이미지 및 자연어 처리 기반 감정분석 챗봇

세나봇

김가영 문유진 장정윤 황정현

목차

프로젝트목적 및 의도

<이미지 및 자연어처리기반 감정 분석 챗봇>

감정분석을통해 사용자들의마음을 알아주고그 감정에 맞는
음악을 추천해주는만족도 높은 맞춤형 챗봇 서비스 제공

역할 분담

김가영

팀장

텍스트를통한 감정분석

문유진

얼굴인식을통한 감정분석

장정윤

얼굴인식을통한 감정분석

황정현

텍스트를통한 감정분석

사용 기술 스택

Python

사용 언어 o

풍부한 머신러닝 라이브러리와 프레임워크를 가진 프로그래밍 언어

Google Colab

개발 환경 o

웹 브라우저에서 파이썬 코드를 작성 및 실행할 수 있는 온라인 에디터

Kakao i Open Builder

설계 플랫폼 o

AI 핵심 기술 결합 카카오의 통합 인공지능 플랫폼

Flask

프레임 워크 o

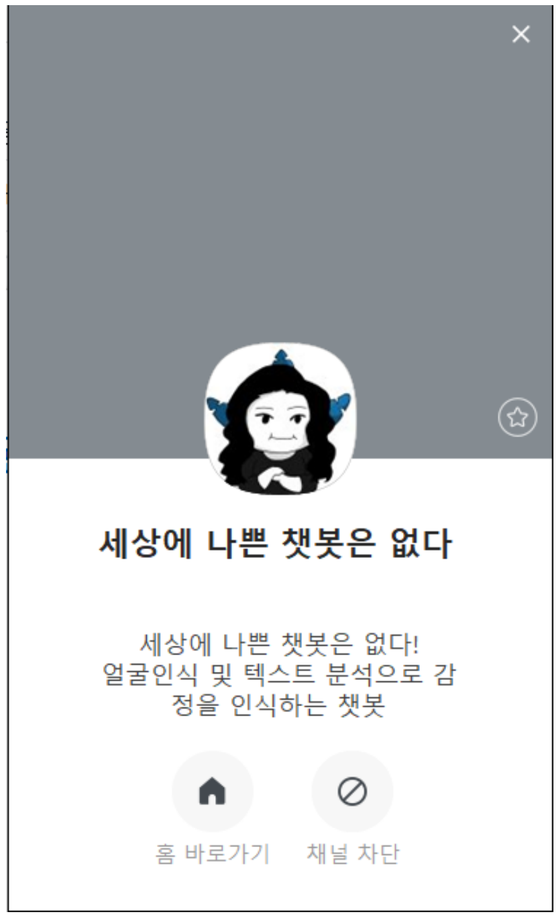
웹 애플리케이션 개발을 위한 파이썬 프레임워크

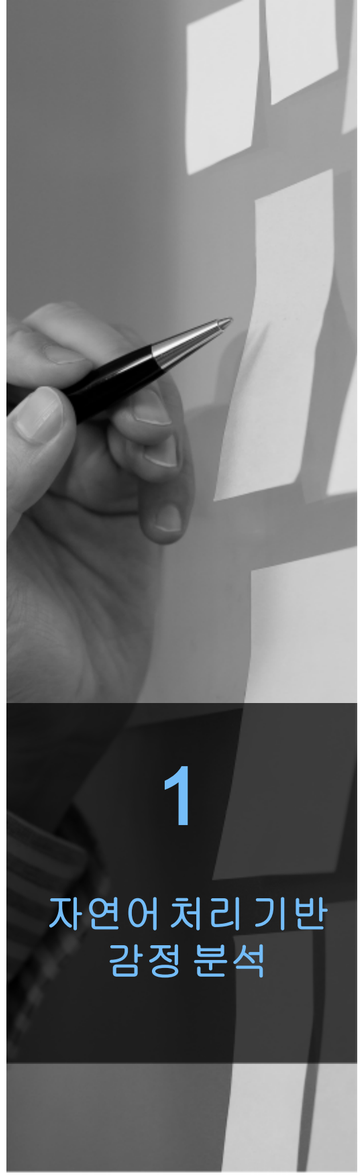
goormide

챗봇 스킬 o

웹 기반의 클라우드 코딩 서비스로 서버 구축을 위해 사용

구현된 기능 소개



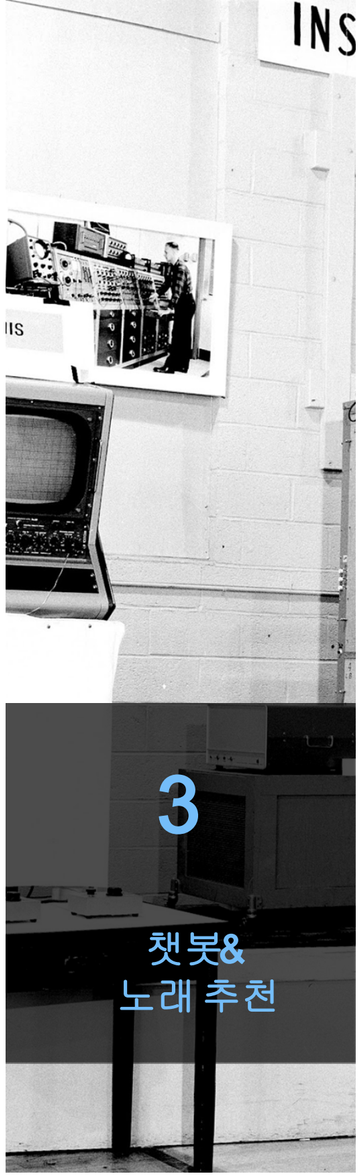


1

자연어 처리 기반
감정 분석

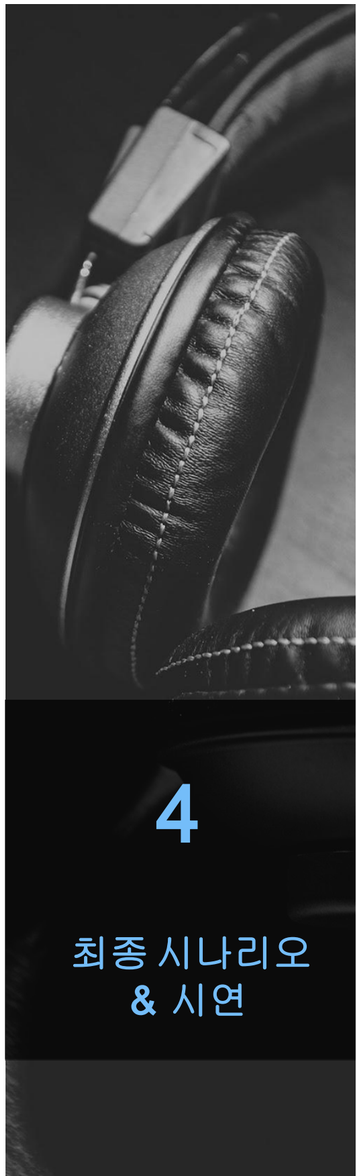
2

얼굴인식 기반
감정 분석



3

챗봇&
노래 추천



4

최종 시나리오
& 시연

구현된 기능 소개

● 자연어 처리 기반 감정 분석

예시)

“어제 공포영화를 봤더니 혼자 못자겠어!”	“아니, 과제 제출일이 내일이라고?”	“한 번만 더 에러창이 뜨면 부숴버릴거야.”	“교수님은 나한테 왜 이렇게 어려운 과제를 주시는 걸까..”	“이번 학기는 왠지 느낌이 좋은 걸~”
공포	놀람	분노	슬픔	행복

- ❏ Ai 허브의 ‘한국어 단발성 대화 데이터셋’을 가공하여 학습 데이터 제작
- ❏ 분석 정확도 60% -> 75%
- ❏ 공포, 놀람, 분노, 슬픔, 행복 다섯 가지 클래스의 감정 분류 가능

구현된 기능 소개

● 자연어 처리 기반 감정 분석 시나리오

구현된 기능 소개

● 자연어 처리 기반 감정 분석 코드

```
class BERTDataset(Dataset):
    def __init__(self, dataset, sent_idx, label_idx, bert_tokenizer, max_len,
                 pad, pair):
        transform = nlp.data.BERTSentenceTransform(
            bert_tokenizer, max_seq_length=max_len, pad=pad, pair=pair)

        self.sentences = [transform([i[sent_idx]]) for i in dataset]
        self.labels = [np.int32(i[label_idx]) for i in dataset]

    def __getitem__(self, i):
        return (self.sentences[i] + (self.labels[i], ))

    def __len__(self):
        return (len(self.labels))
```

```
tokenizer = get_tokenizer()
tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)

data_train = BERTDataset(dataset_train, 0, 1, tok, max_len, True, False)
data_test = BERTDataset(dataset_test, 0, 1, tok, max_len, True, False)
```

```
using cached model. /content/.cache/kobert_news_wiki_ko_cased-1087f8699e.spiece
```

```
[13] train_dataloader = torch.utils.data.DataLoader(data_train, batch_size=batch_size, num_workers=5)
     test_dataloader = torch.utils.data.DataLoader(data_test, batch_size=batch_size, num_workers=5)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader
  cpuset_checked))
```

구현된 기능 소개

● 자연어 처리 기반 감정 분석 코드

```
class BERTClassifier(nn.Module):
    def __init__(self,
                 bert,
                 hidden_size = 768,
                 num_classes=5,
                 dr_rate=None,
                 params=None):
        super(BERTClassifier, self).__init__()
        self.bert = bert
        self.dr_rate = dr_rate
                 |
        self.classifier = nn.Linear(hidden_size , num_classes)
        if dr_rate:
            self.dropout = nn.Dropout(p=dr_rate)

    def gen_attention_mask(self, token_ids, valid_length):
        attention_mask = torch.zeros_like(token_ids)
        for i, v in enumerate(valid_length):
            attention_mask[i][:v] = 1
        return attention_mask.float()

    def forward(self, token_ids, valid_length, segment_ids):
        attention_mask = self.gen_attention_mask(token_ids, valid_length)

        _, pooler = self.bert(input_ids = token_ids, token_type_ids = segment_ids.long(), attention_mask = attention_mask.float().to(token_ids.device))
        if self.dr_rate:
            out = self.dropout(pooler)
        return self.classifier(out)
```

구현된 기능 소개

● 자연어 처리 기반 감정 분석 코드

```
model = BERTClassifier(bertmodel,  dr_rate=0.5).to(device)

no_decay = ['bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
    {'params': [p for n, p in model.named_parameters() if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
    {'params': [p for n, p in model.named_parameters() if any(nd in n for nd in no_decay)], 'weight_decay': 0.0}
]

optimizer = AdamW(optimizer_grouped_parameters, lr=learning_rate)
loss_fn = nn.CrossEntropyLoss()

t_total = len(train_dataloader) * num_epochs
warmup_step = int(t_total * warmup_ratio)

scheduler = get_cosine_schedule_with_warmup(optimizer, num_warmup_steps=warmup_step, num_training_steps=t_total)

def calc_accuracy(X,Y):
    max_vals, max_indices = torch.max(X, 1)
    train_acc = (max_indices == Y).sum().data.cpu().numpy()/max_indices.size()[0]
    return train_acc

train_dataloader
```

```
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:309: FutureWarning: This implementation of AdamW is depr
  FutureWarning,
<torch.utils.data.dataloader.DataLoader at 0x7f7596a6d290>
```

```
print(test_acc)
```

```
75.91335227272727
```

구현된 기능 소개

● 자연어 처리 기반 감정 분석 코드

```
[25] tokenizer = get_tokenizer()
     tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)

     def predict(predict_sentence):

         data = [predict_sentence, '0']
         dataset_another = [data]

         another_test = BERTDataset(dataset_another, 0, 1, tok, max_len, True, False)
         test_dataloader = torch.utils.data.DataLoader(another_test, batch_size=batch_size, num_workers=5)

         model.eval()

         for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(test_dataloader):
             token_ids = token_ids.long().to(device)
             segment_ids = segment_ids.long().to(device)

             valid_length= valid_length
             label = label.long().to(device)

             out = model(token_ids, valid_length, segment_ids)

             test_eval=[]
             for i in out:
                 logits=i
                 logits = logits.detach().cpu().numpy()

                 if np.argmax(logits) == 0:
                     test_eval.append("공포가")
                 elif np.argmax(logits) == 1:
                     test_eval.append("놀람이")
                 elif np.argmax(logits) == 2:
                     test_eval.append("분노가")
                 elif np.argmax(logits) == 3:
                     test_eval.append("슬픔이")
                 elif np.argmax(logits) == 4:
                     test_eval.append("행복이")

             print( test_eval[0] + " 느껴집니다.")

     using cached model. /content/.cache/kobert_news_wiki_ko_cased-1087f8699e.spiece
```

```
end = 1
while end == 1 :
    sentence = input("하고싶은 말을 입력해주세요 : ")
    if sentence == '끝' :
        break
    predict(sentence)
    print("₩n")
```

```
... 하고싶은 말을 입력해주세요 : 어제 공포영화를 봤더니 너무 무서워
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader
  cpuset_checked))
공포가 느껴집니다.


하고싶은 말을 입력해주세요 : 헐 내일이 과제 마감이라니
놀람이 느껴집니다.


하고싶은 말을 입력해주세요 : 한번만 더 에러창뜨면 부숴버릴거야
분노가 느껴집니다.


하고싶은 말을 입력해주세요 : 과제가 너무너무 힘들다..
슬픔이 느껴집니다.


하고싶은 말을 입력해주세요 : 이번학기는 느낌이 좋아!
행복이 느껴집니다.
```

구현된 기능 소개

● 얼굴인식기반 감정 분석

예시)

공포

놀람

분노

슬픔

행복

- ❏ 학습 데이터 :
- ❏ 분석 정확도 : 50% -> 60%
- ❏ 공포, 놀람, 분노, 슬픔, 행복 등 클래스의 감정 분류 가능

구현된 기능 소개

● 얼굴인식기반 감정 분석 시나리오

01 데이터 수집

02 데이터 전처리 (이미지 shape 조정)

03 학습 모델 만들기

04 모델 학습시키기

05 실제로 감정 예측하기

구현된 기능 소개

● 얼굴인식 기반 감정 분석 코드

```
def my_model():
    model = Sequential()
    input_shape = (48,48,1)
    model.add(Conv2D(64, (5, 5), input_shape=input_shape,activation='relu', padding='same'))
    model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(128, (5, 5),activation='relu',padding='same'))
    model.add(Conv2D(128, (5, 5),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Conv2D(256, (3, 3),activation='relu',padding='same'))
    model.add(Conv2D(256, (3, 3),activation='relu',padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))

    model.add(Flatten())
    model.add(Dense(128))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(0.2))
    model.add(Dense(7))
    model.add(Activation('softmax'))

    model.compile(loss='categorical_crossentropy', metrics=['accuracy'],optimizer='adam')
    # UNCOMMENT THIS TO VIEW THE ARCHITECTURE
    #model.summary()

    return model
model=my_model()
model.summary()
```

```
[ ]  test_true = np.argmax(y_test, axis=1)
     test_pred = np.argmax(model.predict(X_test), axis=1)
     print("CNN Model Accuracy on test set: {:.4f}".format(accuracy_score(test_true, test_pred)))
```

```
CNN Model Accuracy on test set: 0.5832
```

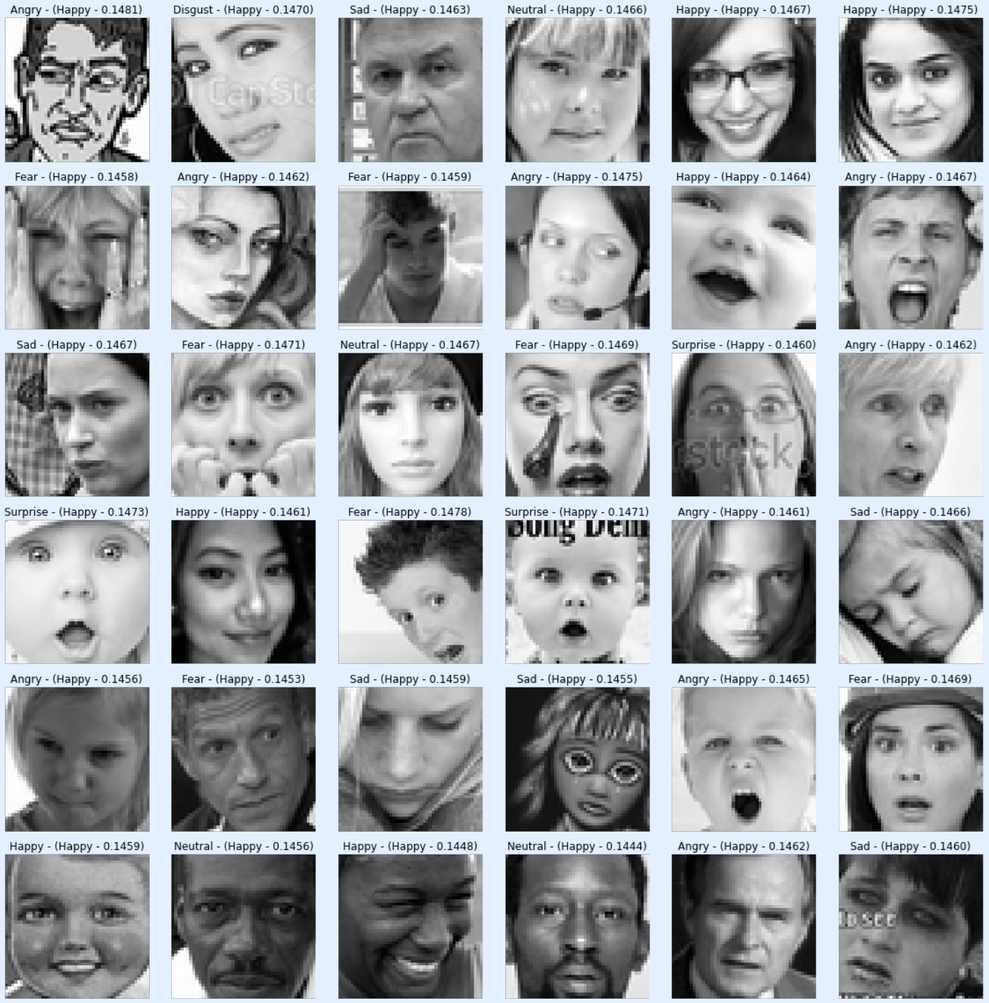
구현된 기능 소개

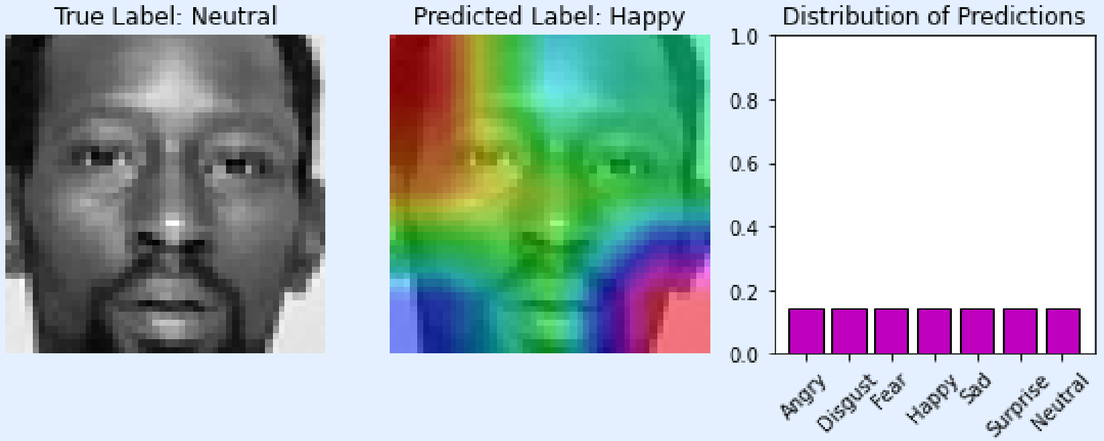
● train 데이터

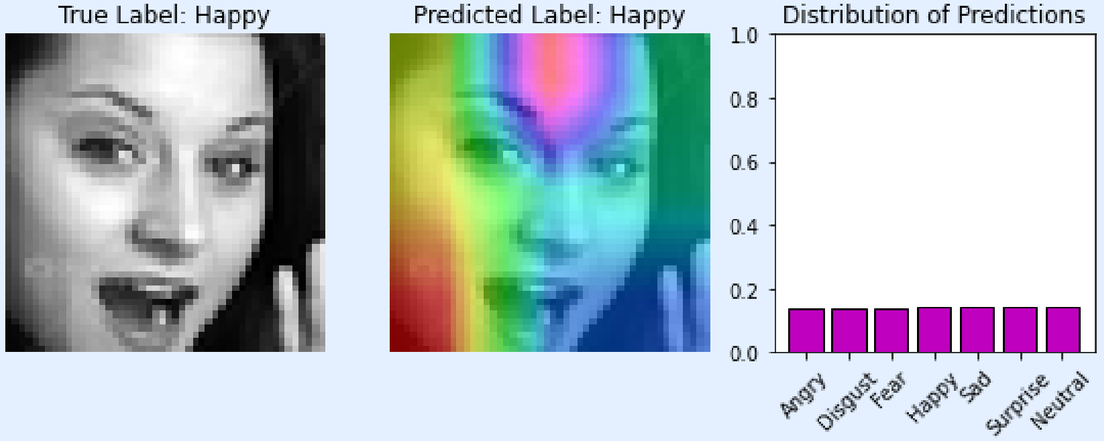
```
fig = plt.figure(1, (20, 20))

k = 0
for label in sorted(df['emotion'].unique()):
    for j in range(7):
        px = df[df['emotion']==label].pixels.iloc[k]
        px = np.array(px.split(' ')).reshape(48, 48).astype('float32')

        k += 1
        ax = plt.subplot(7, 7, k)
        ax.imshow(px, cmap='gray')
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_title(emotions[label])
        plt.tight_layout()
```

구현된 기능 소개

● train 데이터로 훈련 시킨 결과 - test 데이터







구현된 기능 소개

● 얼굴인식기반 감정 분석 코드

```
from skimage import io
img = image.load_img('/content/drive/MyDrive/data/an2.jpeg', grayscale=True, target_size=(48, 48))
show_img=image.load_img('/content/drive/MyDrive/data/an2.jpeg', grayscale=False, target_size=(200, 200))
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)

x /= 255

custom = model.predict(x)
#print(custom[0])
emotion_analysis(custom[0])

x = np.array(x, 'float32')
x = x.reshape([48, 48]);

plt.gray()
plt.imshow(show_img)
plt.show()

m=0.00000000000000000001
a=custom[0]
```

```
for i in range(0,len(a)):
    if a[i]>m:
        m=a[i]
        ind=i

print('Expression Prediction:',objects[ind])
```

```
/usr/local/lib/python3.7/dist-packages/keras_preprocess
  warnings.warn('grayscale is deprecated. Please use '
```


```
Expression Prediction: happy
```

구현된 기능 소개

● goormide 에서 실행한 application.py

application.py

```python
from flask import Flask, request, jsonify

import numpy as np
import pandas as pd

import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Dense, Activation, Dropout, Flatten
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense , Activation , Dropout ,Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from keras.metrics import categorical_accuracy
from keras.models import model_from_json
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.optimizers import *
from tensorflow.keras.layers import BatchNormalization
import keras.backend.tensorflow_backend as K

application = Flask(__name__)

@application.route("/")
def hello():
    return "Hello goorm!"

```

구현된 기능 소개

● 챗봇 & 노래 추천

사용자의 발화
인식 및 응답

데이터 분석
서버와 챗봇
연결(예정)

감정 분석
결과 전달

노래 추천
<슬픔 / 행복>

구현된 기능 소개

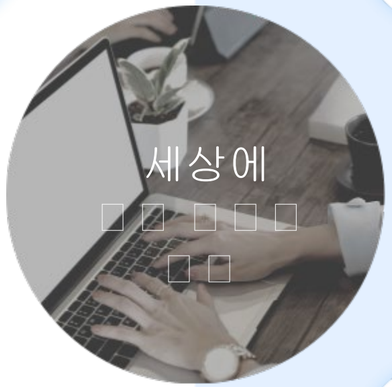
- 최종 작동 시나리오

이용자가 챗봇으로 텍스트
또는 이미지 데이터 전송

01

02

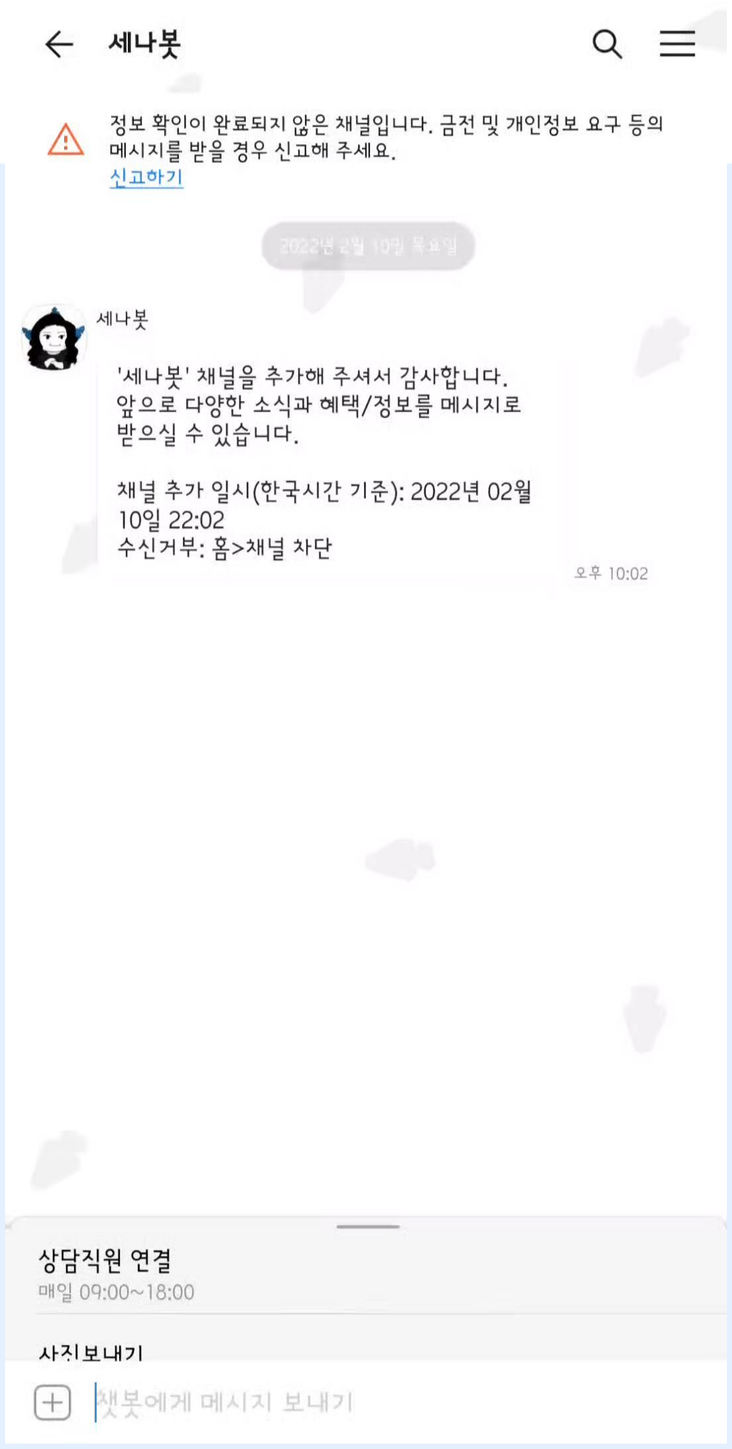
데이터를 스킬서버로 전달하여
텍스트 또는 사진을 분석,
감정 예측



03

예측 결과를
이용자에게 전달

시연 영상


세나봇
정보 확인이 완료되지 않은 채널입니다. 금전 및 개인정보 요구 등의 메시지를 받을 경우 신고해 주세요.
신고하기
2022년 2월 10일 목요일
세나봇
'세나봇' 채널을 추가해 주셔서 감사합니다.
앞으로 다양한 소식과 혜택/정보를 메시지로 받으실 수 있습니다.
채널 추가 일시(한국시간 기준): 2022년 02월 10일 22:02
수신거부: 홈>채널 차단
오후 10:02
상담직원 연결
매일 09:00~18:00
사진보내기
챗봇에게 메시지 보내기

감사합니다