

「나의 AI편친」

자연어 처리 모델을 활용한 개인화 AI휴먼

Team 언어유희

Content

0 | 팀원 소개

1 | 프로젝트 진행 상황

Tech & Out Put 리뷰

2 | 진행 예정 프로세스 소개

3 | Q & A

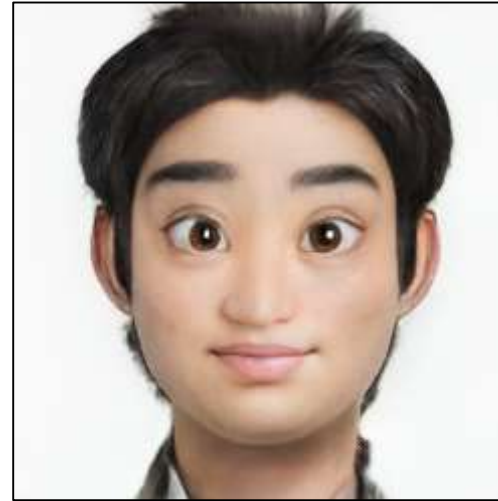
팀원 소개



팀장
장세종



이소담



이 한



남궁지희

나보다 나를 더 잘 아는

당신을 이해하고
당신에게 귀 기울이는 존재



“안녕 🖐️”

난 너의 AI핀친
유희지희야”

YH. Ji Hee

프로젝트 진행 상황





CONTENTS

CONTENTS	2022. 3					2022. 4				2022. 5	
	2월 4주	3월 1주	3월 2주	3월 3주	3월 4주	3월 5주	4월 1주	4월 2주	4월 3주	4월 4주	5월 1일
프로젝트 방향설정											
모델선택											
데이터수집											
논문리뷰											
모델구성											
튜닝											
테스트 및 개선											
최종점검											

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 어머니 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	5					만 내켜 혐오
30	NaN					왜? 중립
31	NaN					뭔가 알리는 기분이야. 불길해 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? 자가 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	5					집 짓는 이유가 그렇게 중 해? 왜, 내가 사기라도 칠까봐? 혐오
42	NaN					또. 험악하게. 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 '[O1 T1]' 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
```


시연 결과

```
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 롤리 ?
유희 > 아
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 아 여보는 학습이 안 됐구나
유희 > 이나 괜찮아
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 너와 나의 연결 고리
유희 > 연결고리가 뭘지 모르겠어
-----
<<< 마이크에 이야기 하세요 >>>
나 >> 우리 부는 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 >> 유튜브남 아니야
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 통갑종에 뭐가 조아

```

Epoch = 10

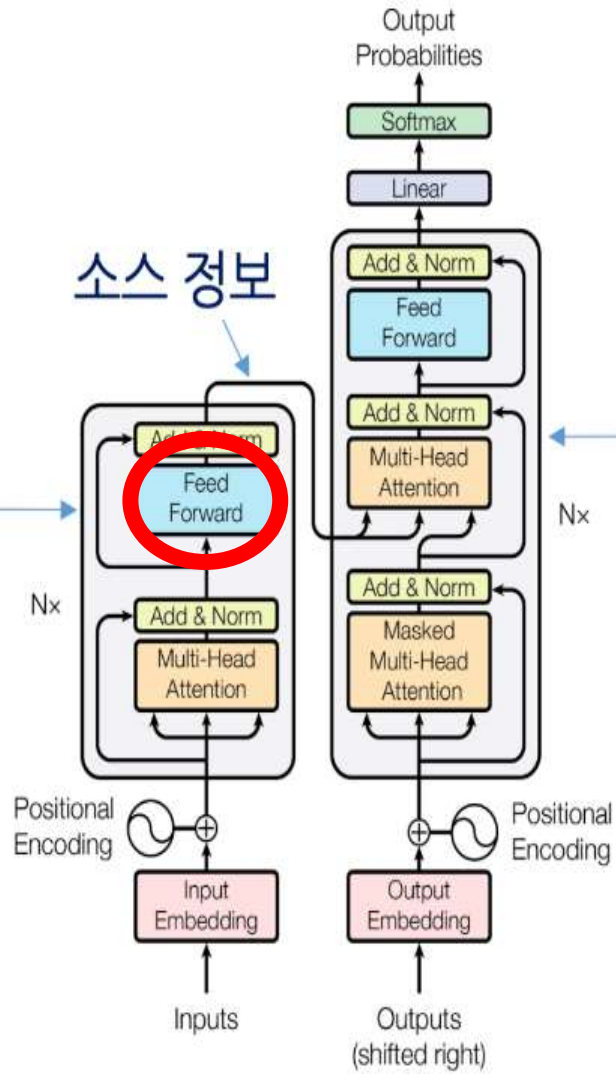
```
The class this function is called from is 'PreTrainedTokenizerFast'.
나 > 안녕 유희야
유희 > 응응 너 오늘 유난히 대답 잘 한다 라임 맞춰가면서
나 > 고마워 칭찬해줘서
유희 > 모든 일에 있어서 노력하는 모습 아주 칭찬해
나 > 어제 술을 너무 많이 마셨어
유희 > 어제 뭐 먹었는데
나 > 이자카야에서 나베랑 소주 마셔
유희 > 술 한잔 할 수 있어
나 > 나 술 좋아해
유희 > 무슨 술이야
나 > 주로 맥주를 마셔
유희 > 소주 좋아하는 사람이라면 다들 막 소주를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

```

Epoch = 20

트랜스포머

Encoder
소스 시퀀스 압축



Decoder
타겟 시퀀스 생성

Feed Forward

- 2개의 Dense Layer와 Relu 함수로 구성

Add & Norm

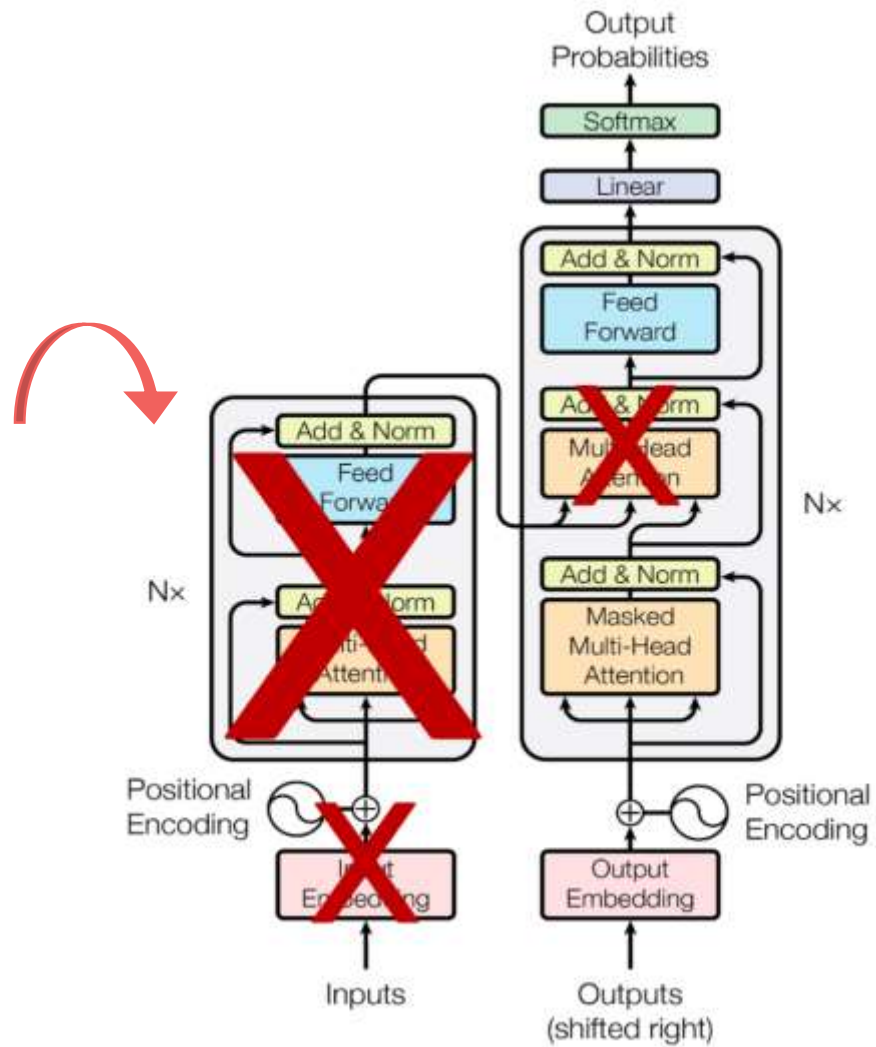
- 레이어 정규화, 잔차연결 역할

Positional Encoding

문장의 의미 파악을 위해
문장의 단어 위치정보가 중요

KoGPT2 (Korean Generative Pre-trained Transformer)

GPT-2 모델을 Fine-Tuning한 한국어 언어모델



- 트랜스포머 Decoder 해당

- 적은 데이터로도 뛰어난 성능

- 문장 생성에 뛰어난 성능

모델과 토큰라이저

```
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)
```

```
train_set = ChatbotDataset(Chatbot_Data, max_len=40)
```

```
train_dataloader = DataLoader(train_set, batch_size=32, num_workers=0, shuffle=True,
                               collate_fn=collate_batch,)
```

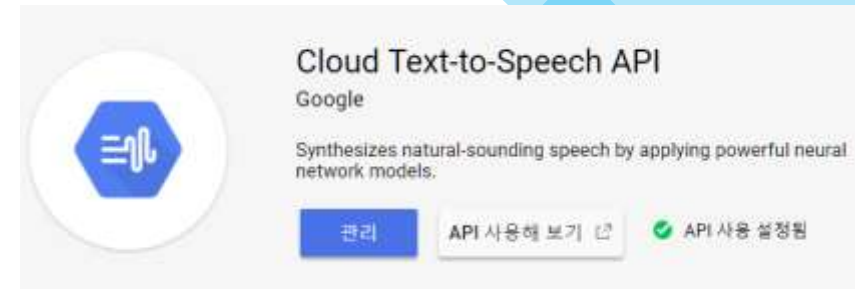
파인 튜닝

```
learning_rate = 3e-5
criterion = torch.nn.CrossEntropyLoss(reduction="none")
optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate)

epoch = 10 # 학습 횟수
Sneg = -1e18 # 오차가 음수가 되는 것을 방지하기 위한 최소값

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")
```


STT - TTS





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



```
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
```



STT / TTS

```
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 = "종료된 채팅을 종료합니다"
35             print(f"종료 > {s}")
36             google_tts.synthesize_text(s)
37             playsound.playsound(f"{__ChatBot__}\\google_api_set\\output.mp3")
38             break
39
40         print(f"U >> {s}")
41         s = ""
42         while True:
43             input_ids = torch.LongTensor([tokenizer.encode([U_TOKEN + s + SENT + SENT + U_TOKEN + s])).unsqueeze(-1)
44             output = model(input_ids)
45             output = torch.nn.functional.softmax(output, dim=-1).squeeze().numpy().tolist()[0][1:]
46             token = tokenizer.convert_ids_to_tokens(output)
47             if token == END:
```

Python: Debug Console

```
<<< 파이썬에 마하가 하세요 >>>
나 >> 뭐 하고 있네
유희 > 그냥

<<< 마이크에 이어가 하세요 >>>
나 >> 나는 실실하구나
유희 > 상상하면 병원 가리지

<<< 마이크에 이어가 하세요 >>>
나 >> 이혼 이혼하지

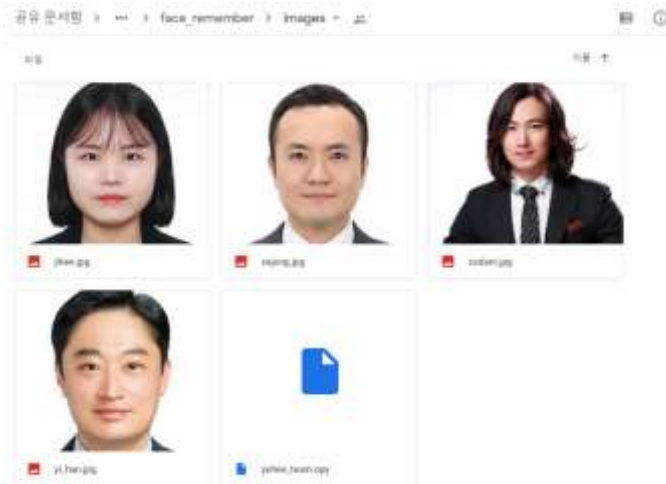
PS D:\_ChatBot> cd .\ChatBot; & "C:\Users\bitcamp\OneDrive\Documents\python\ms" "C:\Users\bitcamp\OneDrive\Documents\python\python-3.9.7\python-3.9.7-1924987327\python-3.9.7\python\python\launcher" "39Mo" -- "d:\_ChatBot\google_api_set\google_api_chat.py"

<<< 파이썬에 마하가 하세요 >>>

<<< 마이크에 이어가 하세요 >>>
```



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()
```

```
# 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)
```

```
    descs[name] = encode_faces(img_rgb, img_shapes)[0]
```

```
    np.save('images/yuhee_team.npy', descs)
```

```
print(descs)
```

```
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.dat')
```

```
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)
```

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)

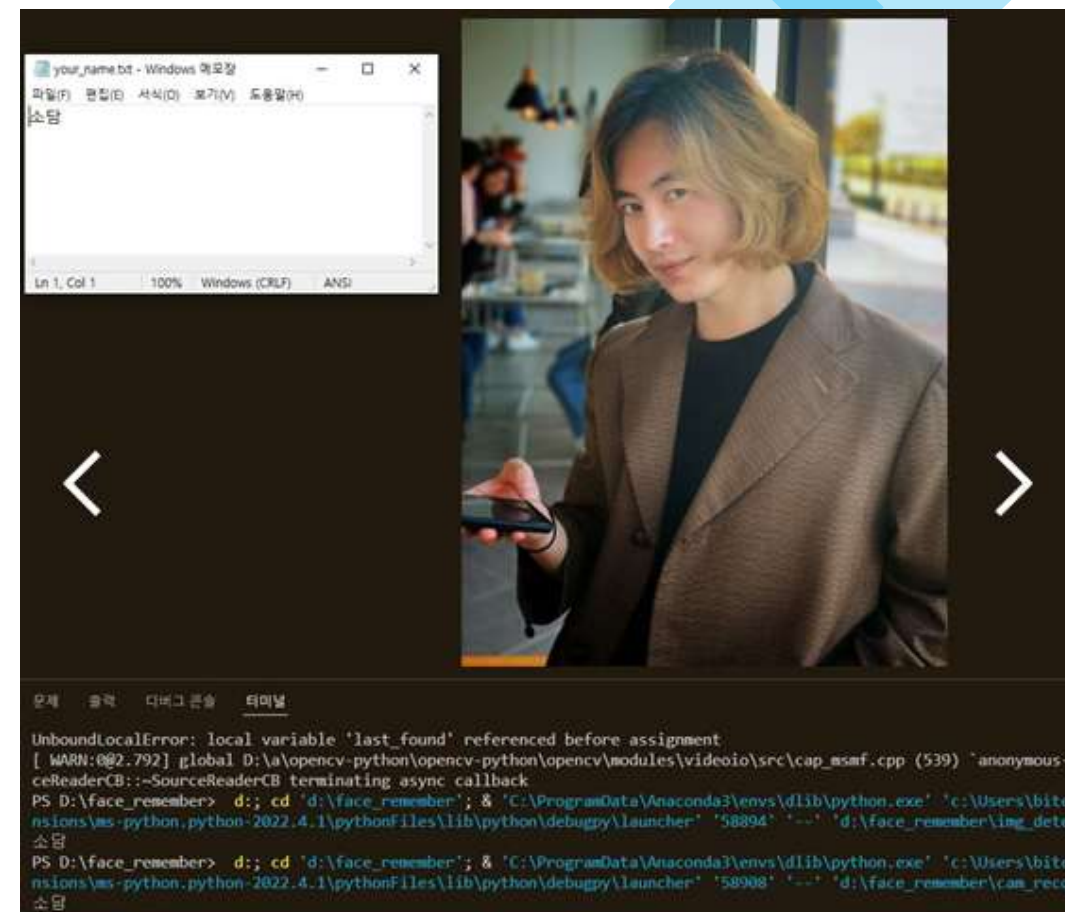
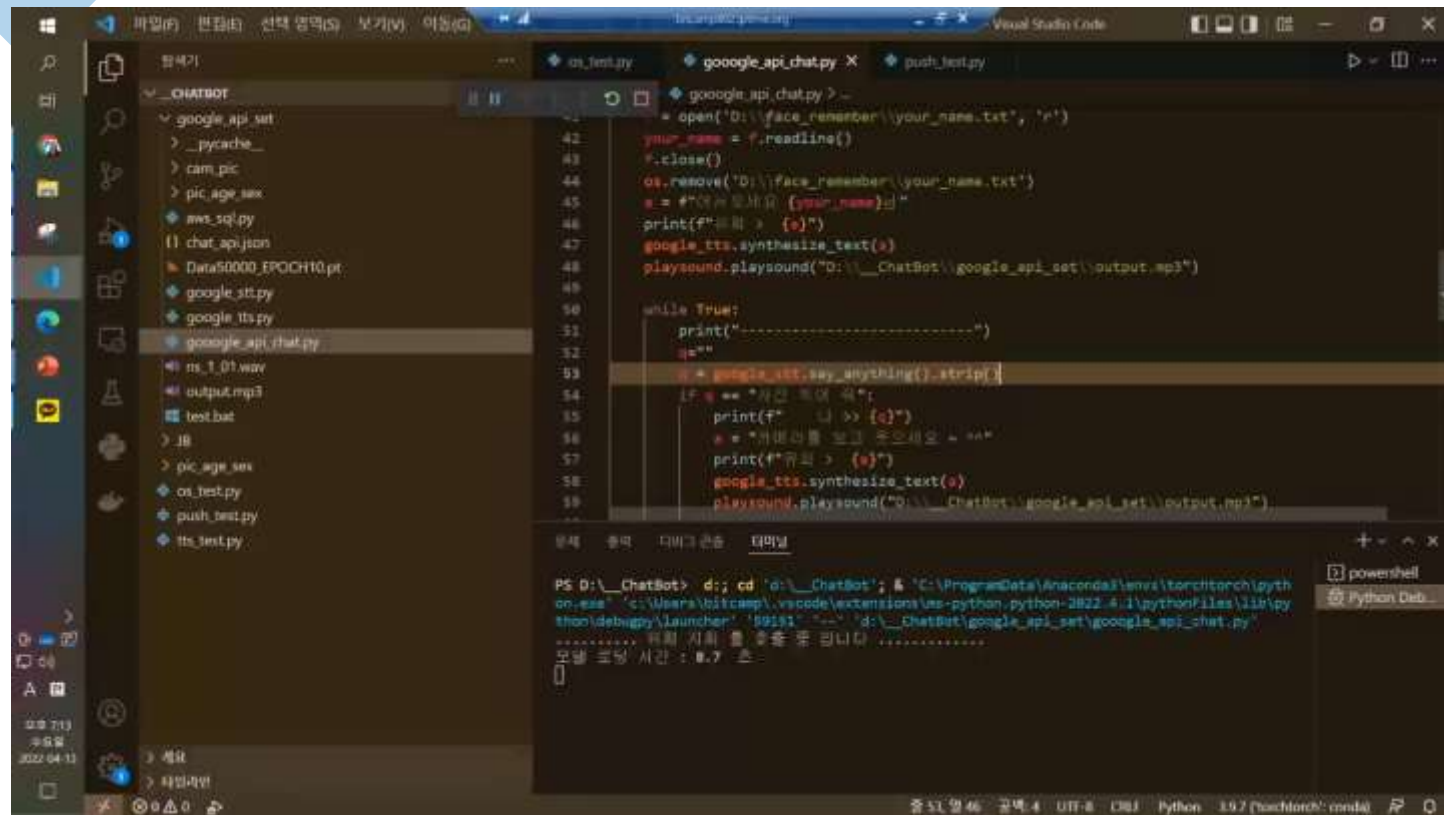
    for k, d in enumerate(dets):
        shape = sp(img_rgb, d)
        face_descriptor = facerec.compute_face_descriptor(img_rgb, shape)
        last_found = {'name': 'unknown', 'dist': 0.6, 'color': (0, 0, 255)}

        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
```





얼굴인식



Face detection

```
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']
```

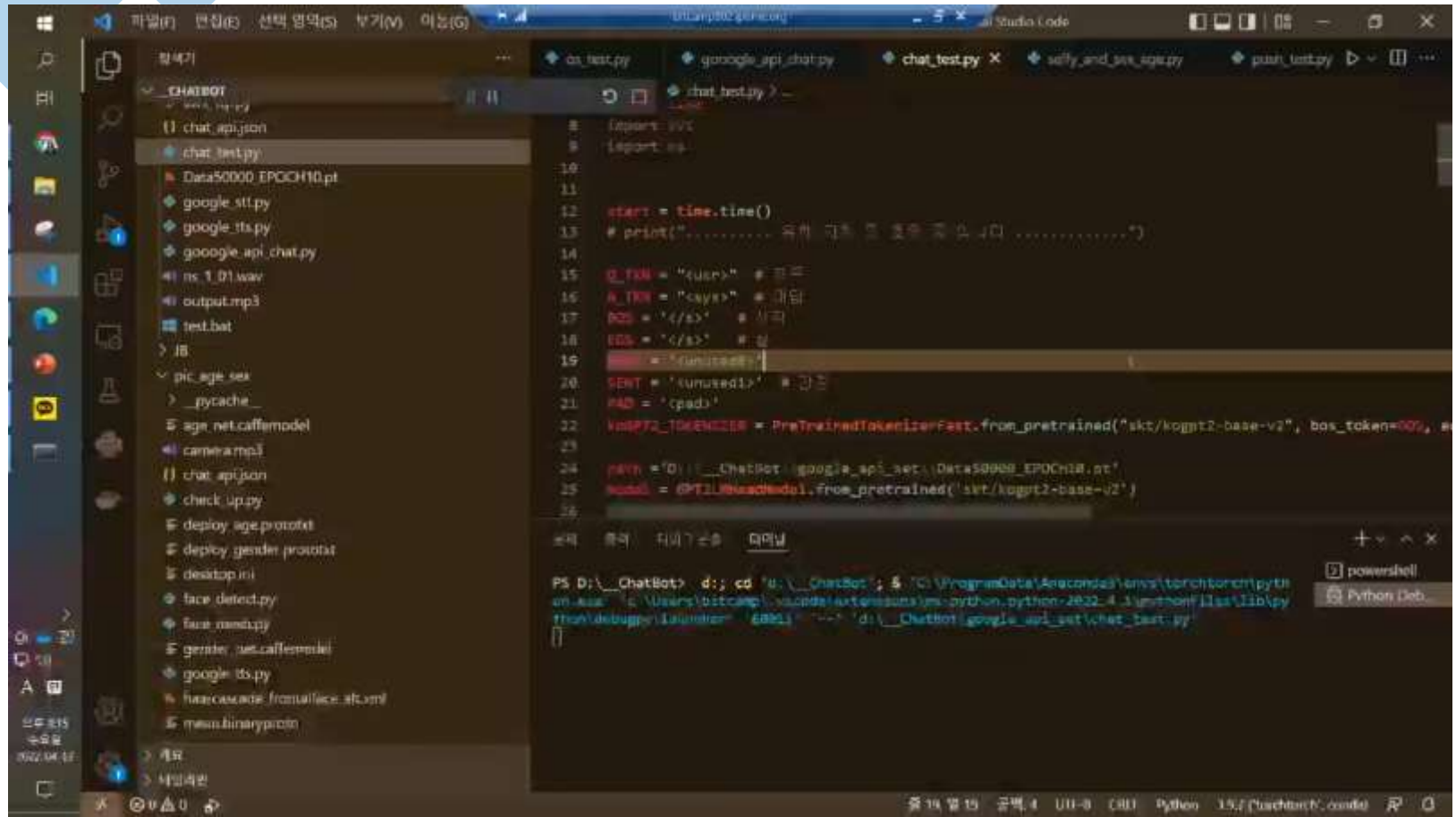
```
# gender detection
gender_net.setInput(blob)
gender_preds = gender_net.forward()
gender = gender_preds.argmax()

# 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



나이 & 성별



Object detection

```
# 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



Object detection

```
labels = []
for i in range(len(bboxes)):
    if i in indexes:
        x, y, w, h = bboxes[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)
```

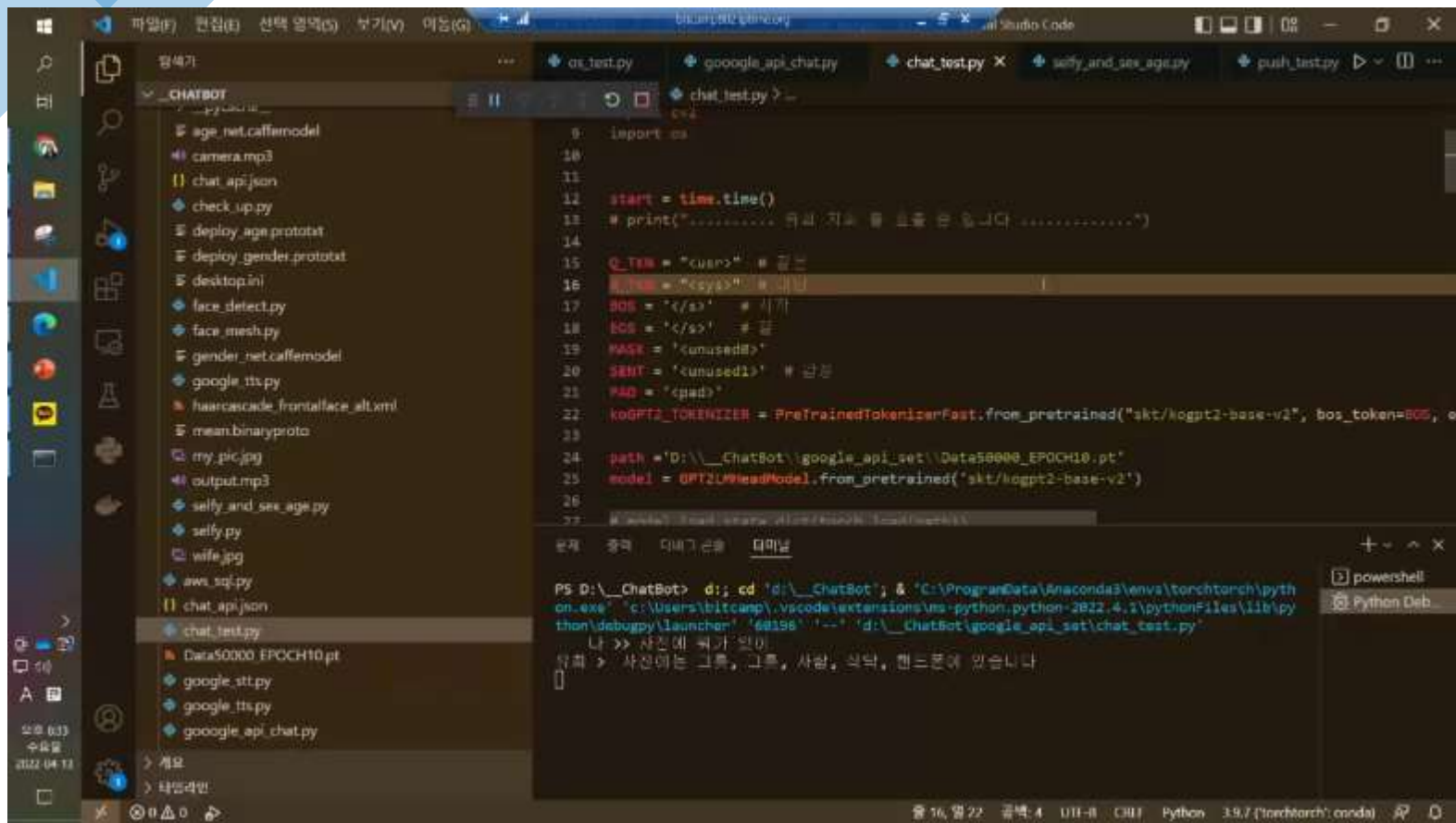


사진 분석

▶ 진행 예정 프로세스 소개



유희지희(챗봇) 스킨 구현



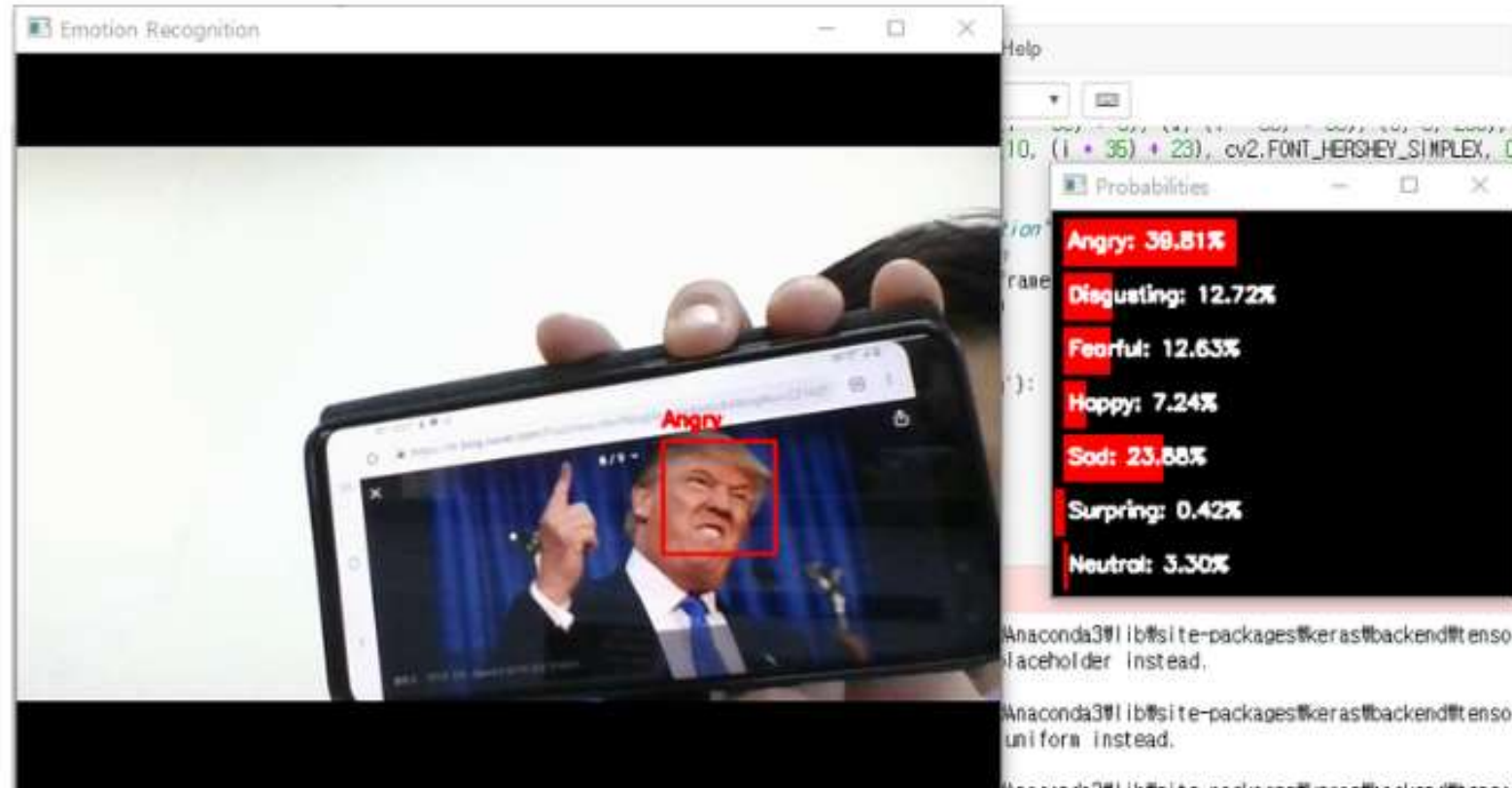
애니메이션화 해줘



진행 예정



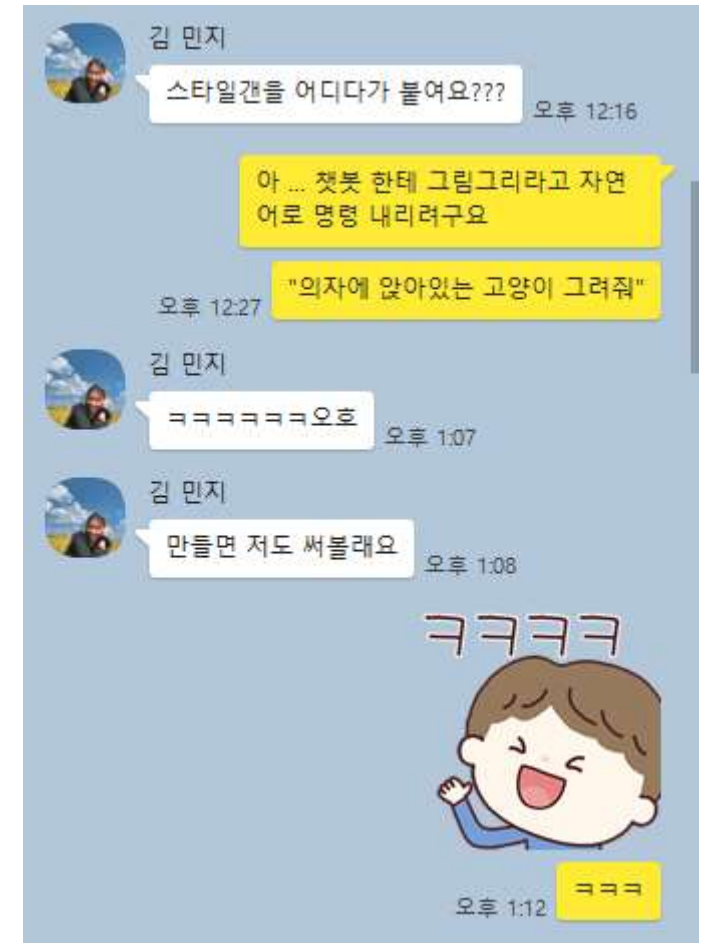
나 오늘 어때 보여



진행 예정



언덕위에 앉아있는
고양이 **그림 그려줘**





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&A

