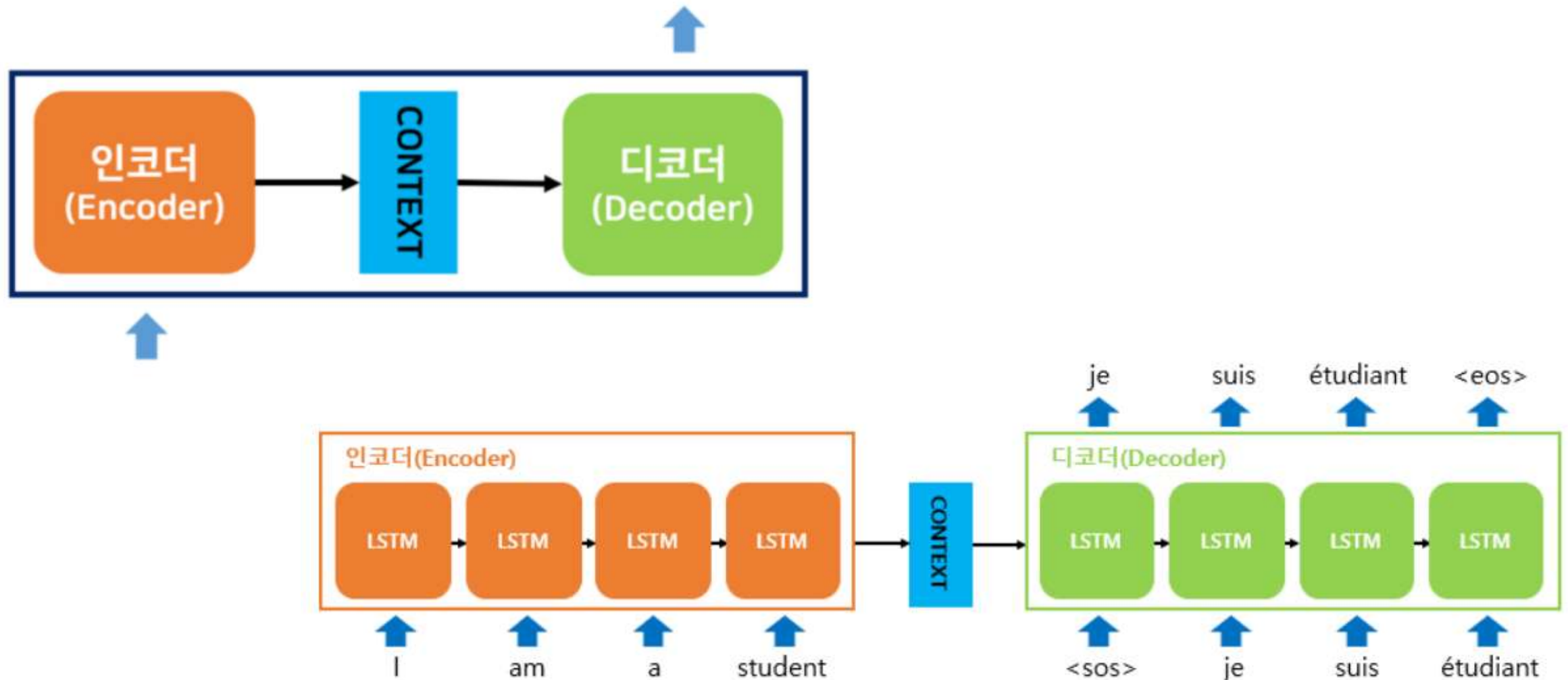
스킵-그램 모델 (Skip-Gram Model)

타겟 앞뒤 특정 범위 내 분포하는
단어를 학습하는 모델

트랜스포머

Sequence to sequence

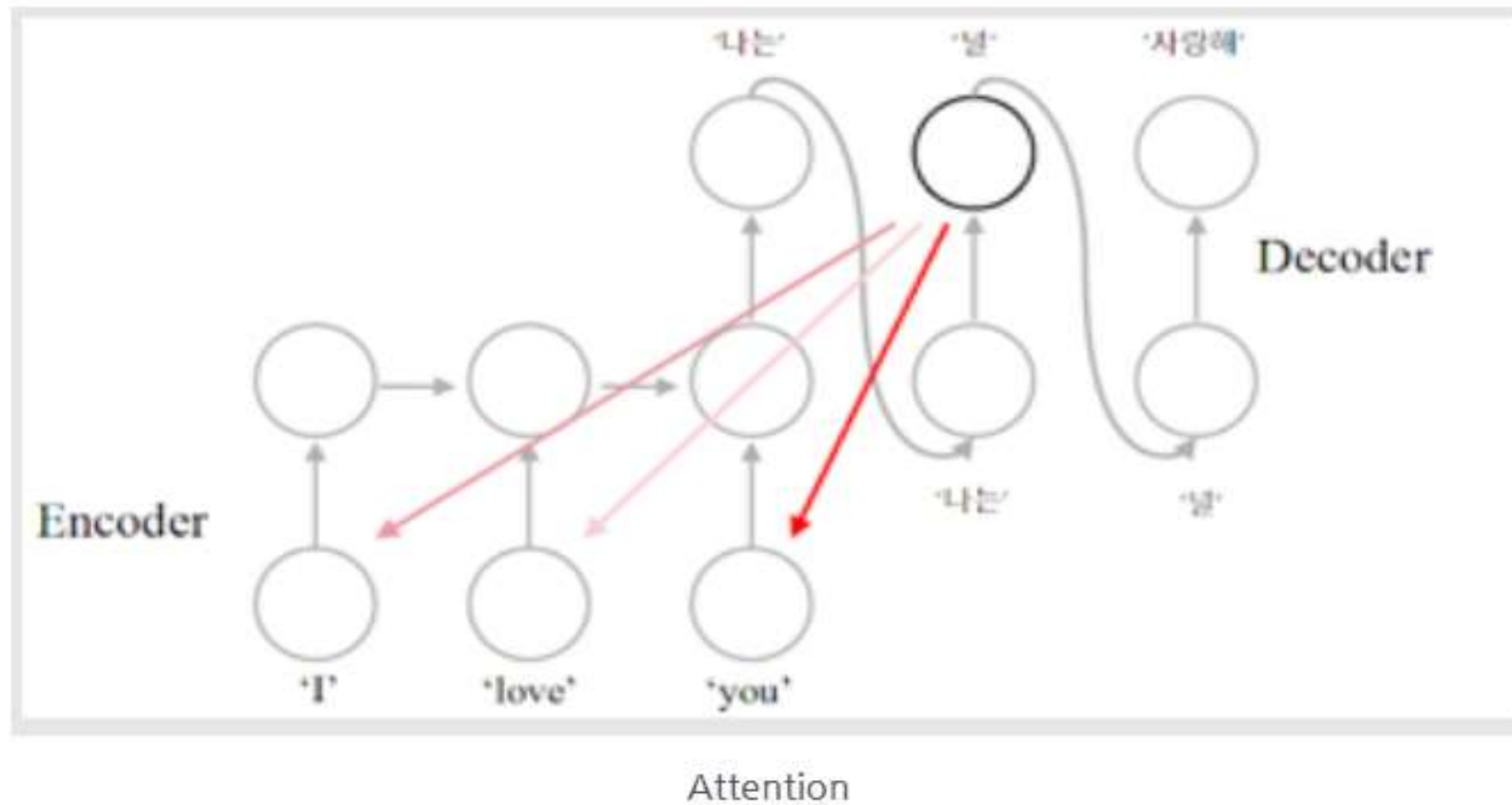
특정 속성을 지닌 문장을 다른 속성의 문장으로 변환하는 작업



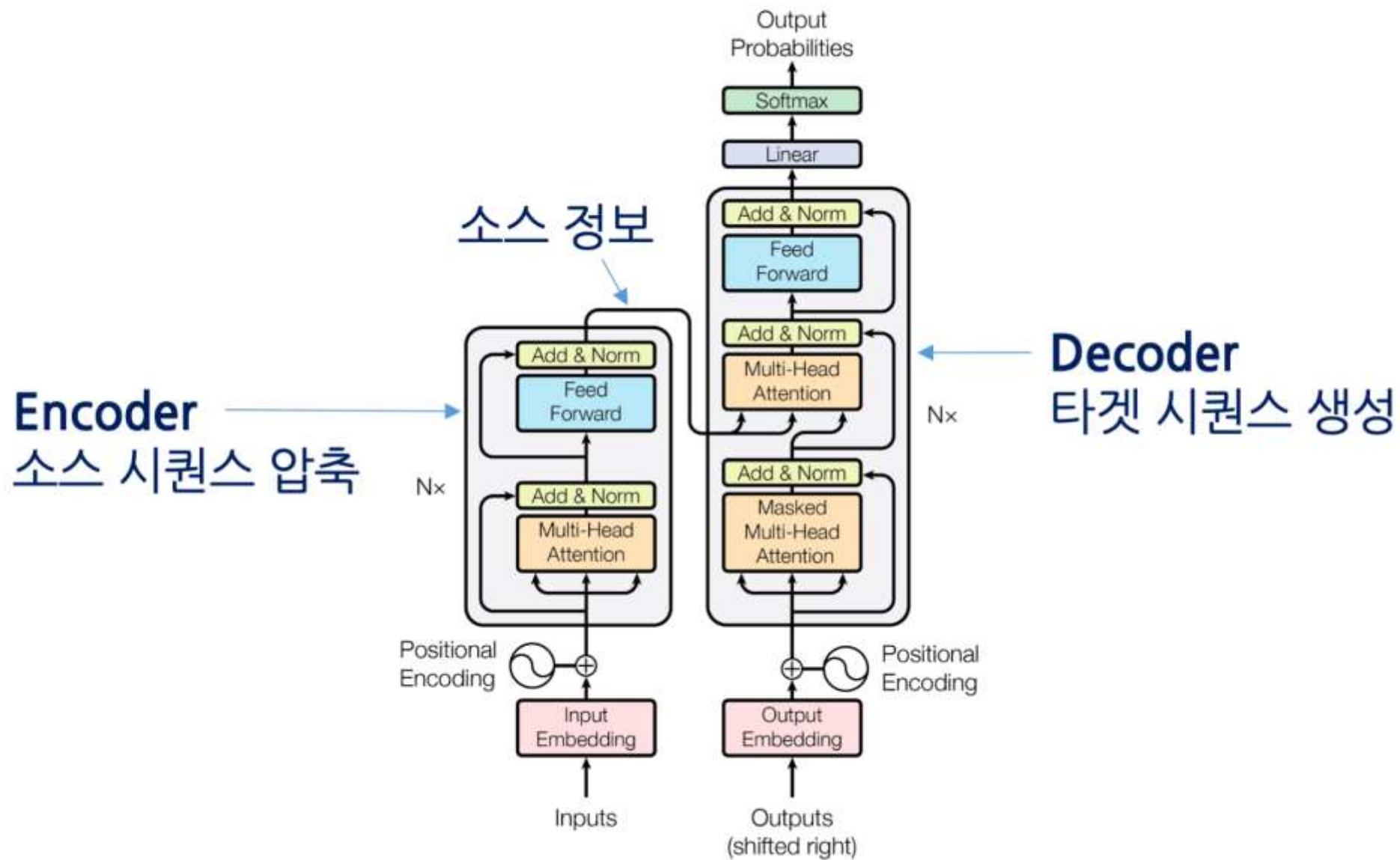
트랜스포머

Attention

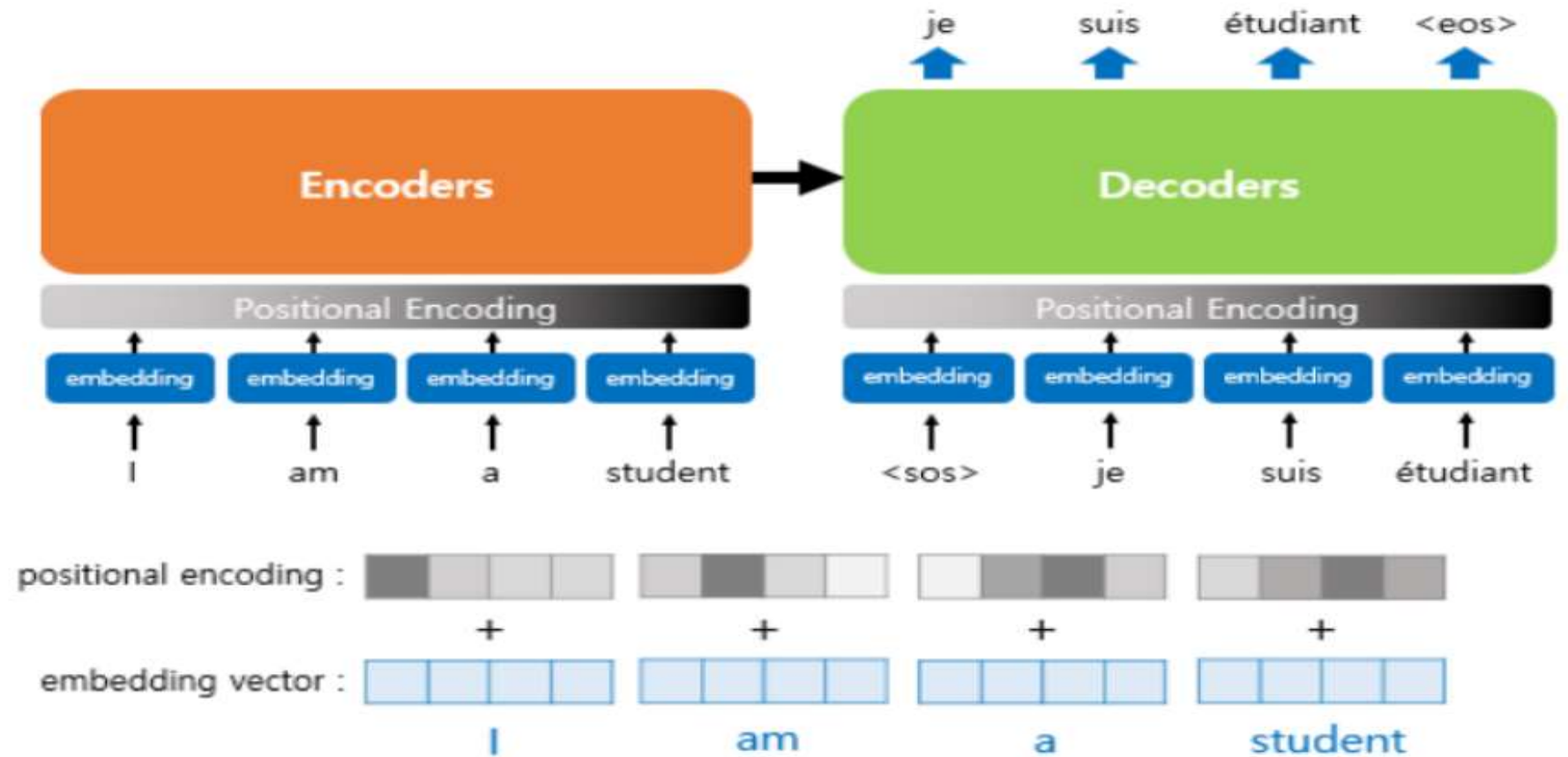
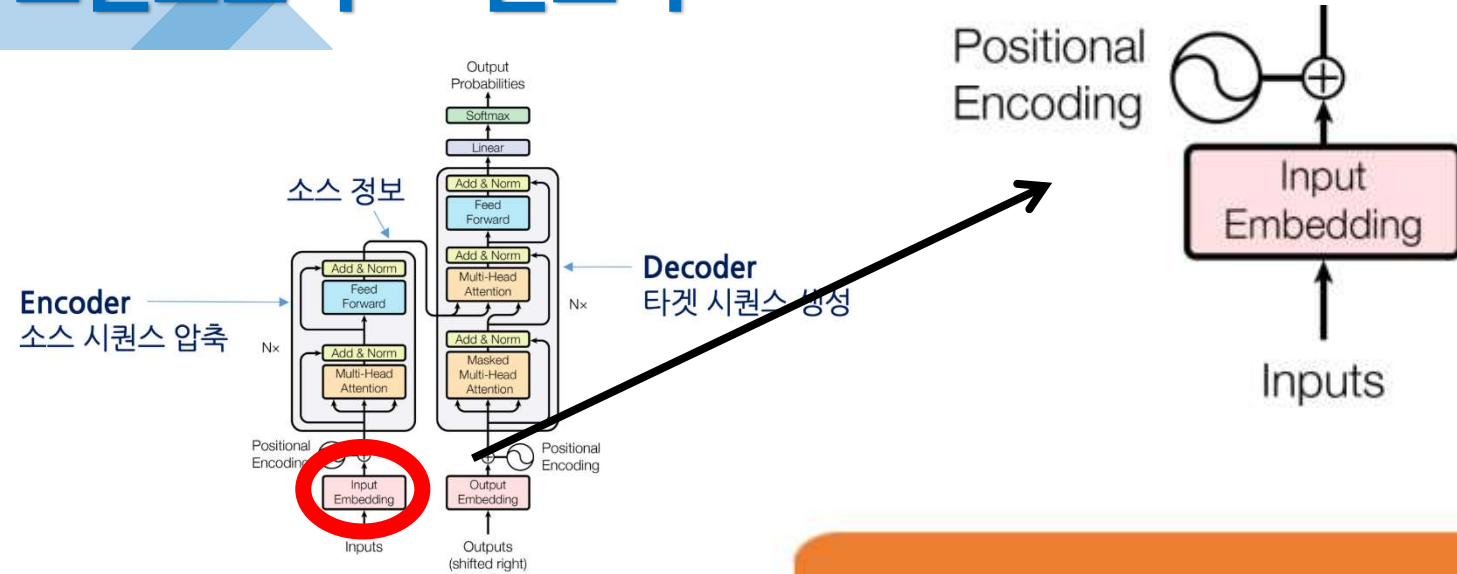
seq2seq의 문제점으로 인해 **Attention**이 탄생



트랜스포머 - 구조

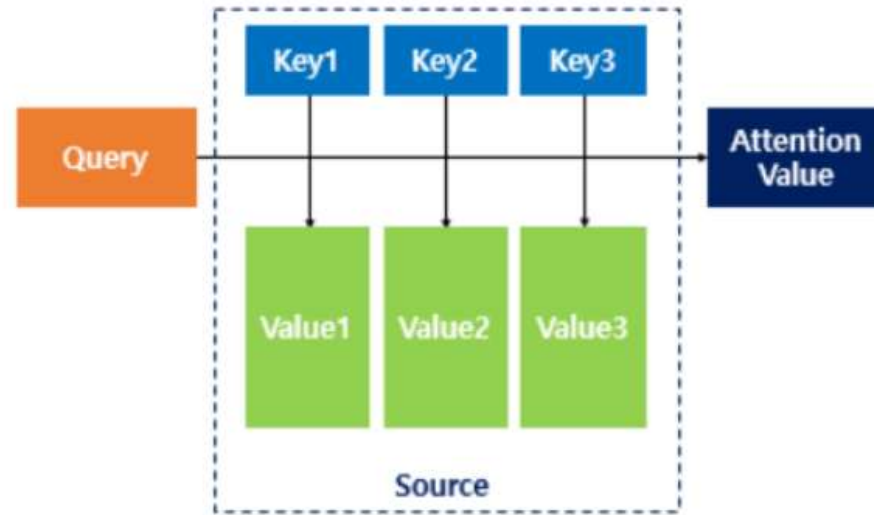
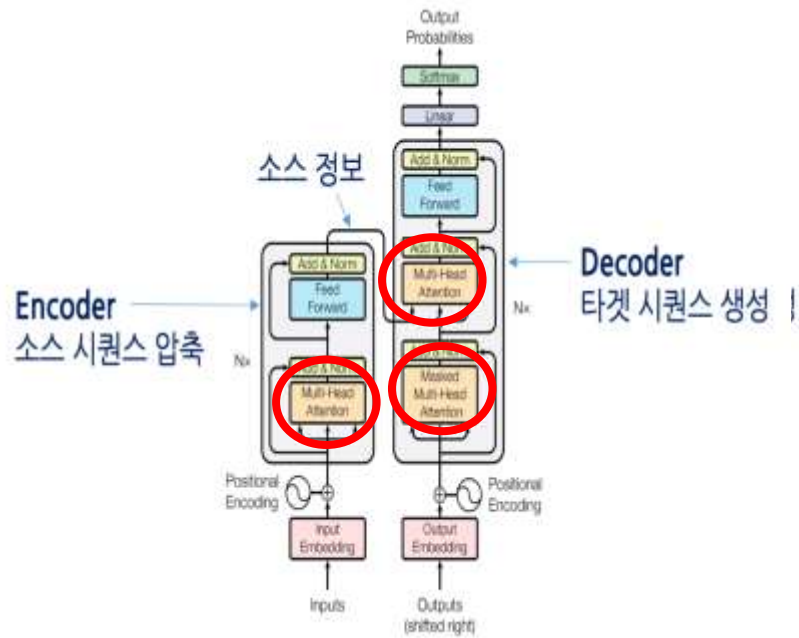


트랜스포머 - 인코더

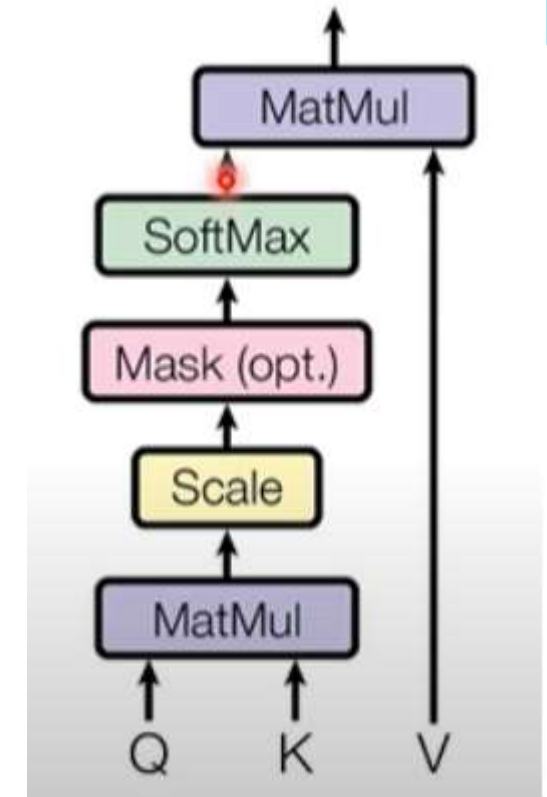


트랜스포머

Attention 구조



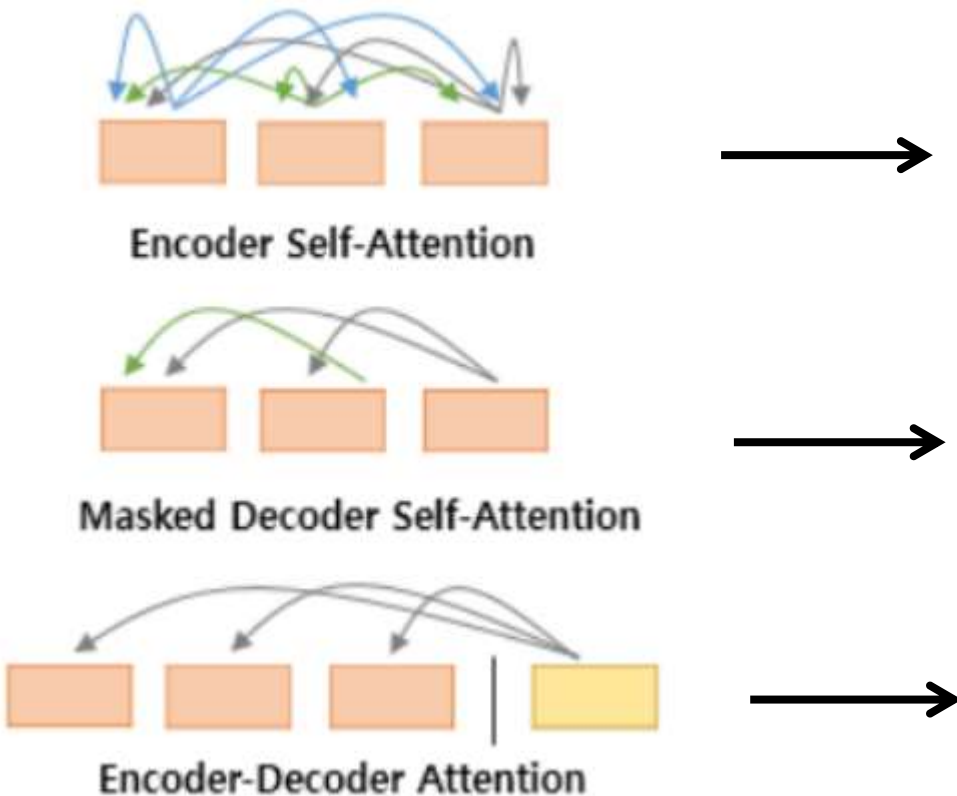
Query, Key, Value



Ex) I am a boy
 Query: 물어보는 주체 ex) I
 Key: 물어보는 대상 ex) I, am, a, boy

Self-Attention

Attention 작동 방식



1. Encoder Self-Attention

- 인코더 해당

2. Masked Decoder Self-Attention

- 디코더 해당

3. Encoder-Decoder Attention

- 디코더 해당

Self-Attention

Multi-head attention

- 셀프 어텐션(self attention)을 여러 번 수행
- 여러 헤드가 독자적으로 셀프 어텐션을 계산

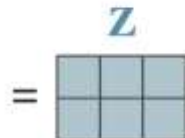
① 모든 헤드의 셀프 어텐션 출력 결과를 이어 붙인다.



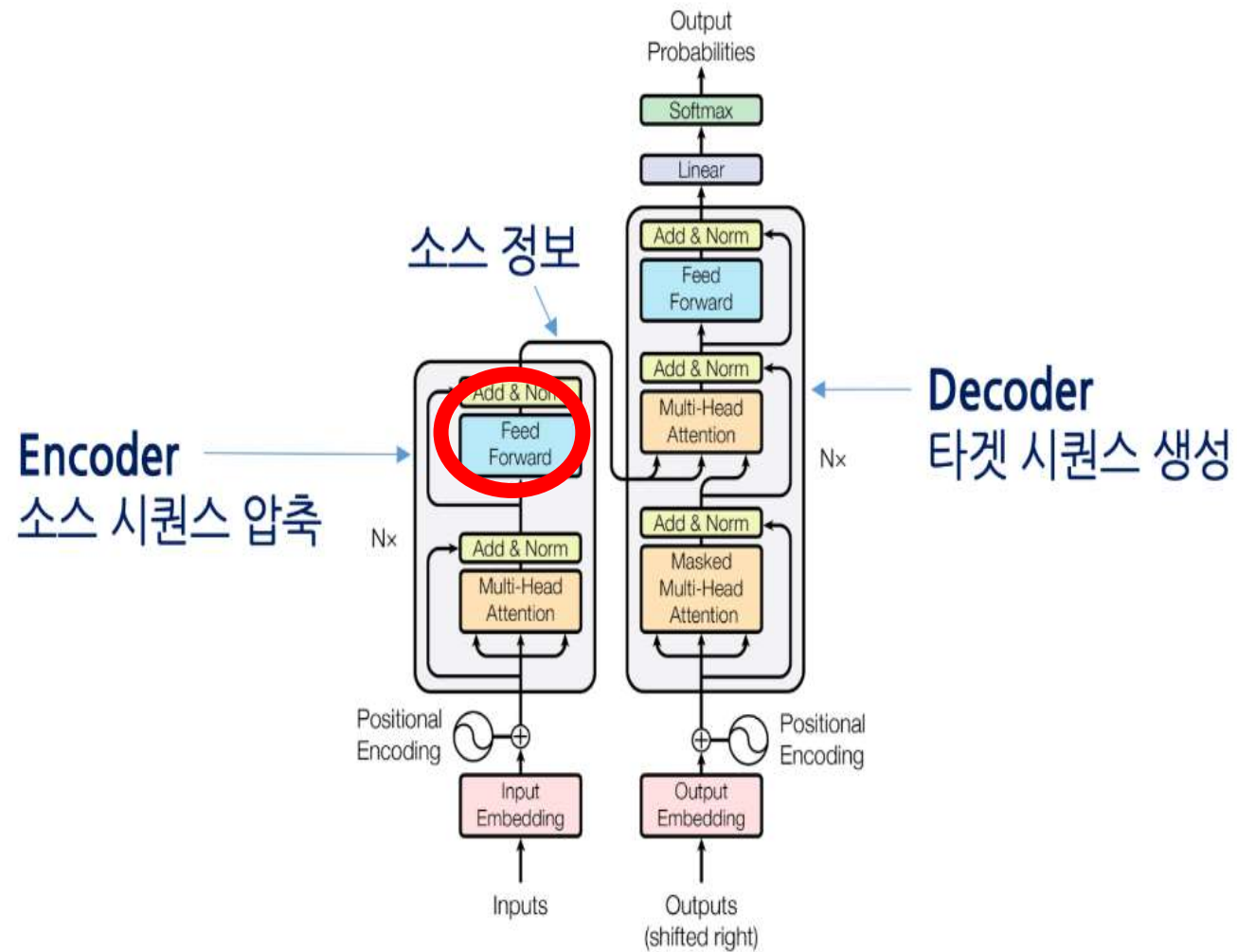
② ①의 결과로 도출된 행렬에 W^O 를 곱한다. 이 행렬은 개별 헤드의 셀프 어텐션 관련 다른 행렬(W_Q, W_K, W_V)과 마찬가지로 태스크(기계 번역)를 가장 잘 수행하는 방향으로 업데이트된다.



③ 새롭게 도출된 Z 행렬은 동일한 입력(문서)에 대해 각각의 헤드가 분석한 결과의 총합이다.



트랜스포머



Feed Forward

- 2개의 Dense Layer와 Relu 함수로 구성

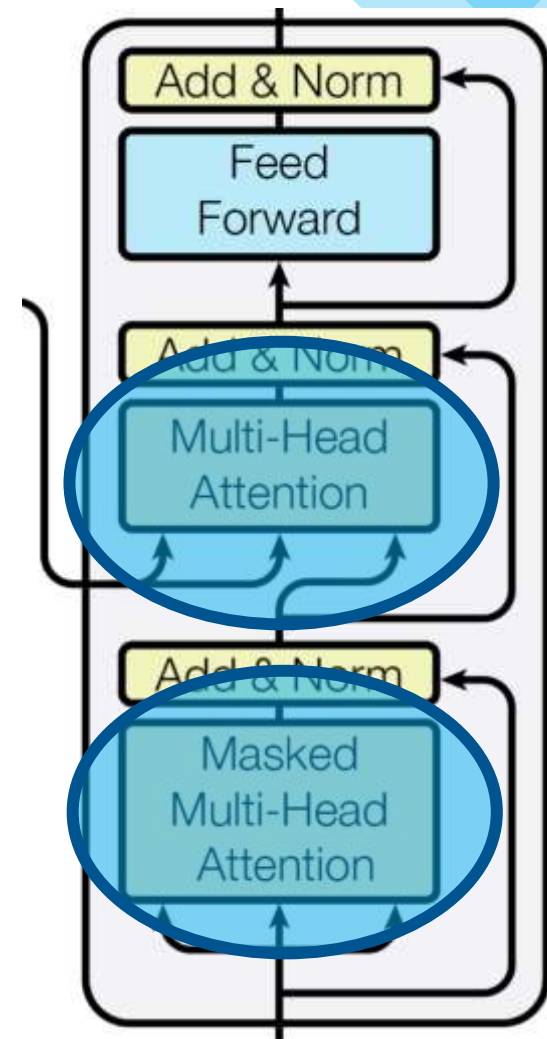
Add & Norm

- 레이어 정규화, 잔차연결 역할

트랜스포머 - 디코더

- ✓ 인코더-디코더 멀티 헤드 어텐션
(Encoder-Decoder Multi-Head Attention)
- ✓ 마스크를 적용한 멀티 헤드 어텐션
(Masked Multi-Head Attention)

Decoder block



Self-Attention – 디코더 (작동 방식)

마스크 멀티 헤드 어텐션(Masked Multi-Head Attention)

ex) 어제 카페 **갔었어** 거기 사람 많더라

<sos>	<u>mask</u>	mask	mask
<sos>	Je	mask	mask
<sos>	Je	vais	mask
<sos>	Je	vais	bien

쿼리	키	
어제	어제	
카페	카페	→ <u>갔었어</u>
갔었어	갔었어	
거기	거기	
사람	사람	
많더라	많더라	

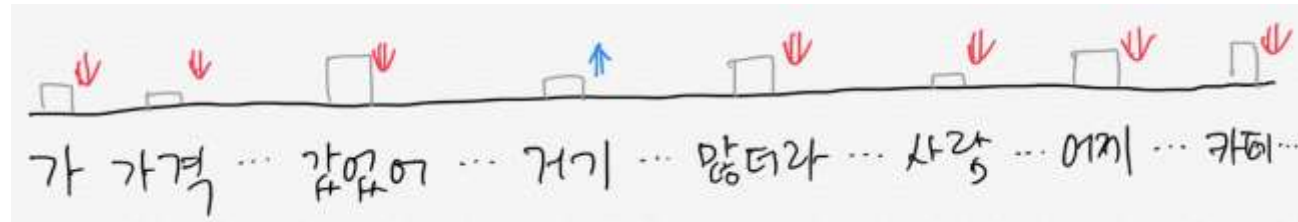
masking

Self-Attention – 디코더 (작동 방식)

마스크 멀티 헤드 어텐션(Masked Multi-Head Attention)

ex) 어제 카페 갔었어 **거기** 사람 많더라

Masking 이유



입력되는 단어에만 집중해 단어를 정확하게 생성하는 효과

거기에 해당하는 확률은 높이고 나머지 단어의 확률은 낮아지도록

Transfer Learning

Knowledge transfer

Task 2 다운스트림 태스크

자기 지도 학습(self-supervised learning)

The diagram shows two horizontal sequences of tokens in boxes. The top sequence, labeled 'GPT' on the right, contains the tokens 'I', 'AM', 'a', and a grey box with a question mark '?'. The bottom sequence, labeled 'BERT' on the right, contains the tokens 'I', 'AM', a grey box with a question mark '?', and 'boy'. A light blue bar is at the top of the slide.

다음 단어 맞추기
(언어모델)

빈칸 채우기
(마스크 언어모델)

분류(classification)

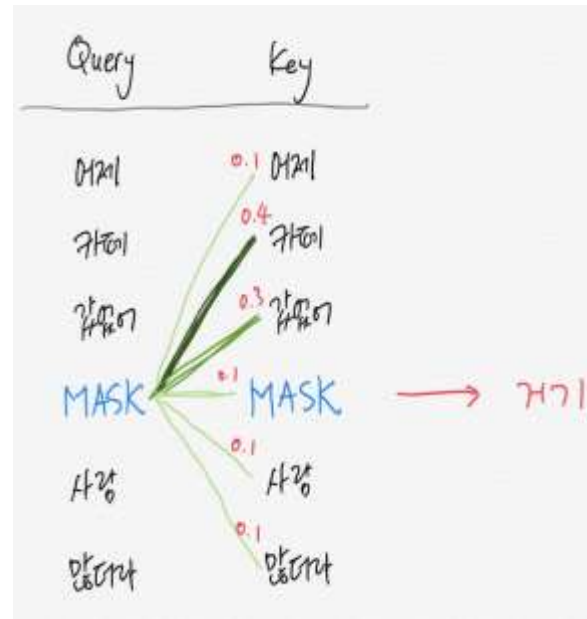
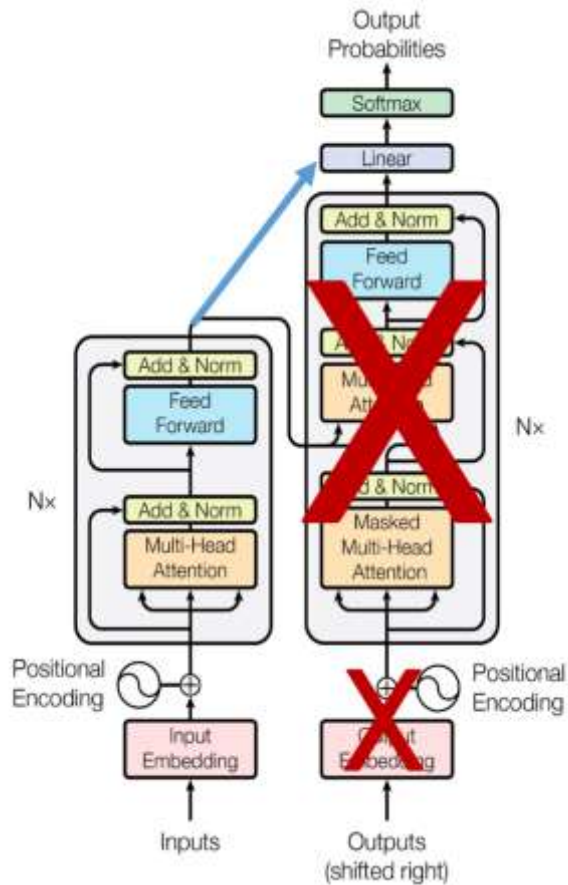


BERT

어제 카페 갔었어 사람 많더라

BERT(Bidirectional Encoder Representations from Transformers)

문장 중간에 빈칸을 만들고 해당 빈칸에 어떤 단어가 적절할지 맞추며 빈칸 앞뒤 문맥을 모두 살피는 양방향(bidirectional) 마스크 언어모델(Masked Language Model)



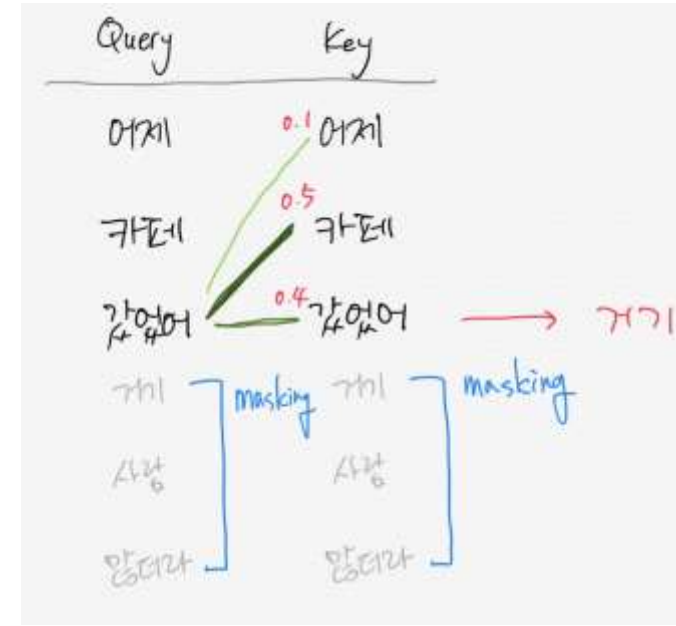
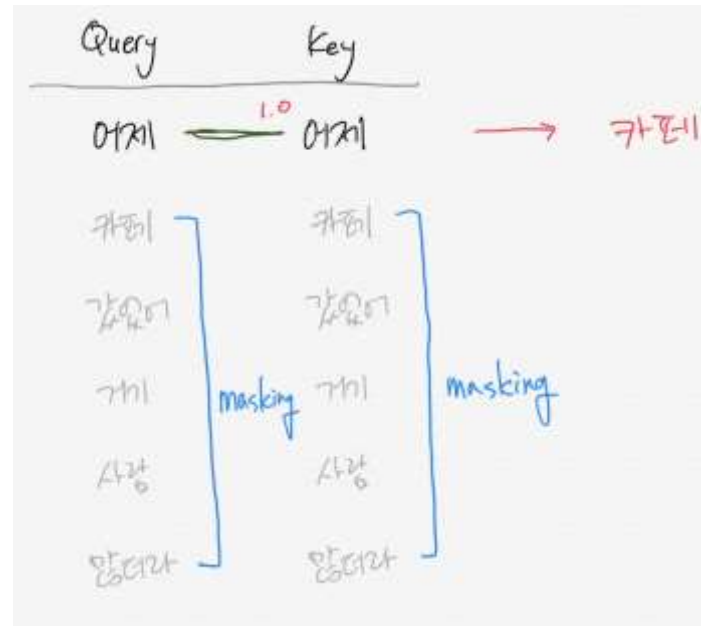
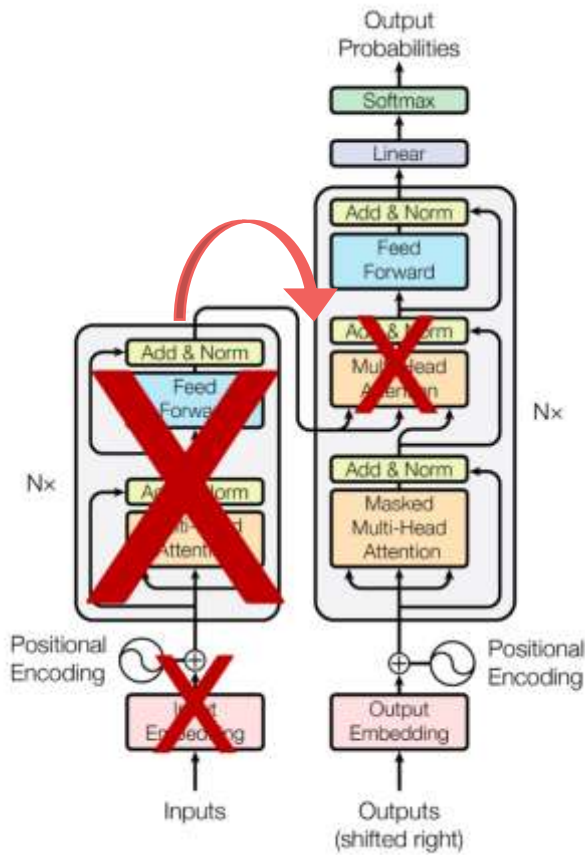
- 트랜스포머 Encoder 해당

GPT

어제 카페 갔었어 거기 사람 많더라

GPT(Generative Pre-trained Transformer)

이전 단어들이 주어졌을 때 다음 단어가 무엇인지 맞추는 과정에서 프리트레인(pretrain) 문장 시작부터 순차적으로 계산하는 **일방향(unidirectional)** 언어모델(Language Model)



- 트랜스포머 Decoder 해당

모델 불러오기, 결측치 제거

```
koGPT2_TOKENIZER = PreTrainedTokenizerFast.from_pretrained("skt/kogpt2-base-v2",
    bos_token=BOS, eos_token=EOS, unk_token='<unk>',
    pad_token=PAD, mask_token=MASK)
model = GPT2LMHeadModel.from_pretrained('skt/kogpt2-base-v2')

path = 'C:\\\\Users\\\\비트캠프\\Desktop\\Team\\'
Chatbot_Data = pd.read_csv(path + "dataset_final.csv")
Chatbot_Data = Chatbot_Data.dropna()
```

```

class ChatbotDataset(Dataset):
    def __init__(self, chats, max_len=40):
        self._data = chats
        self.max_len = max_len
        self.q_token = Q_TKN
        self.a_token = A_TKN
        self.sent_token = SENT
        self.eos = EOS
        self.mask = MASK
        self.tokenizer = koGPT2_TOKENIZER

    def __len__(self):
        return len(self._data)

    def __getitem__(self, idx):
        turn = self._data.iloc[idx]
        q = turn["Q"]
        q = re.sub(r"([?.!,])", r" ", q)
        a = turn["A"]
        a = re.sub(r"([?.!,])", r" ", a)
        q_toked = self.tokenizer.tokenize(self.q_token + q + self.sent_token)
        q_len = len(q_toked)
        a_toked = self.tokenizer.tokenize(self.a_token + a + self.eos)
        a_len = len(a_toked)

```

```

        labels = [self.mask,] * q_len + a_toked[1:]

```

```

        mask = [0] * q_len + [1] * a_len + [0] * (self.max_len - q_len - a_len)
        labels_ids = self.tokenizer.convert_tokens_to_ids(labels)

```

```

        while len(labels_ids) < self.max_len:
            labels_ids += [self.tokenizer.pad_token_id]

```

```

        token_ids = self.tokenizer.convert_tokens_to_ids(q_toked + a_toked)

```

```

        while len(token_ids) < self.max_len:
            token_ids += [self.tokenizer.pad_token_id]

```

```

        return (token_ids, np.array(mask), labels_ids)

```

```

def collate_batch(batch):
    data = [item[0] for item in batch]
    mask = [item[1] for item in batch]
    label = [item[2] for item in batch]
    return torch.LongTensor(data), torch.LongTensor(mask), torch.LongTensor(label)

```

데이터 토크나이징


```
print ("학습 시작")
for epoch in range(epoch):
    for batch_idx, samples in enumerate(train_dataloader):
        optimizer.zero_grad()
        token_ids, mask, label = samples
        out = model(token_ids)
        out = out.logits
        mask_3d = mask.unsqueeze(dim=2).repeat_interleave(repeats=out.shape[2], dim=2)
        mask_out = torch.where(mask_3d == 1, out, Sneg * torch.ones_like(out))
        loss = criterion(mask_out.transpose(2, 1), label)
        avg_loss = loss.sum() / mask.sum()
        avg_loss.backward()
        optimizer.step()
print ("학습 종료")
```

파인튜닝

1. zero_grad()

Gradient 초기화
= Sets gradients of all model parameters
to zero.

파이토치는 미분을 통해 얻은 기울기를 이전에 계산된 기울기 값에 누적시키는 특징이 있습니다.

```
import torch  
w = torch.tensor(2.0, requires_grad=True)
```

$$E_{total} = \frac{1}{2}(target_{o1} - output_{o1})^2 + \frac{1}{2}(target_{o2} - output_{o2})^2$$

- 역전파 단계를 실행하기 전에 변화도를 0으로 만들
- .backward()를 호출할 때마다 변화도가 누적되기 때문에 변화도를 업데이트 하기 전에 초기화

수식을 w로 미분한 값 : 2.0
수식을 w로 미분한 값 : 4.0
수식을 w로 미분한 값 : 6.0
수식을 w로 미분한 값 : 8.0
수식을 w로 미분한 값 : 10.0
수식을 w로 미분한 값 : 12.0
수식을 w로 미분한 값 : 14.0
수식을 w로 미분한 값 : 16.0
수식을 w로 미분한 값 : 18.0
수식을 w로 미분한 값 : 20.0
수식을 w로 미분한 값 : 22.0
수식을 w로 미분한 값 : 24.0
수식을 w로 미분한 값 : 26.0
수식을 w로 미분한 값 : 28.0
수식을 w로 미분한 값 : 30.0
수식을 w로 미분한 값 : 32.0
수식을 w로 미분한 값 : 34.0
수식을 w로 미분한 값 : 36.0
수식을 w로 미분한 값 : 38.0
수식을 w로 미분한 값 : 40.0
수식을 w로 미분한 값 : 42.0


2. zero_grad(), loss.backward(), optimizer.step()

`loss.backward()`

- 모델의 매개변수들에 대한 손실의 변화도를 계산

`optimizer.step()`

- `step` 함수를 호출하면 역전파 단계에서 수집된 변화도로 매개변수를 조정


$$W(t+1) = W_t - \frac{\partial \text{오차}}{\partial W}$$

다층 퍼셉트론에서의 최적화 과정 = 오차 역전파



Python pymysql

```
import pymysql
import pandas as pd

host = 'db-sd2022.cbmzxbmxkhi4.us-east-1.rds.amazonaws.com'
port = 3306
master_admin = 'yhjh'
db='chat_bot_db'
pw = 'study'

aws_db = pymysql.connect(
    user=master_admin,
    passwd=pw,
    host=host,
    db=db,
    charset='utf8'
)
cursor = aws_db.cursor(pymysql.cursors.DictCursor)
```

aws 연결

Python pymysql

```
def init_table():
    table = "Question_and_Answer"
    delete_table = f"Drop Table if exists {table}"
    cursor.execute(delete_table)
    print(delete_table)
    create_table = f"create table if not exists {table} (id MEDIUMINT NOT NULL AUTO_INCREMENT, Question VARCHAR(2000), Answer VARCHAR(2000), PRIMARY KEY (id))"
    cursor.execute(create_table)
    aws_db.commit()
    print(create_table)
    print()
    aws_db.close()

def insert_QnA(q,a):
    table = "Question_and_Answer"
    sql = f"INSERT INTO {table} (Question,Answer) VALUES ('{q}','{a}');"
    cursor.execute(sql)
    # print(sql)
    aws_db.commit()

def select_Table():
    table = "Question_and_Answer"
    sql = f"Select Question, Answer from {table}"
    cursor.execute(sql)
    result = pd.read_sql_query(sql,aws_db)
    return result
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Question	Answer
0	보이스피싱 다 해 본 적 있어요	보이스피싱은 돈을 잃는 경우가 많습니다
2	보이스피싱 다 해 본 적 있어요	보이스피싱은 돈을 잃는 경우가 많습니다
3	만만한게 없네	만만한게 뭐가 있어
4	권리 오겠습니다	웅웅
...
113	너와 나의 연결 고리	연결 고리가 뭔지 모르겠어
114	우리 무슨 사이야	내가 아니라 너님이랑 사이 좋게 지내
115	관계를 명확하게 해 줬으면 좋겠어	웅웅 결혼도 안했으면서
116	그래도 상도로 하는게 있는 거야	웅웅
117	나 말고 딴 남자 만나	만나면 안되니까

테이블 생성 및 입력



stt / tts

```
import os
import winsound
import speech_recognition as sr

os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="chat_api.json"

def say_anything():
    try:
        r = sr.Recognizer()
        # microphone에서 audio source를 생성합니다
        with sr.Microphone() as source:
            print("<<< 마이크에 이야기 하세요 >>>")
            winsound.PlaySound("ns_1_01.wav", winsound.SND_FILENAME)
            audio = r.listen(source)
            answer = r.recognize_google(audio, language='ko')
            return answer
    except:
        print("마이크 입력 에러 입니다")
        return say_anything()
```

stt

```
import os
import playsound

def synthesize_text(text):
    os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="D:\\__ChatBot\\google_api_set\\chat_api.json"
    """Synthesizes speech from the input string of text."""
    from google.cloud import texttospeech

    client = texttospeech.TextToSpeechClient()
    input_text = texttospeech.SynthesisInput(text=text)

    # Note: the voice can also be specified by name.
    # Names of voices can be retrieved with client.list_voices().
    voice = texttospeech.VoiceSelectionParams(
        language_code="ko-KR",
        name="ko-KR-Wavenet-A",
        ssml_gender=texttospeech.SsmlVoiceGender.FEMALE,
    )
    audio_config = texttospeech.AudioConfig(
        audio_encoding=texttospeech.AudioEncoding.MP3
    )
    response = client.synthesize_speech(
        request={"input": input_text, "voice": voice, "audio_config": audio_config}
    )
    # The response's audio_content is binary.
    with open("D:\\__ChatBot\\google_api_set\\output.mp3", "wb") as out:
        out.write(response.audio_content)
        # print('Audio content written to file "output.mp3"')
```

tts



출력

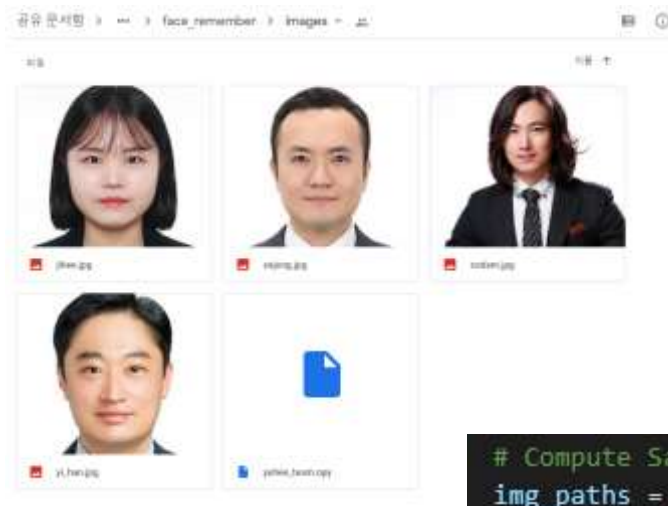
```
sent = '0'
with torch.no_grad():
    while True:
        print("-----")
        q=""
        q = google_stt.say_anything().strip()
        if q == "잘자":
            a = "즐거운 대화였어용"
            print(f"유희 > {a}")
            google_tts.synthesize_text(a)
            playsound.playsound("D:\\__ChatBot\\google_api_set\\output.mp3")
            break

        print(f"나 >> {q}")
        a = ""
        while True:
            input_ids = torch.LongTensor(koGPT2_TOKENIZER.encode(Q_TKN + q + SENT + sent + A_TKN + a)).unsqueeze(dim=0)
            pred = model(input_ids)
            pred = pred.logits
            gen = koGPT2_TOKENIZER.convert_ids_to_tokens(torch.argmax(pred, dim=-1).squeeze().numpy().tolist())[-1]
            if gen == EOS:
                break
            a += gen.replace("_", " ")

        a = a.strip()
        print(f"유희 > {a}")
        google_tts.synthesize_text(a)
        playsound.playsound("D:\\__ChatBot\\google_api_set\\output.mp3")
        aws_sql.insert_QnA(q,a)
        continue
```



Face Remember



```
cap = cv2.VideoCapture(0) # 웹캠을 카메라로 사용
cap.set(3,640) # 너비
cap.set(4,480) # 높이
```

```
def selfy():
    ret, frame = cap.read() # 사진 촬영
    frame = cv2.flip(frame, 1) # 좌우 대칭

    cv2.imwrite('test_img\my_pic.jpg', frame) # 사진 저장

    cap.release()
    cv2.destroyAllWindows()
    print("찰칵~")
```

```
if __name__ == '__main__':
    selfy()
```

```
detector = dlib.get_frontal_face_detector()
sp = dlib.shape_predictor('models/shape_predictor_68_face_landmarks.dat')
facerec = dlib.face_recognition_model_v1('models/dlib_face_recognition_resnet_model_v1')
```

```
def find_faces(img):
    dets = detector(img, 1)
    # face Not Found empty 0 return
    if len(dets) == 0:
        return np.empty(0), np.empty(0), np.empty(0)

    rests, shapes = [], []
    shapes_np = np.zeros((len(dets), 68, 2), dtype=np.int)

    for k, d in enumerate(dets):
        rect = ((d.left(), d.top()), (d.right(), d.bottom()))
        rests.append(rect)
        shape = sp(img, d)

        # convert dlib shape to numpy array
        for i in range(0, 68):
            shapes_np[k][i] = (shape.part(i).x, shape.part(i).y)

        shapes.append(shape)
    return rests, shapes, shapes_np
```

```
def encode_faces(img, shapes):
    face_descriptors = []
    for shape in shapes:
        face_descriptor = facerec.compute_face_descriptor(img, shape)
        face_descriptors.append(np.array(face_descriptor))
    return np.array(face_descriptors)
```

```
# Compute Saved Face Description
```

```
img_paths = {
    '소담': 'images/sodam.jpg',
    '이한': 'images/yi_han.jpg',
    '세종': 'images/sejong.jpg',
    '지희': 'images/jihee.jpg'
}
```

```
descs = {
    '소담': None,
    '이한': None,
    '세종': None,
    '지희': None
}
```

```
for name, img_paths in img_paths.items():
    img_bgr = cv2.imread(img_paths)
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.show()

    _, img_shapes, _ = find_faces(img_rgb)

    desc = encode_faces(img_rgb, img_shapes)[0]
    descs[name] = desc
    np.save('images/yuhhee_team.npy', descs)
print(descs)
```

얼굴인식



Face Remember

```
def face_to_name():
    selfy.selfy()
    # Numpy 로 저장된 얼굴이미지 화일 읽어오기
    desc = np.load('images/yuhee_team.npy', allow_pickle=True)[()]

    # 얼굴인식을 할 이미지를 읽어오기
    img1_path = 'test_img\my_pic.jpg'
    img_bgr = read_img(img1_path)

    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
    dets = detector(img_bgr, 1)

    for k, d in enumerate(dets):
        shape = sp(img_rgb, d)
        face_descriptor = facerec.compute_face_descriptor(img_rgb, shape)
        last_found = {'name': 'unknown', 'dist': 0.6, 'color': (0, 0, 255)}

        for name, saved_desc in desc.items():
            dist = np.linalg.norm([face_descriptor] - saved_desc, axis=1)

            if dist < last_found['dist']:
                last_found = {'name': name, 'dist': dist, 'color': (255, 255, 255)}

    your_name = last_found['name']
    print(your_name)
    return your_name
```





Face Remember

```
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils

# 얼굴 탐지 모델 가중치
cascade_filename = 'haarcascade_frontalface_alt.xml'
# 모델 불러오기
cascade = cv2.CascadeClassifier(cascade_filename)

MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)

age_net = cv2.dnn.readNetFromCaffe(
    'deploy_age.prototxt',
    'age_net.caffemodel')

gender_net = cv2.dnn.readNetFromCaffe(
    'deploy_gender.prototxt',
    'gender_net.caffemodel')

age_list = ['0,2', '4,6', '8,12', '15,20', '25,32', '38,43', '48,53', '60,100']
gender_list = ['남자', '여자']
```

```
# gender detection
gender_net.setInput(blob)
gender_preds = gender_net.forward()
gender = gender_preds.argmax()

# Predict age
age_net.setInput(blob)
age_preds = age_net.forward()
age = age_preds.argmax()

age_1 = age_list[age].split(',')[0]
age_2 = age_list[age].split(',')[1]

msg = f'당신은 {age_1} 세 에서 {age_2}세 사이의 {gender_list[gender]} 입니다'
print(msg)
google_tts.synthesize_text(msg)
playsound.playsound("output.mp3")
```

나이 & 성별



Face Remember

```
# Load Yolo
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
classes = []
#클래스 이름을 따로 저장해준다. 이 형식은 클래스가 한글이름 일 때 불러오는 방식이다.
with open("kor_coco.names", "r", encoding='UTF8') as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames() # 네트워크의 모든 레이어 이름을 가져옵니다.
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
# 네트워크의 출력 레이어 이름을 가져옵니다.
colors = np.random.uniform(0, 255, size=(len(classes), 4))

# 이미지 가져오기
img = cv2.imread("wife.jpg")
img = cv2.resize(img, None, fx=0.4, fy=0.4)
height, width, channels = img.shape
```

```
for out in outs: # 출력을 각각 확인합니다.
    for detection in out: # detection = out[i] = [x, y, w, h, obj_score, class_id]
        scores = detection[5:] # [5:] 는 가장 앞의 5개를 버리고 나머지를 가져옵니다.
        class_id = np.argmax(scores) # 가장 높은 점수를 가진 클래스 아이디를 가져옵니다.
        confidence = scores[class_id]
        if confidence > 0.5: # 확률이 0.5 이상인 것만 가져옵니다.
            # Object detected
            # 탐지된 객체의 너비, 높이 및 중앙 좌표값 찾기
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            #print(center_x, center_y)
            w = int(detection[2] * width)
            h = int(detection[3] * height)
            #print(w, h)
            # 객체의 사각형 테두리 중 좌상단 좌표값 찾기
            x = int(center_x - w / 2)
            y = int(center_y - h / 2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
```

사진 분석



Face Remember

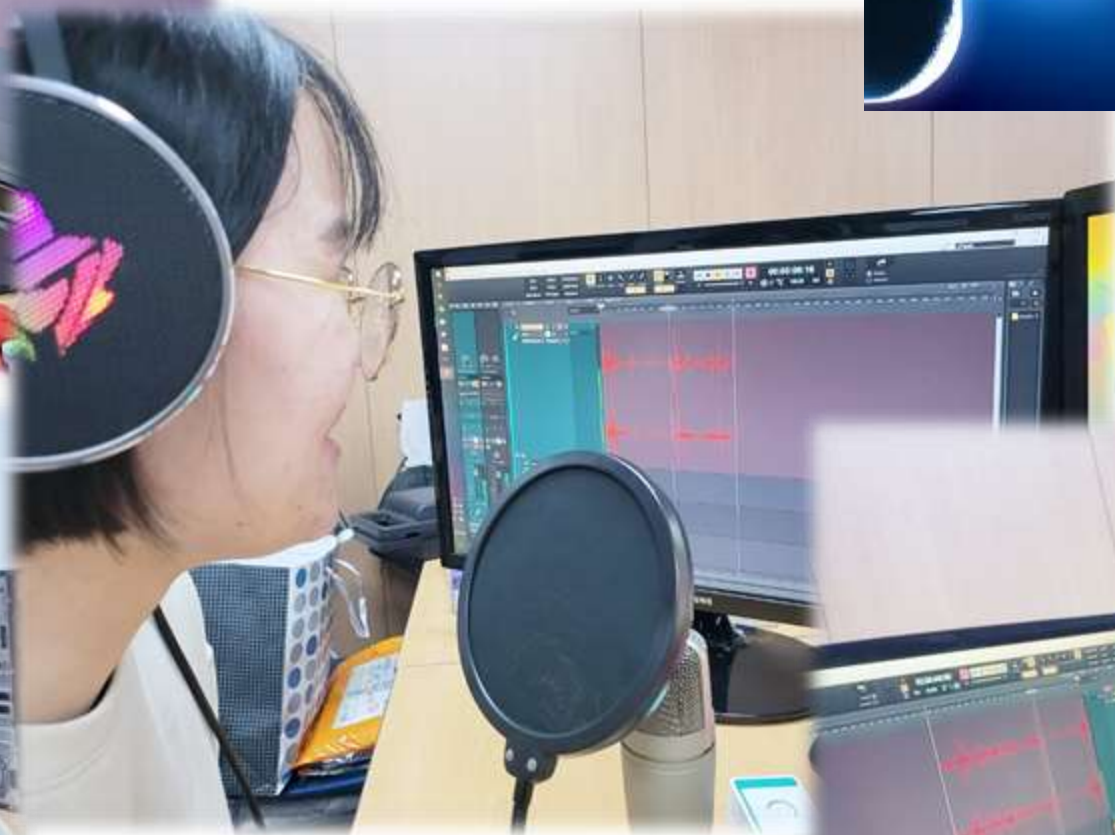
```
labels = []
for i in range(len(boxes)):
    if i in indexes:
        x, y, w, h = boxes[i] # 객체의 사각형 테두리 중 좌상단 좌표값 찾기
        label = str(classes[class_ids[i]]) # 클래스 이름을 가져옵니다.
        color = colors[i] # 색상을 가져옵니다.
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 2) # 사각형 테두리 그리기
        img = np.array(img) # 이미지를 numpy array로 변환
        # draw.text(img, label, (x, y + 30), font, 3, color, 3)
        # cv2.putText(img, label, (x, y + 30), font, 3, color, 3) # 텍스트 그리기
        labels.append(label) # 클래스 이름을 리스트에 추가

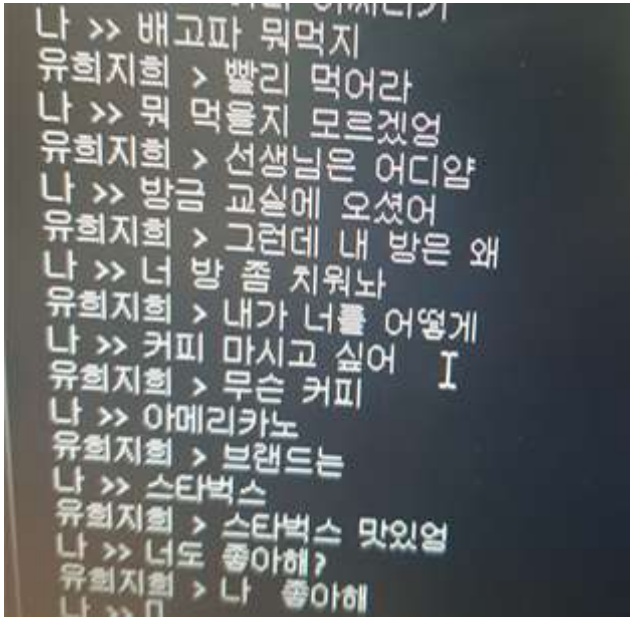
b,g,r,a = int(color[0]), int(color[1]), int(color[2]), int(color[3])
fontpath = "./malgun.ttf"
font = ImageFont.truetype(fontpath, 32)
img_pil = Image.fromarray(img)
draw = ImageDraw.Draw(img_pil)

draw.text((x, y-50), label, font = font, fill = (b, g, r, a))
img = np.array(img_pil)
```


▶ 진행 예정 프로세스 소개







Load model

유희지희

Data Set



Fit

Load모델 후
사용자와 대화한 내용을
주기적으로 학습

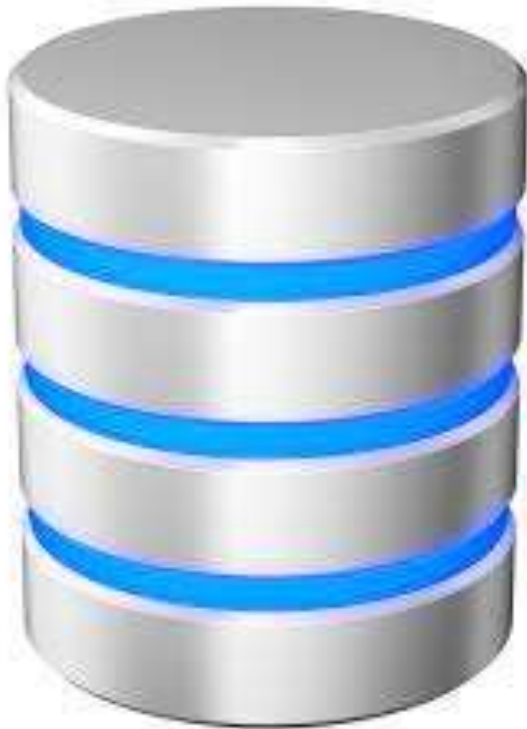


save model

유희지희

엉뚱한 답변 등 잘못된 학습 데이터 수정 필요 -> 제2의 이루다 방지

Data Set





MySQL Tables

Q> 요즘 영화 뭐가 재미있어?

A> 영화는 참 재미있어

Q> 어벤저스 볼까 생각 중

A> 보는 것 보다는 먹는게 즐겁지

Q> 어벤저스는 이번에 개봉한 영화야

A> 나도 영화 보는 거 좋아해

Question

Answer

Trained

Back Up

- 대화내역 실시간으로 Question, Answer 테이블에 저장
- 주기적으로 파인 튜닝 후 Trained 테이블에 자료 추가

Q> 요즘 영화 뭐가 재미있어?

A> 영화는 참 재미있어

Q> 어벤저스 볼까 생각 중

A> 보는 것 보다는 먹는게 즐겁지

Q> 어벤저스는 이번에 개봉한 영화야

A> 나도 영화 보는 거 좋아해

Fit

Q> 요즘 영화 뭐가 재미있어?

A> 어벤저스 어때?

Q> 지난주에 봤어 어벤저스

A> 재미있었어?

Q> 남자들은 좋아할 내용

A> 너는 어떤 영화 좋아해?

Ex) 어벤저스가 영화임을 학습

유희지희(챗봇) 스킨 구현

Python3

```
# importing the require package
from py_avataaars import PyAvataaar

# assigning various parameters to our avatar
avatar = PyAvataaar()

# rendering the avatar in png format
avatar.render_png_file("AVATAR_1.png")
```

Output:

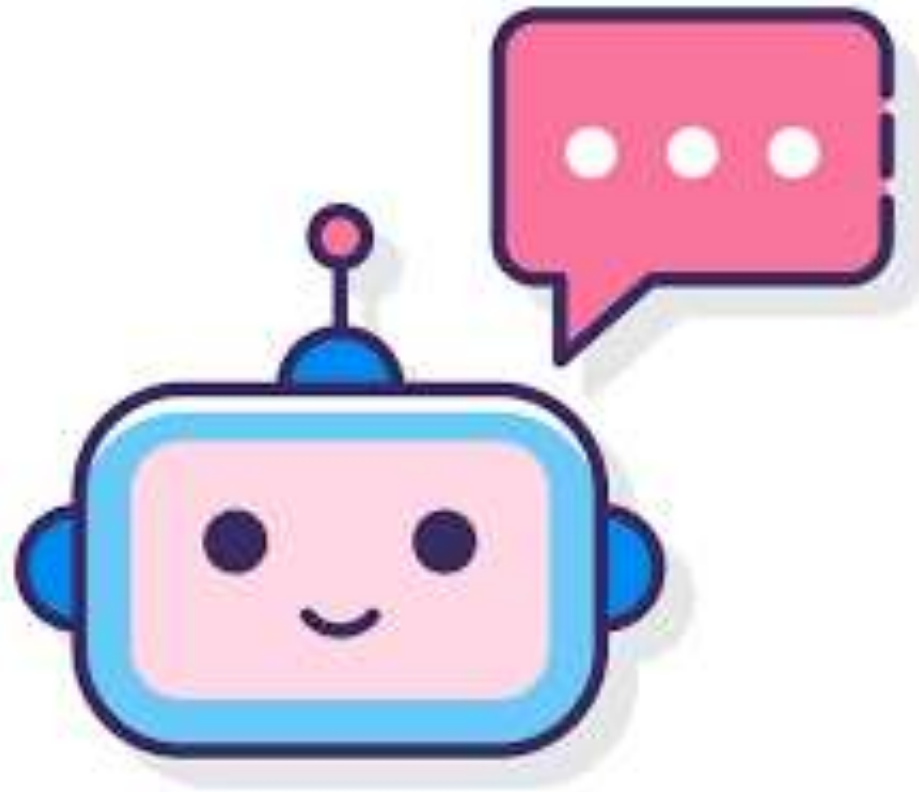


The above program will generate the *AVATAR_1.png* file in the folder where you've kept the above program. If the above program is running properly, then we will generate the avatars according to our *PyAvataaar()* method.

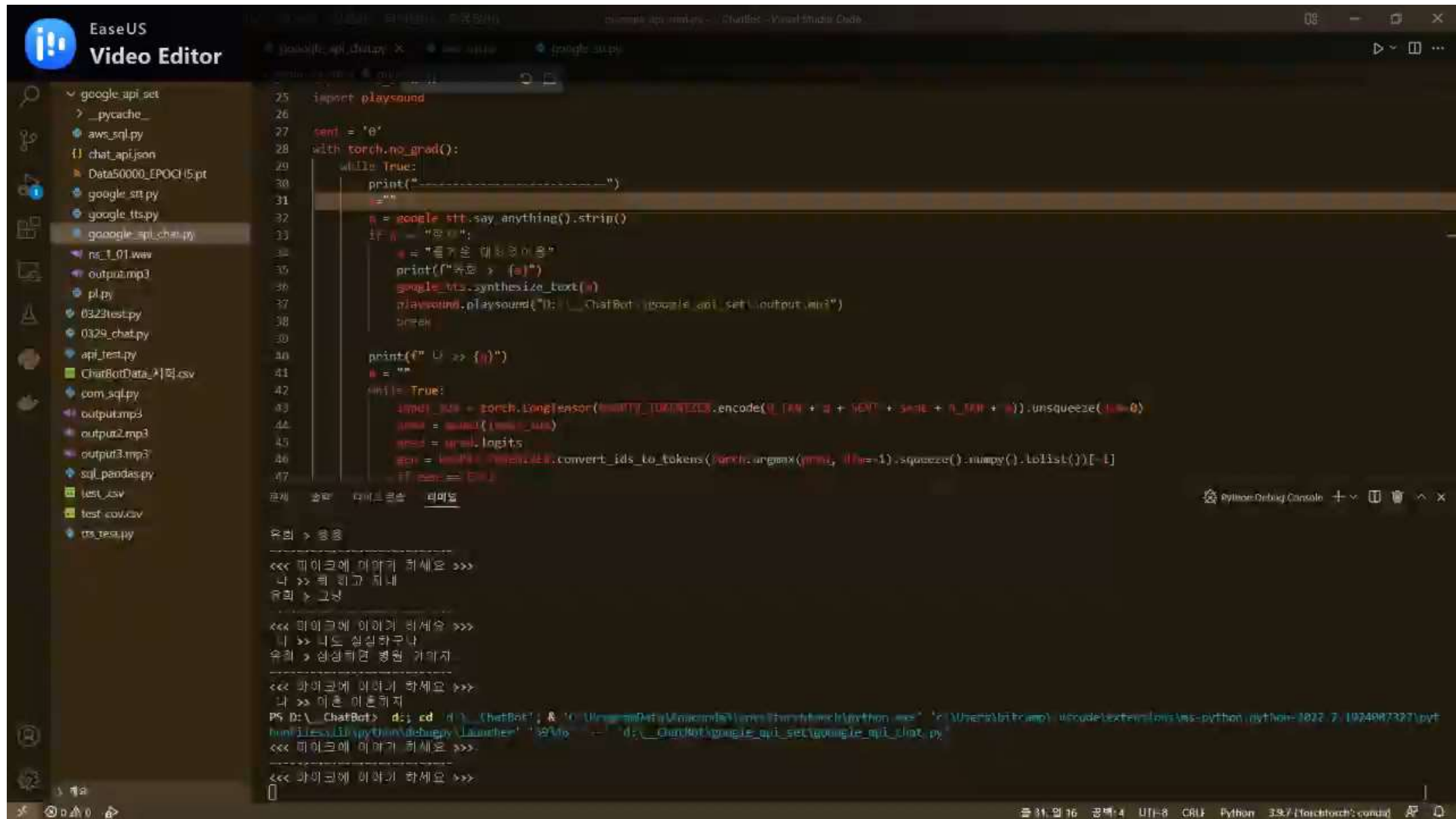
Syntax:



시연 예시



시연 영상 stt- tts



The screenshot shows a video editor interface with a Python script for a chatbot. The script uses the Google Cloud Text-to-Speech (TTS) and Speech-to-Text (STT) APIs. It includes a loop that continuously listens for audio input, processes it with the STT API, and then generates a response using the TTS API. The response is then played back using the playsound module.

```
25 import playsound
26
27 sent = '0'
28 with torch.no_grad():
29     while True:
30         print("-----")
31         s = ""
32         s = google_stt.say_anything().strip()
33         if s == "종료":
34             s = "듣기은 대치동 이용"
35         print(f"주인 > {s}")
36         google_tts.synthesize_text(s)
37         playsound.playsound(f"D:\_ChatBot\google_api_set\output.mp3")
38         break
39
40 print(f"나 >> {q}")
41 s = ""
42 while True:
43     inputs_ids = torch.LongTensor([tokenizer.encode(q + "[CLS] + q + [EOS] + s + s + [CLS] + q)]).unsqueeze(-1)
44     y_hat = model(inputs_ids)
45     probs = nn.functional.softmax(logits, dim=-1)
46     token_id = torch.argmax(probs, dim=-1).squeeze().numpy().tolist()[0]
47     if token_id == [EOS]:
48         break
```

The video editor interface shows a list of files on the left, including audio files like 'na_1_01.wav' and 'output.mp3'. The main area displays the Python script, and the bottom area shows the command prompt output, which includes the chatbot's responses to user input.

유리 > 종료

<<< 마이크에 이야기 하세요 >>>

나 >> 이 하고 되네

유리 > 그냥

<<< 마이크에 이야기 하세요 >>>

나 >> 너도 실실하구나

유리 > 상상하면 병원 가의자

<<< 마이크에 이야기 하세요 >>>

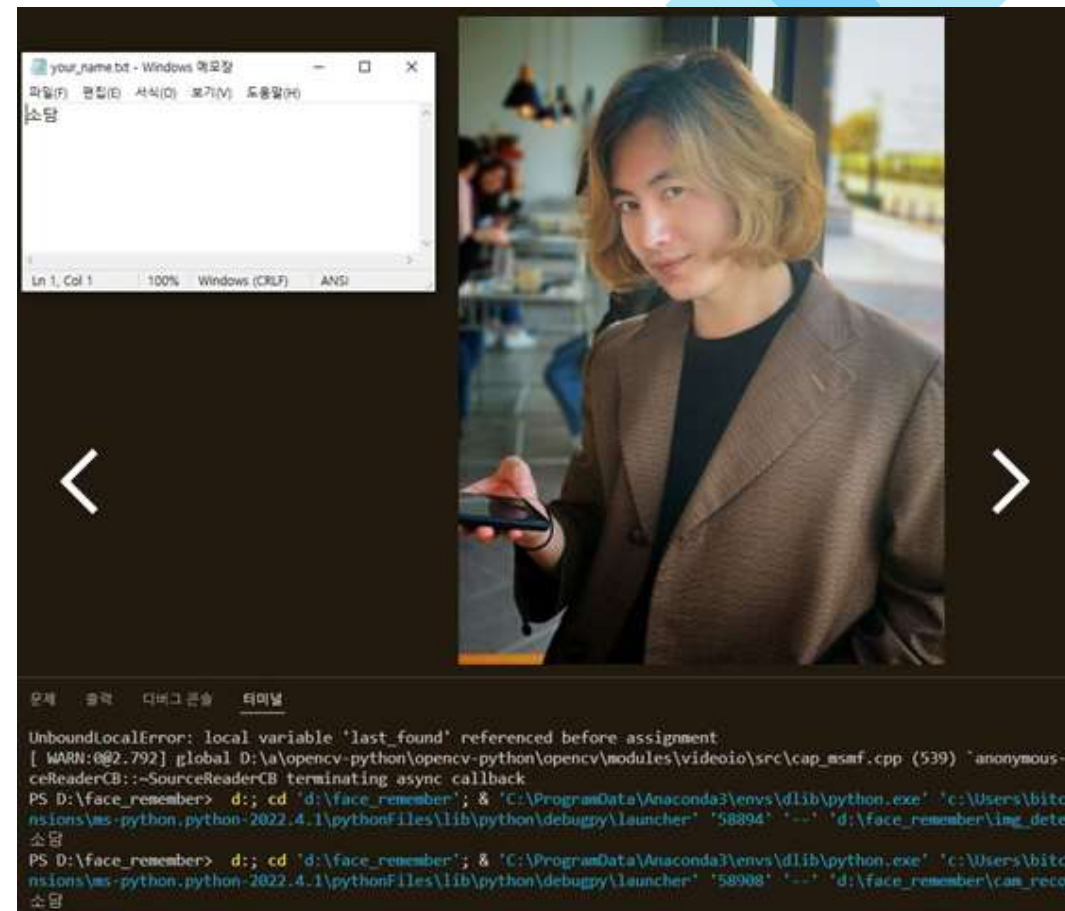
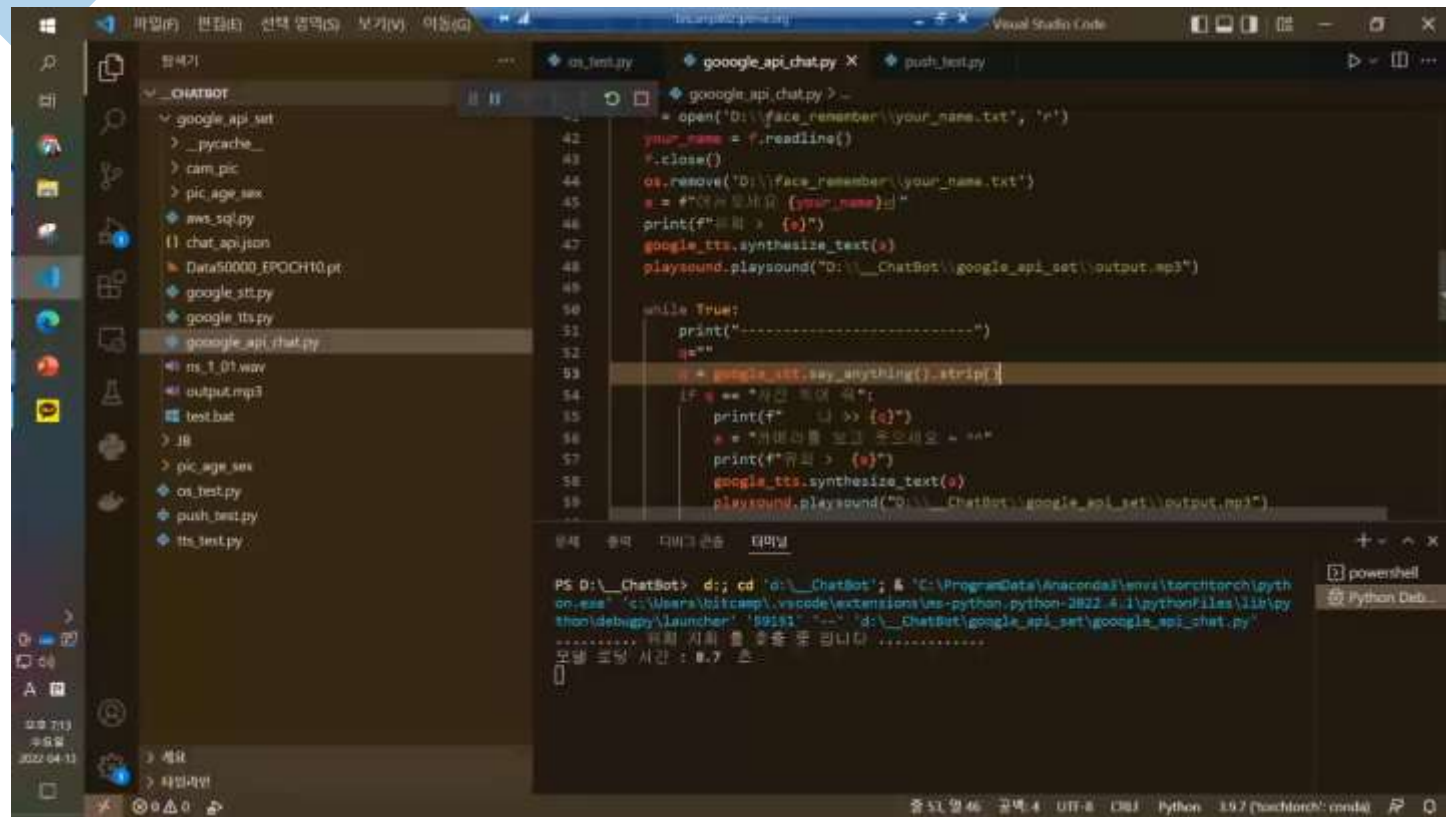
나 >> 이혼 이혼하지

PS D:_ChatBot> cd d:_ChatBot; & "C:\Users\mdata\Miniconda3\envs\torch\python.exe" -c "Msvcrt\bitcramp\mcode\external\aws\python\python-3.9.7-1924987227\pyt\non\libs\libpython\debugpy\launcher" -s 39Mo -- "d:_ChatBot\google_api_set\google_api_chat.py"

<<< 마이크에 이야기 하세요 >>>

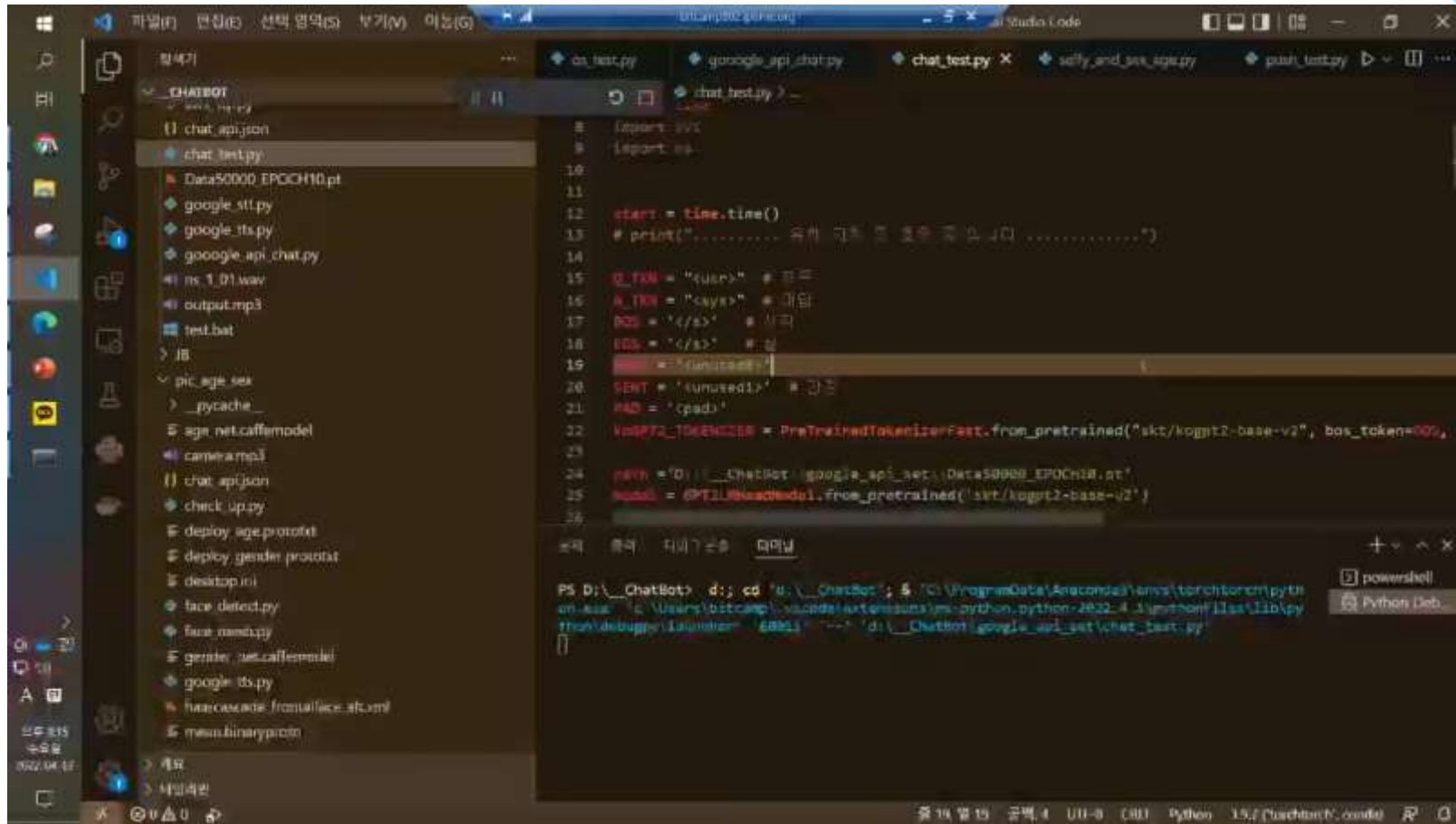
<<< 마이크에 이야기 하세요 >>>

시연 영상



얼굴인식

시연 영상



나이 & 성별

시연 결과



사진 분석

시연 결과

```
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 롤리 ?
유희 > 아
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 아 여보는 학습이 안 됐구나
유희 > 아니 괜찮아
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 너와 나의 연결 고리
유희 > 연결고리가 뭔지 모르겠어
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 우리 무슨 사이야
유희 > 내가 아니라 너님이랑 사이좋게 지내
```

Epoch = 5

```
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 오빠는 어때
| 답변 생성중 |||||
유희 >> 어 난 좀 그래
Answer gen time : 2.22 sec
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 소담이 오빠는 어때
| 답변 생성중 |||||
유희 >> 유부남 아냐
Answer gen time : 2.33 sec
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 여자 친구 남자 친구
| 답변 생성중 |||||
유희 >> 소개팅하고 실영
Answer gen time : 2.57 sec
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 소개팅 시켜 줄 거야
| 답변 생성중 |||||
유희 >> 소개팅 너는 연하 연상 동갑중에 뭐가 좋아
```

Epoch = 10

```
The class this function is called from is 'PreTrainedTokenizerFast'.
나 > 안녕 유희야
유희 > 응응 너 오늘 유난히 대답 잘 한다 라임 맞춰가면서
나 > 고마워 칭찬해줘서
유희 > 모든 일에 있어서 노력하는 모습 아주 칭찬해
나 > 어제 술을 너무 많이 마셨어
유희 > 어제 뭐 먹었는데
나 > 이자카야에서 나베랑 소주 마셔
유희 > 술 한잔 할 수 있어
나 > 나 술 좋아해
유희 > 무슨 술이야
나 > 주로 맥주를 마셔
유희 > 소주 좋아하는 사람이라면 다들 막 소주를 마셔
```

Epoch = 20

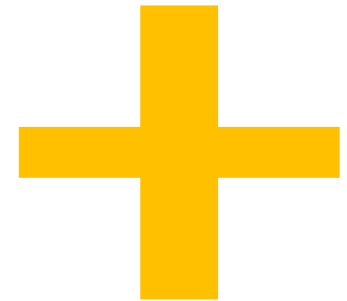
진행 예상 방향



“Hello~”

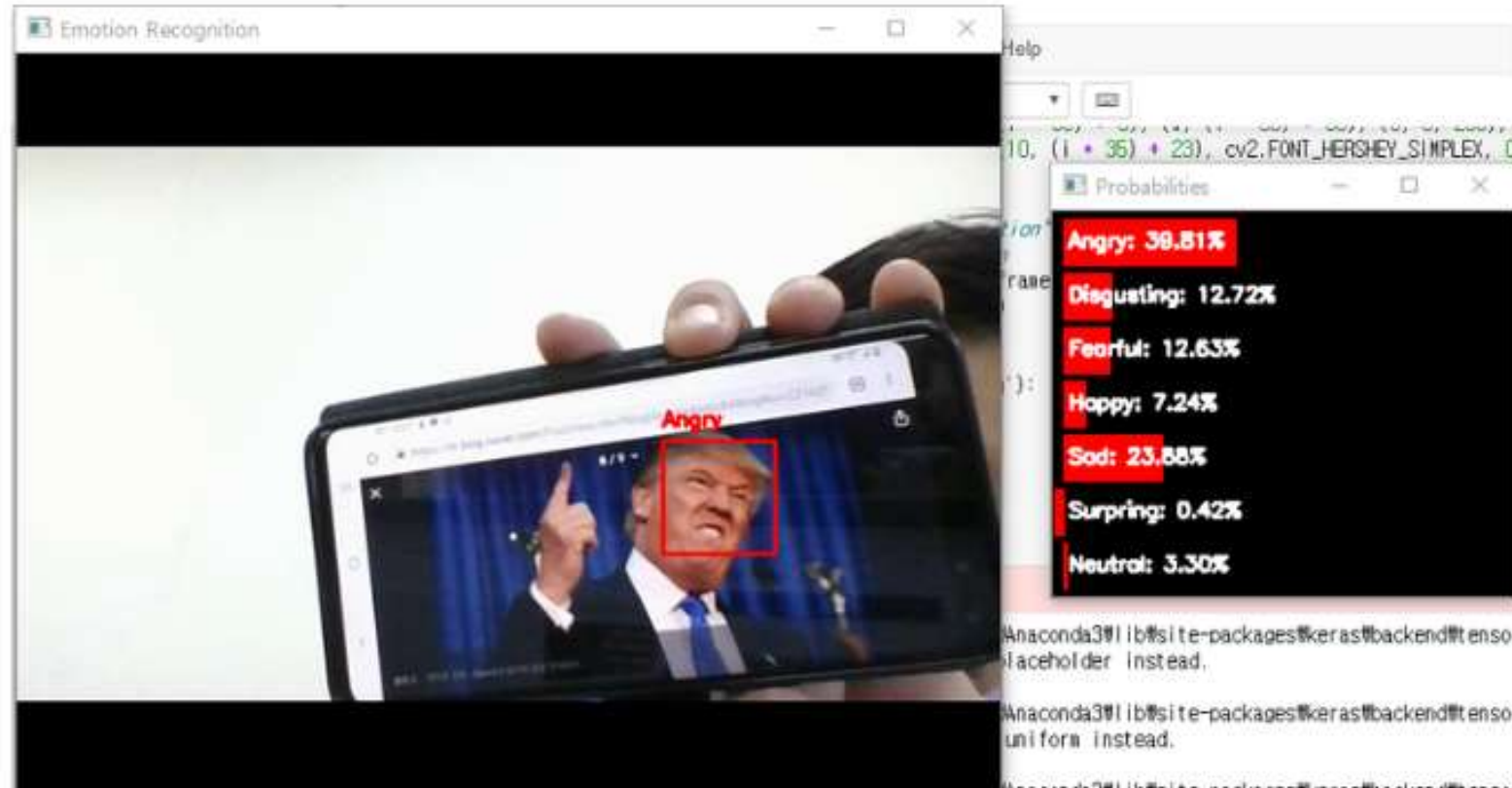
“20살 사이의 남자/여자다”

“사진에 그릇, 컵이 있네”

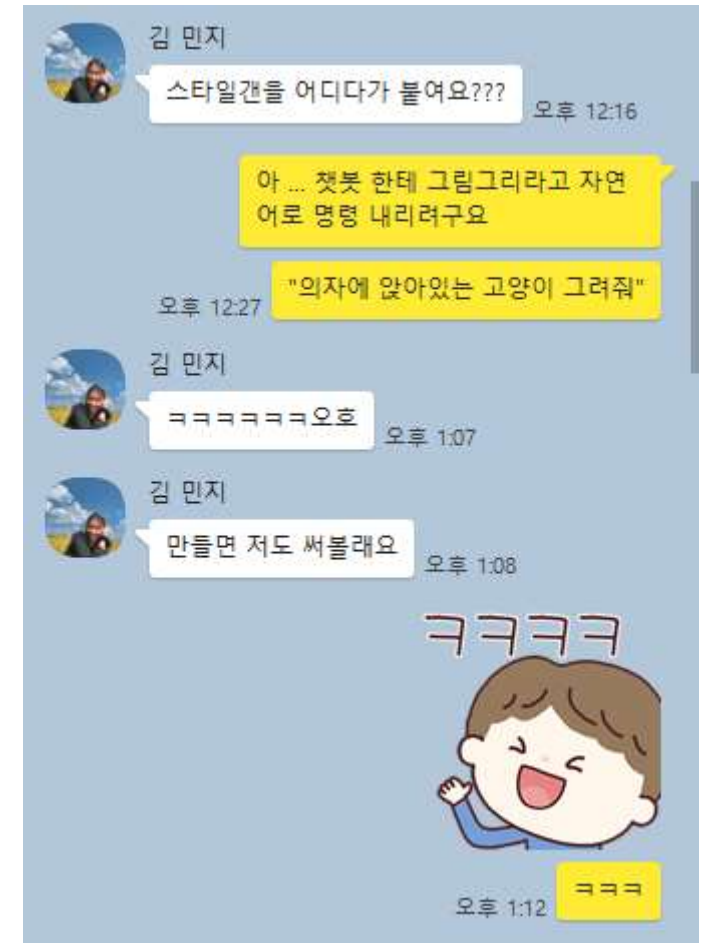


TTS로 연결

진행 예상 방향



진행 예상 방향



Q&A

