



챗봇 프로젝트 08

머신러닝(딥러닝)

Artificial Intelligence

인공지능

사고나 학습등 인간이 가진
지적 능력을 컴퓨터를 통해
구현하는 기술



Machine Learning

머신러닝

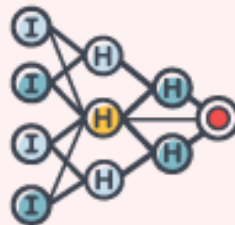
컴퓨터가 스스로 학습하여
인공지능의 성능을
향상 시키는 기술 방법



Deep Learning

딥러닝

인간의 뉴런과 비슷한
인공신경망 방식으로
정보를 처리



들어가기에 앞서

지식이나 실제 개발 경험이 적은 초보자의 입장에서

기본적인 딥러닝 모델과 다양한 개발 기술을 이용해 챗봇의 시스템을

만들어보는 데 목적을 두고 있기 때문에

딥러닝 모델에 대한 자세한 설명은 생략되어 있음

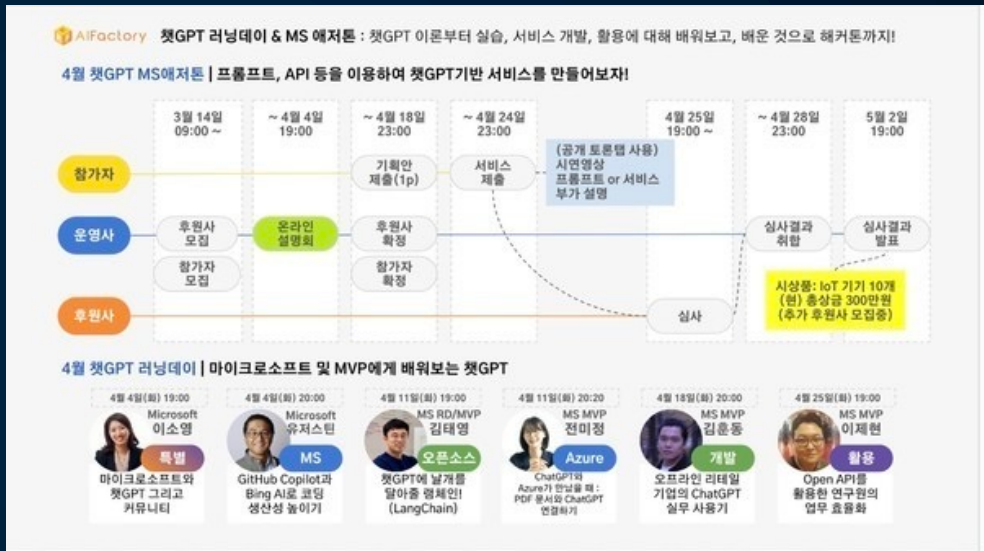
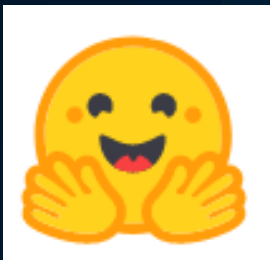
Transformer

2017년 구글에서 발표한 논문을 통해 공개된 딥러닝 모델.
GPT(Generative Pre-trained Transformer)의 기반이 된 모델이기도 하다.

자연어로 프롬프트를 입력하면 답을 내는 지금의 AI는 모두 여기서 파생됨
(ChatGPT, Midjourney, Stable Diffusion, Dall-E..)

기존의 순환신경망(RNN) 방식은 순차적으로 연산을 한다는 단점을
Transformer 모델은 이를 병렬로 처리함으로써
단어간의 연관성 파악, 이해 능력을 획기적으로 높임.

또한 Transformer 모델은 순차적 텍스트, 이미지, 비디오 데이터 등을 사용하는
모든 어플리케이션에 적용될 수 있다.



https://www.youtube.com/watch?v=qEjIMHKmcvA&ab_channel=AINetwork

라이브러리 설치

텐서플로우 2.2 – 딥러닝 모델 실습

```
pip install tensorflow==2.2
```

사이킷런 – 머신러닝 도구 제공

```
pip install sklearn ( 혹은, pip install scikit-learn)
```

Segeval – 모델의 평가를 위한 라이브러리

```
pip install segeval
```

판다스 – 데이터 분석 및 처리

```
pip install pandas xlr
```

Matplotlib – 데이터 시각화 도구 제공

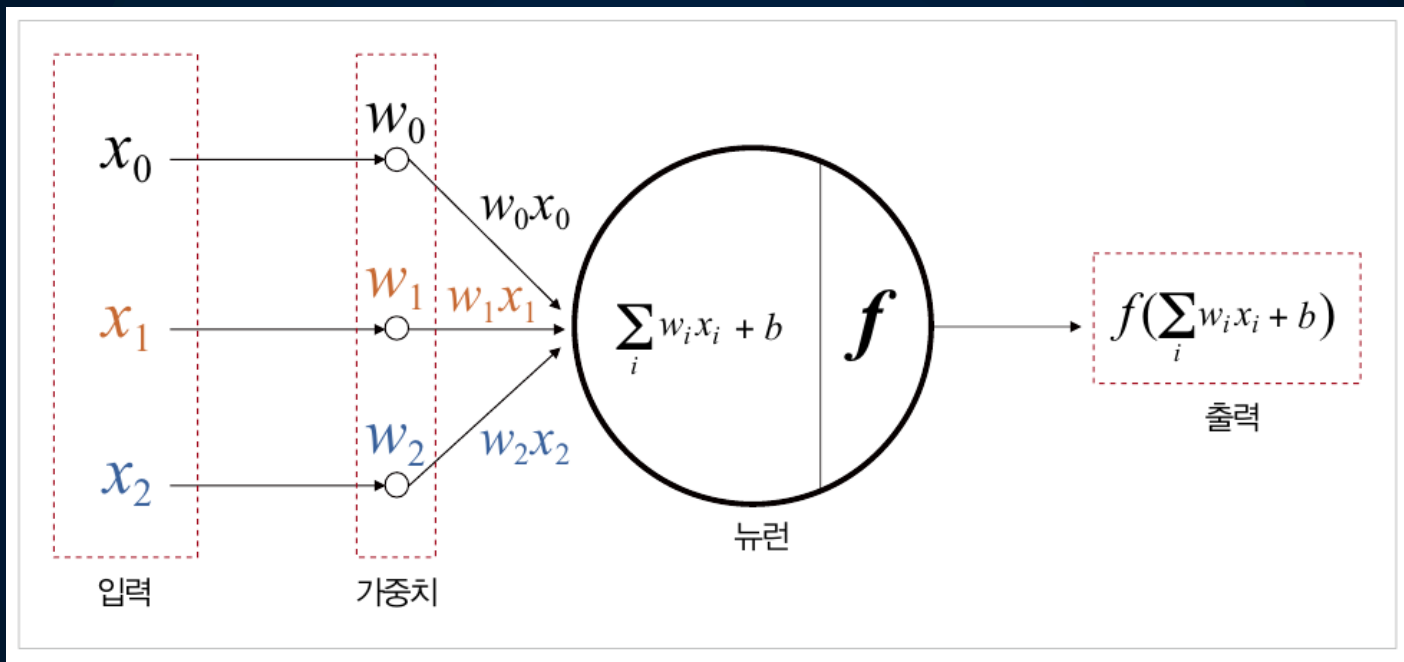
```
pip install matplotlib
```

인공신경망

인공 신경망(Artificial Neural Network)은 두뇌의 신경세포인 뉴런을 수학적으로 모방한 모델
두뇌에 있는 뉴런은 100조 개 이상의 연결체 시냅스로 연결되어 망을 구성하고 있다.

각 뉴런은 다른 뉴런에서 입력 신호를 받아 일정 크기 이상의 신호인지 확인하고,
임계치를 넘어서는 입력이 들어오면 다른 뉴런으로 신호를 보내는 형태로 구성되어 있다.

인공 신경망 역시 입력값이 임계치를 넘어 활성화 되면 다음 뉴런으로 출력값을 내보낸다.

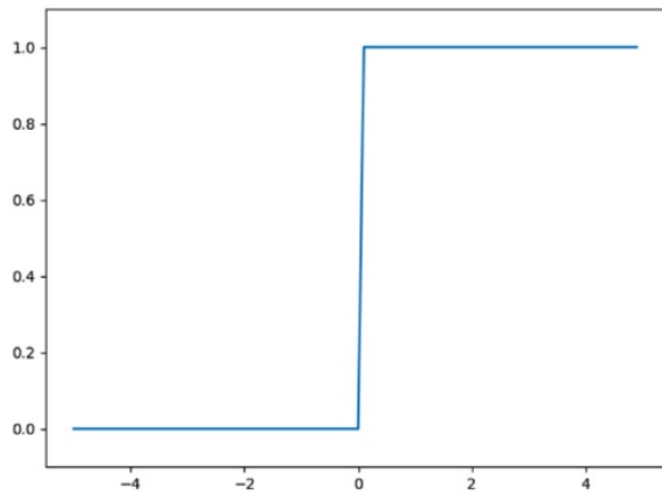


스텝 함수(계단 함수)

스텝 함수는 입력값이 0보다 클 때 1로,
0 이하일 때 0으로 만든다.
(양수일때만 활성화)

때문에 합격/불합격, 참/거짓 같은
이진 분류 문제에 사용된다.

하지만 결과를 너무 극단적으로 나눠서
실제 다양한 환경에 적용하기에는
문제가 있다.



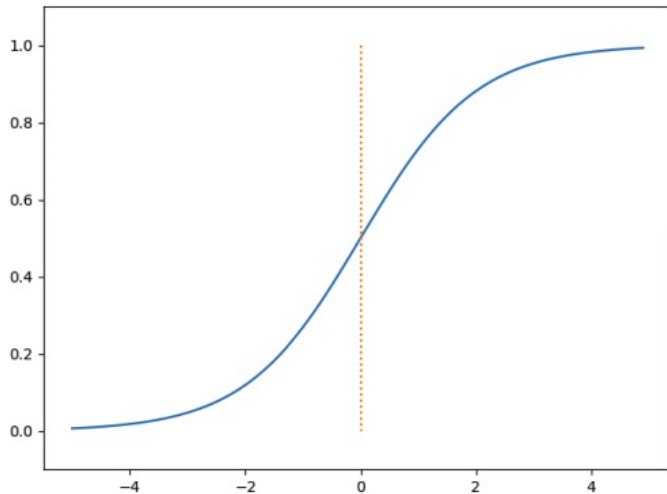
시그모이드 함수

x축은 입력값, y축은 결과값
0에서 1까지의 값을 반환한다.

연속된 데이터이기 때문에 스텝 함수와
다르게 매끄러운 그래프 형태를 보인다.

임계점을 기준으로 0에 가까운 값,
1에 가까운 값으로 이진 분류를 할 수 있다.

아무리 작은 값, 큰 값이 들어와도
결과값의 범위는 0~1의 사이라서
신경망이 깊어질수록 학습의 정확도가
떨어진다는 문제점이 있다

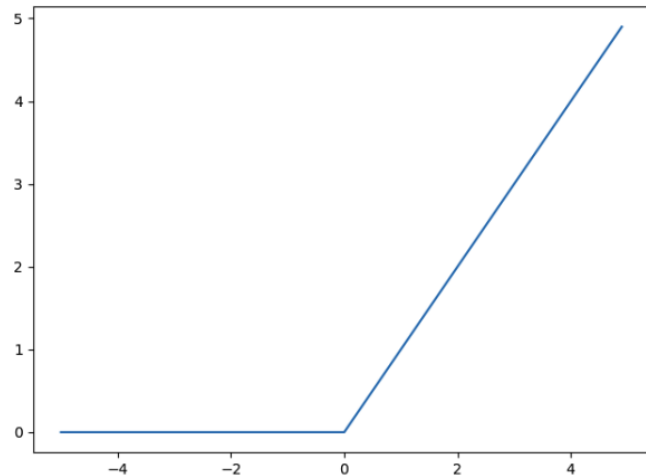


ReLU 함수

입력값이 0 이상인 경우 기울기가 1인 직선,
0보다 작을 땐 결과값이 0이다.

시그모이드 함수에 비해 연산이 적어
학습 속도가 빠르다.

시그모이드 함수에 있었던 문제를
어느정도 개선할 수 있어서 뉴런의
활성화 함수로 많이 사용된다.



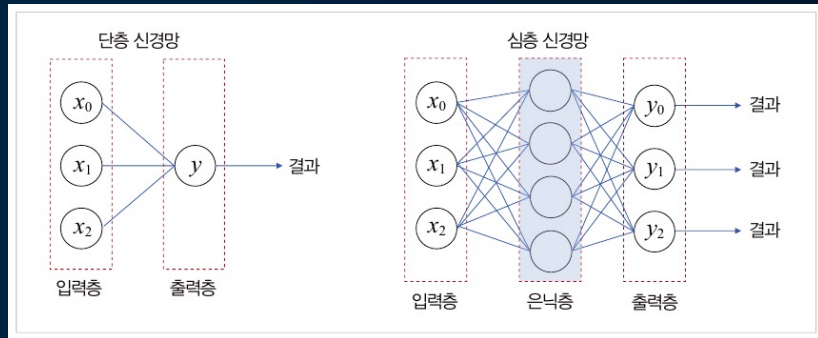
단층 신경망, 심층 신경망

실제로 신경망 모델을 적용할 때는 보통 1개의 뉴런만 사용하지 않음.

문제가 복잡할수록 뉴런 수가 늘어나고 신경망 계층도 깊어지게 된다.

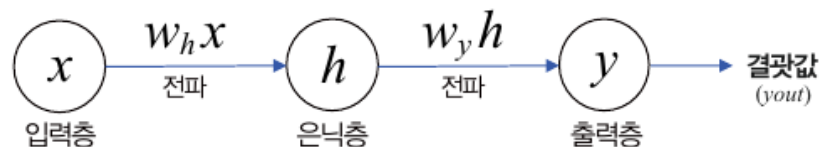
입력층과 출력층 외에 1개 이상의 은닉층을 가지고 있는 신경망 구조를 심층 신경망 (Deep Neural Network, DNN) 이라 한다.

신경망 계층이 깊게 구성되어 각각의 뉴런을 학습 시킨다 하여 딥러닝 모델이라 부른다.

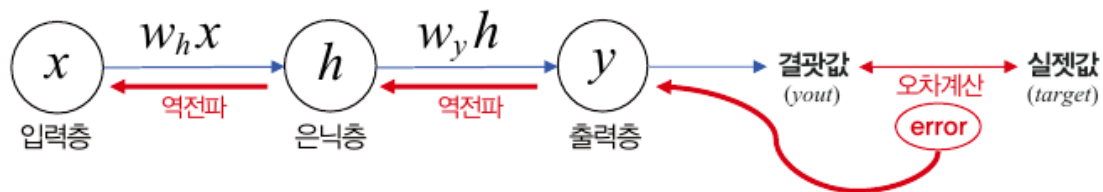


순전파와 역전파

순전파



역전파



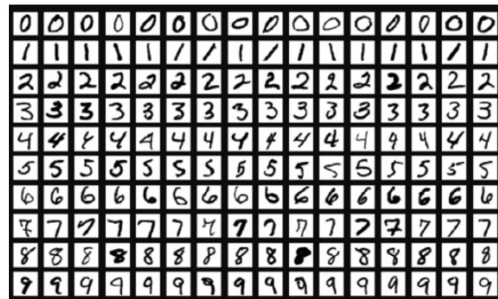
딥러닝 분류 모델 생성

텐서플로 2.1에 포함된 케라스 API를 이용하여 간단한 딥러닝 모델을 만들기

딥러닝 학습에서 유명한 예제인 MNIST 분류 예제

MNIST는 사람이 손글씨로 쓴 0~9까지의 숫자를 이미지화한 데이터셋.

MNIST의 숫자 이미지는 28x28 픽셀 크기의 흑백이미지고, 총 6만개의 학습이미지와 1만개의 테스트 이미지를 포함하고 있다.



〈MNIST 데이터셋 샘플〉

mnist.py

필요한 모듈 임포트

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
```

MNIST 데이터셋 가져오기

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0 # 데이터 정규화
```

tf.data를 사용하여 데이터셋을 섞고 배치 만들기

```
ds = tf.data.Dataset.from_tensor_slices((x_train,
y_train)).shuffle(10000)
train_size = int(len(x_train) * 0.7) # 학습셋: 검증셋 = 7:3
train_ds = ds.take(train_size).batch(20)
val_ds = ds.skip(train_size).batch(20)
```

MNIST 분류 모델 구성

```
model = Sequential()  
model.add(Flatten(input_shape=(28, 28)))  
model.add(Dense(20, activation='relu'))  
model.add(Dense(20, activation='relu'))  
model.add(Dense(10, activation='softmax'))
```

모델 생성

```
model.compile(loss='sparse_categorical_crossentropy',  
              optimizer='sgd', metrics=['accuracy'])  
# model.compile(loss='categorical_crossentropy', optimizer='sgd',  
metrics=['accuracy'])
```

모델 학습

```
hist = model.fit(train_ds, validation_data=val_ds, epochs=10)
```

모델 평가

```
print('모델 평가')  
model.evaluate(x_test, y_test)
```


모델 정보 출력

```
model.summary()
```

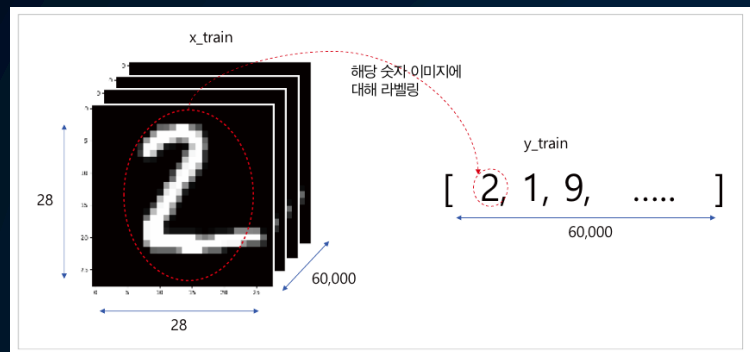
모델 저장

```
model.save('mnist_model.h5')
```

학습 결과 그래프 그리기

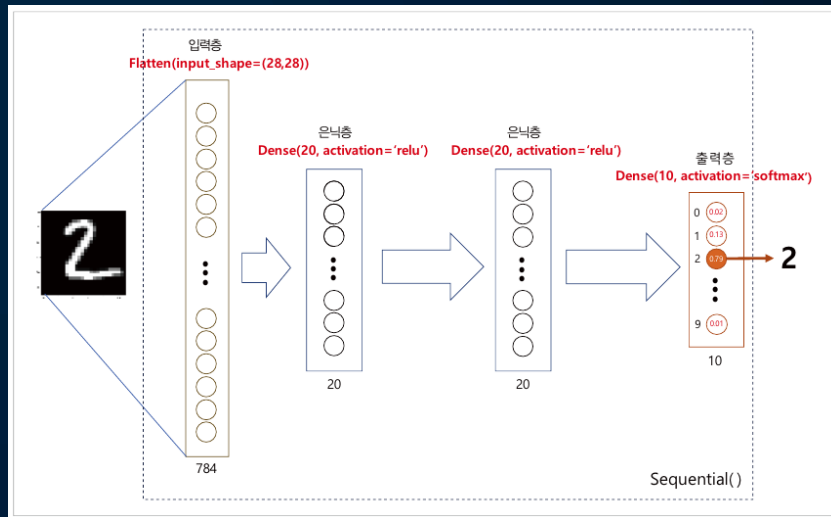
```
fig, loss_ax = plt.subplots()
acc_ax = loss_ax.twinx()
loss_ax.plot(hist.history['loss'], 'y', label='train loss')
loss_ax.plot(hist.history['val_loss'], 'r', label='val loss')
acc_ax.plot(hist.history['accuracy'], 'b', label='train acc')
acc_ax.plot(hist.history['val_accuracy'], 'g', label='val acc')
loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuracy')
loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')
plt.show()
```

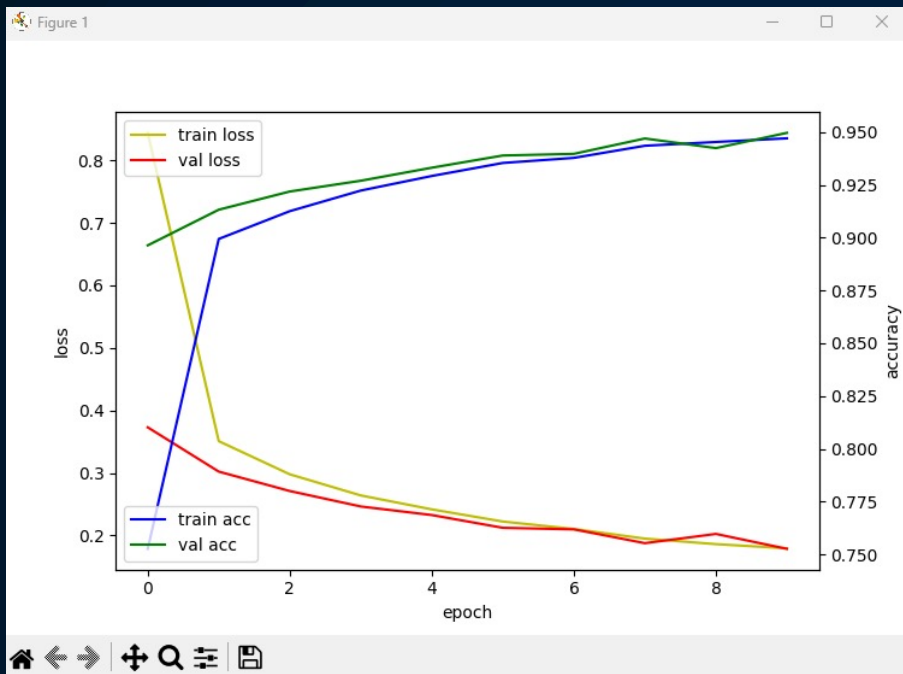
MNIST 데이터셋의 배열 구조. 숫자 이미지와 해당 숫자에 대한 라벨링



MNIST 예제의 신경망 모델
1개의 입력층과 출력층, 2개의 은닉층을 가진다.

출력층은 0~9 숫자에 대한 10개의 뉴런
가중치 계산을 통해 나온 결과값이 가장 높은
뉴런의 값을 최종적으로 출력한다.





matplotlib.pyplot으로 그린 그래프

x축(epoch) : 학습 횟수

y축(loss, accuracy) : 손실값, 정확도

학습 횟수가 증가할수록 학습과
검증 데이터의 정확도가 올라가고
손실값은 감소하는 모습

실행 시 그래프 출력, mnist_model.h5 파일 생성

오류 발생 시, 라이브러리 버전 차이로 인한 오류일 확률 높음.

```
pip install protobuf==3.20.*
```

```
pip install numpy==1.23.4
```

mnist-test.py

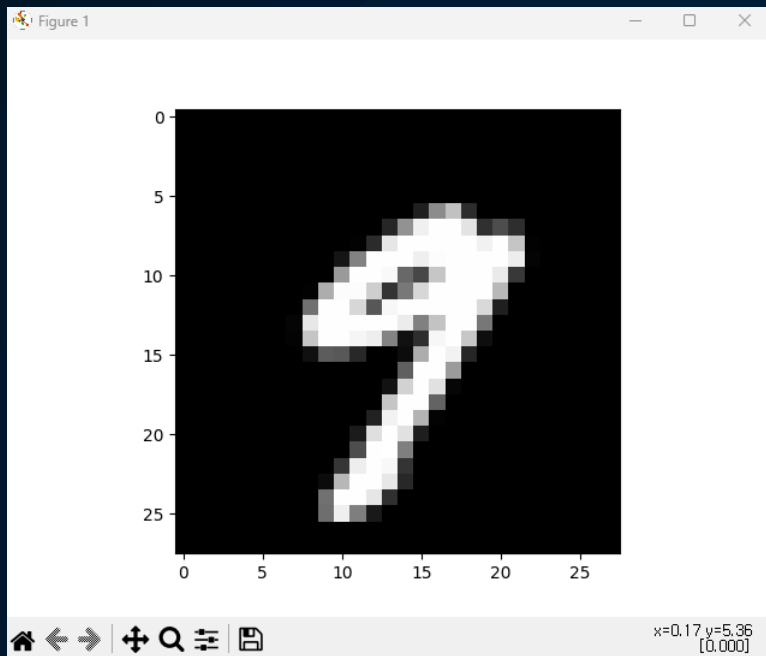
```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt

# MNIST 데이터셋 가져오기
_, (x_test, y_test) = mnist.load_data()
x_test = x_test / 255.0 # 데이터 정규화

# 모델 불러오기
model = load_model('mnist_model.h5')
model.summary()
model.evaluate(x_test, y_test, verbose=2)

# 테스트셋에서 20번째 이미지 출력
plt.imshow(x_test[20], cmap="gray")
plt.show()

# 테스트셋의 20번째 이미지 클래스 분류
picks = [20]
predict = model.predict_classes(x_test[picks])
print("손글씨 이미지 예측값 : ", predict)
```



```
=====
flatten (Flatten)          (None, 784)          0
-----
dense (Dense)              (None, 20)          15700
-----
dense_1 (Dense)            (None, 20)          420
-----
dense_2 (Dense)            (None, 10)          210
=====

Total params: 16,330
Trainable params: 16,330
Non-trainable params: 0

-----
313/313 - 0s - loss: 0.1921 - accuracy: 0.9433
WARNING:tensorflow:From mnist-test.py:20: Sequential.predict_classes (from
e removed after 2021-01-01.
Instructions for updating:
Please use instead: * `np.argmax(model.predict(x), axis=-1)`, if your mo
st-layer activation). * `(model.predict(x) > 0.5).astype("int32")`, if y
ast-layer activation).
손글씨 이미지 예측값 : [9]
```

테스트셋의 20번째 데이터 이미지를 출력,
앞에서 학습된 딥러닝 모델로 판별 결과 손글씨 이미지의 예측값 [9]를 출력.

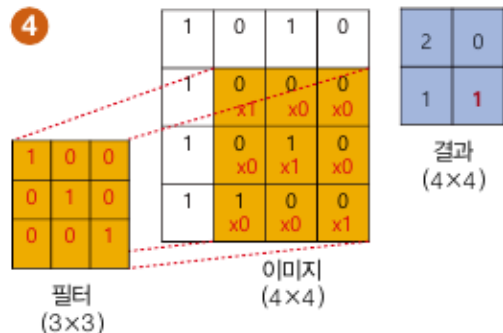
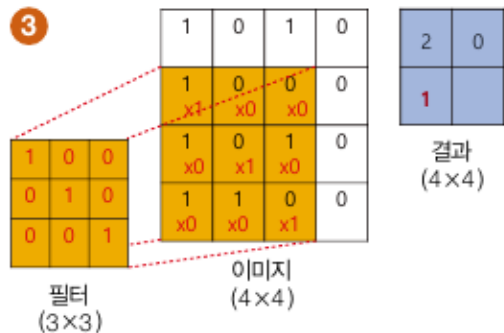
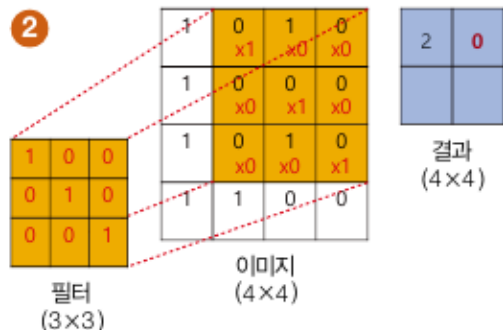
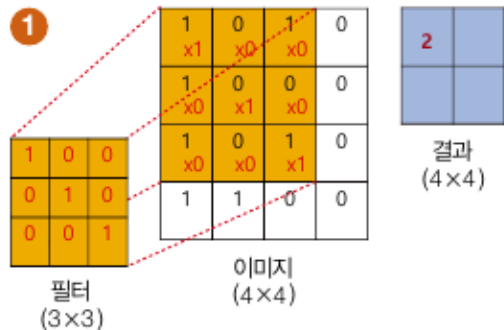
`x_test[20], picks = [20]` 숫자를 바꿔서 실행할 시, 다른 이미지 테스트 가능.

문장 분류를 위한 CNN 모델

챗봇 엔진에 문장 의도 분류를 위해 사용하는 CNN(Convolutional Neural Network) 모델

CNN은 합성곱 신경망으로 불리며, 이미지 관련 분야에서 대표적으로 사용되는 모델이다.

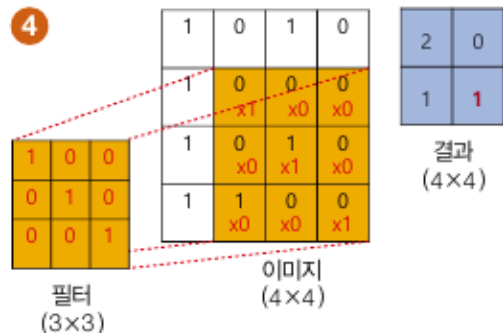
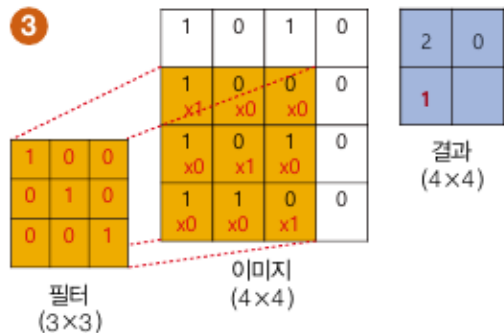
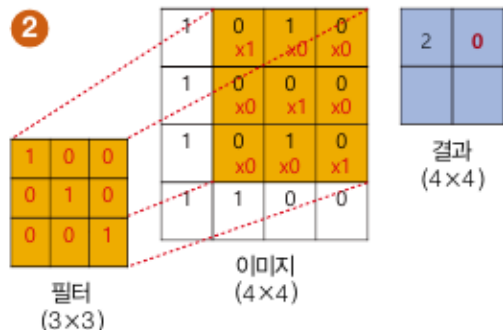
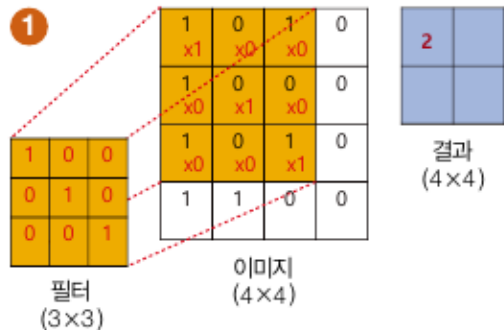
최근에는 자율주행 자동차 및 얼굴인식 등 이미지를 판별하는 분야에서 많이 사용하고 있다.



4x4 크기의 이미지 데이터 행렬에
3x3 크기의 필터를 적용한
합성곱 연산 과정 예시

필터 위치를 이동 시키며
이미지값의 항목별로 각각
곱한 뒤 전부 더해주면 합성곱 연산 완료

나머지 칸도 동일한 방법으로
더 이상 이동을 할 수 없을 때까지
반복.



4x4 크기의 이미지 데이터 행렬에
3x3 크기의 필터를 적용한
합성곱 연산 과정 예시

필터 위치를 이동 시키며
이미지값의 항목별로 각각
곱한 뒤 전부 더해주면 합성곱 연산 완료

나머지 칸도 동일한 방법으로
더 이상 이동을 할 수 없을 때까지
반복.

챗봇 문답 데이터 감정 분류 모델 구현

CNN은 주로 이미지 분류에 사용되지만, 경우에 따라 자연어 분류에서도 좋은 성능을 보인다.

데이터셋

https://github.com/songys/Chatbot_data

ChatbotData.csv 다운로드

Q : 질문

A : 답변

L : 감정

Label	의미
0	일상다반사
1	이별(부정)
2	사랑(긍정)

```
Q,A,label
12시 땀!,하루가 또 가네요.,0
1지망 학교 떨어졌어,위로해 드립니다.,0
3박4일 놀러가고 싶다,여행은 언제나 좋죠.,0
3박4일 정도 놀러가고 싶다,여행은 언제나 좋죠.,0
PPL 심하네,눈살이 찌푸려지죠.,0
SD카드 망가졌어,다시 새로 사는 게 마음 편해요.,0
SD카드 안돼,다시 새로 사는 게 마음 편해요.,0
SNS 맞팔 왜 안하지ㅠㅠ,잘 모르고 있을 수도 있어요.,0
SNS 시간낭비인 거 아는데 매일 하는 중,시간을 정하고 해보세요.,0
SNS 시간낭비인데 자주 보게됨,시간을 정하고 해보세요.,0
SNS보면 나만 빠고 다 행복해보여,자랑하는 자리니까요.,0
가끔 궁금해,그 사람도 그럴 거예요.,0
가끔 뭐하는지 궁금해,그 사람도 그럴 거예요.,0
가끔은 혼자인게 좋다,혼자를 즐기세요.,0
가난한 자의 설움,돈은 다시 들어올 거예요.,0
가만 있어도 땀난다,땀을 식혀주세요.,0
가상화폐 쫄쫄 망함,어서 잊고 새출발 하세요.,0
가스를 켜고 나갔어,빨리 집에 돌아가서 끄고 나오세요.,0
가스를 켜놓고 나온거 같아,빨리 집에 돌아가서 끄고 나오세요.,0
```

cnn-classfication.py

필요한 모듈 임포트

```
import pandas as pd
import tensorflow as tf
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dense,
Dropout, Conv1D, GlobalMaxPool1D, concatenate
```

데이터 읽어오기

```
train_file = "./chatbot_data.csv"
data = pd.read_csv(train_file, delimiter=',')
features = data['Q'].tolist()
labels = data['label'].tolist()
```

단어 인덱스 시퀀스 벡터

```
corpus = [preprocessing.text.text_to_word_sequence(text) for text
in features]
```

```
tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(corpus)
sequences = tokenizer.texts_to_sequences(corpus)
word_index = tokenizer.word_index
MAX_SEQ_LEN = 15 # 단어 시퀀스 벡터 크기
padded_seqs = preprocessing.sequence.pad_sequences(sequences,
maxlen=MAX_SEQ_LEN, padding='post')

# 학습용, 검증용, 테스트용 데이터셋 생성 ③
# 학습셋: 검증셋: 테스트셋 = 7:2:1
ds = tf.data.Dataset.from_tensor_slices((padded_seqs, labels))
ds = ds.shuffle(len(features))
train_size = int(len(padded_seqs) * 0.7)
val_size = int(len(padded_seqs) * 0.2)
test_size = int(len(padded_seqs) * 0.1)
train_ds = ds.take(train_size).batch(20)
val_ds = ds.skip(train_size).take(val_size).batch(20)
test_ds = ds.skip(train_size + val_size).take(test_size).batch(20)
```

하이퍼파라미터 설정

dropout_prob = 0.5

EMB_SIZE = 128

EPOCH = 5

VOCAB_SIZE = len(word_index) + 1 # 전체 단어 수

CNN 모델 정의

input_layer = Input(shape=(MAX_SEQ_LEN,))

embedding_layer = Embedding(VOCAB_SIZE, EMB_SIZE,

input_length=MAX_SEQ_LEN)(input_layer)

dropout_emb = Dropout(rate=dropout_prob)(embedding_layer)

conv1 = Conv1D(filters=128, kernel_size=3, padding='valid',
activation=tf.nn.relu)(dropout_emb)

pool1 = GlobalMaxPool1D()(conv1)

conv2 = Conv1D(filters=128, kernel_size=4, padding='valid',
activation=tf.nn.relu)(dropout_emb)

pool2 = GlobalMaxPool1D()(conv2)

conv3 = Conv1D(filters=128, kernel_size=5, padding='valid',
activation=tf.nn.relu)(dropout_emb)

pool3 = GlobalMaxPool1D()(conv3)

3, 4, 5- gram 이후 합치기

```
concat = concatenate([pool1, pool2, pool3])
hidden = Dense(128, activation=tf.nn.relu)(concat)
dropout_hidden = Dropout(rate=dropout_prob)(hidden)
logits = Dense(3, name='logits')(dropout_hidden)
predictions = Dense(3, activation=tf.nn.softmax)(logits)
```

모델 생성

```
model = Model(inputs=input_layer, outputs=predictions)
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

모델 학습

```
model.fit(train_ds, validation_data=val_ds, epochs=EPOCH, verbose=1)
```

모델 평가(테스트 데이터셋 이용)

```
loss, accuracy = model.evaluate(test_ds, verbose=1)
print('Accuracy: %f' % (accuracy * 100))
print('loss: %f' % (loss))
```

모델 저장

```
model.save('cnn_model.h5')
```

```
atrix:
2023-05-21 04:09:11.223214: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]
Epoch 1/5
414/414 [=====] - 4s 10ms/step - loss: 0.9080 - accuracy: 0.5494 - val_loss: 0.7332 - val_accuracy: 0.6299
Epoch 2/5
414/414 [=====] - 4s 10ms/step - loss: 0.6878 - accuracy: 0.6589 - val_loss: 0.3981 - val_accuracy: 0.8651
Epoch 3/5
414/414 [=====] - 4s 10ms/step - loss: 0.4130 - accuracy: 0.8470 - val_loss: 0.2252 - val_accuracy: 0.9272
Epoch 4/5
414/414 [=====] - 4s 10ms/step - loss: 0.2434 - accuracy: 0.9193 - val_loss: 0.1077 - val_accuracy: 0.9662
Epoch 5/5
414/414 [=====] - 4s 10ms/step - loss: 0.1605 - accuracy: 0.9512 - val_loss: 0.0741 - val_accuracy: 0.9776
60/60 [=====] - 0s 764us/step - loss: 0.0745 - accuracy: 0.9763
Accuracy: 97.631133
loss: 0.074473
```

학습 횟수(Epoch)가 늘어날 수록 loss는 줄어듦, accuracy는 증가하는 모습.
실행 성공 시, cnn_model.h5 모델 파일 생성.

MySQL

MySQL은 가장 많이 사용되는 오픈소스 관계형 데이터베이스 관리 시스템(RDBMS)이다.

챗봇 시스템의 학습 데이터 관리를 위해 MySQL을 사용.

파이썬에서 MySQL을 어떻게 연동하고 사용하는지 알아보고,

이를 활용해 학습툴을 만들어본다.

(chatbot2)

`pip install PyMySQL`

`pip install openpyxl`

명령어	설명
select	데이터 테이블에서 데이터를 조회합니다.
insert	데이터 테이블에 데이터를 삽입합니다.
update	데이터 테이블의 데이터를 변경합니다.
delete	데이터 테이블의 데이터를 삭제합니다.

MySQL 설치

<https://dev.mysql.com/downloads/windows/installer/8.0.html>

MySQL 다운로드 후 실행


Developer Default 선택 후, 설치 진행

MySQL Installer 8.0.33

Select Operating System:
Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer	8.0.33	2.4M	Download
(mysql-installer-web-community-8.0.33.0.msi) MD5: 2a330cf24915964cca87e04dbb34e5d3 Signature			
Windows (x86, 32-bit), MSI Installer	8.0.33	428.3M	Download
(mysql-installer-community-8.0.33.0.msi) MD5: 3d4c833d05a6e0a330b0c411d4d1cc Signature			

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

설치 과정 중, 비밀번호 설정 '1234'

설치 완료 후, MySQL 8.0 Command line Client 실행

`show databases;` 입력 시 현재 데이터베이스 목록 출력

`create database homestead;` homestead라는 이름의 데이터베이스 생성

```
mysql> create schema homestead
-> ;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| homestead |
| information_schema |
| mysql |
| performance_schema |
| sakila |
| sys |
| world |
+-----+
```

db-conn.py

```
import pymysql
db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    print("DB 연결 성공")

except Exception as e:
    print(e)
finally:
    if db is not None:
        db.close()
        print("DB 연결 닫기 성공")
```

```
(chatbot2) C:\Users\yubeen\python-chatbot\DB>python db_conn.py  
DB 연결 성공  
DB 연결 닫기 성공
```

DB 정상 연결 시,

DB 연결 성공

DB 연결 닫기 성공

메시지 출력

make-table.py

```
import pymysql

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    sql = '''
    CREATE TABLE tb_student (
        id int primary key auto_increment not null,
        name varchar(32),
        age int,
        address varchar(32)
    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
    '''
```

```
with db.cursor() as cursor:
    cursor.execute(sql)

except Exception as e:
    print(e)

finally:
    if db is not None:
        db.close()
```

실행 후, mysql cli client에서

`use homestead;` 만든 DB 선택

`show tables;` 선택한 DB 안에 있는 테이블 조회

tb_student 테이블이 생성되어있으면 성공.

```
mysql> show tables;
+-----+
| Tables_in_homestead |
+-----+
| tb_student           |
+-----+
1 row in set (0.00 sec)
```

insert-data.py

```
import pymysql

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    sql = '''
    INSERT tb_student(name, age, address) values('Kei', 35,
    'Korea')
    '''
    with db.cursor() as cursor:
        cursor.execute(sql)
    db.commit()
```

```
except Exception as e:  
    print(e)  
  
finally:  
    if db is not None:  
        db.close()
```

실행 후, mysql cli client에서

```
select * from tb_student;
```

선택한 테이블에서 데이터 전체를 조회

코드 내 sql 문에 입력한 데이터가 저장됐으면 성공

```
mysql> select * from tb_student  
-> ;  
+----+-----+-----+-----+  
| id | name | age  | address |  
+----+-----+-----+-----+  
|  1 | Kei  |   35 | Korea   |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

update-data.py

```
import pymysql

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    id = 1  # E//O/E/ id (PK)
    sql = '''
        UPDATE tb_student set name="케이", age=36 where id=%d
        ''' % id
    with db.cursor() as cursor:
        cursor.execute(sql)
    db.commit()
```



```
except Exception as e:  
    print(e)  
  
finally:  
    if db is not None:  
        db.close()
```

실행 후, mysql cli client에서

`select * from tb_student;`

선택한 테이블에서 데이터 전체를 조회

기존 데이터에서 이름 나이의 데이터가
수정 되었으면 성공.

```
mysql> select * from tb_student;  
+----+-----+-----+-----+  
| id | name | age  | address |  
+----+-----+-----+-----+  
|  1 | 케이 |   36 | Korea   |  
+----+-----+-----+-----+  
1 row in set (0.00 sec)
```

delete-data.py

```
import pymysql

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    id = 1  # E//O/E/ id (PK)
    sql = '''
        DELETE from tb_student where id=%d
        ''' % id
    with db.cursor() as cursor:
        cursor.execute(sql)
    db.commit()
```

```
except Exception as e:  
    print(e)  
  
finally:  
    if db is not None:  
        db.close()
```

실행 후, mysql cli client에서

```
select * from tb_student;
```

선택한 테이블에서 데이터 전체를 조회

테이블 데이터가 삭제 되었으면 성공.

```
mysql> select * from tb_student;  
Empty set (0.00 sec)
```

select-data.py

```
import pymysql
import pandas as pd

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
    students = [
        {'name': 'Kei', 'age': 36, 'address': 'PUSAN'},
        {'name': 'Tony', 'age': 34, 'address': 'PUSAN'},
        {'name': 'Jaeyoo', 'age': 39, 'address': 'GWANGJU'},
        {'name': 'Grace', 'age': 28, 'address': 'SEOUL'},
        {'name': 'Jenny', 'age': 27, 'address': 'SEOUL'},
    ]
```

```
for s in students:
    with db.cursor() as cursor:
        sql = '''
            insert tb_student(name, age, address)
values("%s",%d,"%s")
            ''' % (s['name'], s['age'], s['address'])
        cursor.execute(sql)
    db.commit()

cond_age = 30
with db.cursor(pymysql.cursors.DictCursor) as cursor:
    sql = '''
    select * from tb_student where age > %d
    ''' % cond_age
    cursor.execute(sql)
    results = cursor.fetchall()
print(results)
```

```
cond_name = 'Grace'
with db.cursor(pymysql.cursors.DictCursor) as cursor:
    sql = '''
        select * from tb_student where name="%s"
        ''' % cond_name
    cursor.execute(sql)
    result = cursor.fetchone()
    print(result['name'], result['age'])

df = pd.DataFrame(results)
print(df)
```

```
except Exception as e:
    print(e)
```

```
finally:
    if db is not None:
        db.close()
```

```
(chatbot2) C:\Users\yubeen\python-chatbot\DB>python select-data.py
[{'id': 2, 'name': 'Kei', 'age': 36, 'address': 'PUSAN'}, {'id': 3, 'name': 'Tony', 'age': 34, 'address': 'PUSAN'}, {'id': 4, 'name': 'Jaeyoo', 'age': 39, 'address': 'GWANGJU'}]
Grace 28
```

	id	name	age	address
0	2	Kei	36	PUSAN
1	3	Tony	34	PUSAN
2	4	Jaeyoo	39	GWANGJU

코드 실행 시, 테이블에 다수의 데이터 추가 후,

조건에 맞는 데이터를 출력

mysql cli client에서

`select * from tb_student;`

선택한 테이블에서 데이터 전체를 조회

추가한 다수의 데이터 컬럼 확인 가능.

```
mysql> select * from tb_student;
+----+-----+-----+-----+
| id | name  | age  | address |
+----+-----+-----+-----+
| 2  | Kei   | 36   | PUSAN   |
| 3  | Tony  | 34   | PUSAN   |
| 4  | Jaeyoo | 39   | GWANGJU |
| 5  | Grace | 28   | SEOUL   |
| 6  | Jenny | 27   | SEOUL   |
+----+-----+-----+-----+
5 rows in set (0.00 sec)
```

실습. User 데이터베이스를 생성, user_info 테이블을 생성하여 아래 표와 같이 데이터를 입력해보자
(데이터베이스, 테이블, 컬럼 등의 이름은 임의대로 지어도 상관없음)

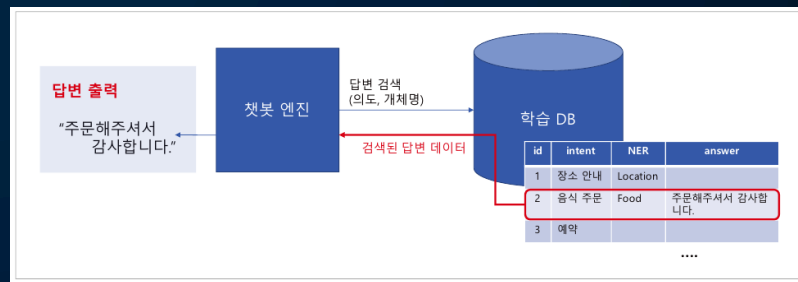
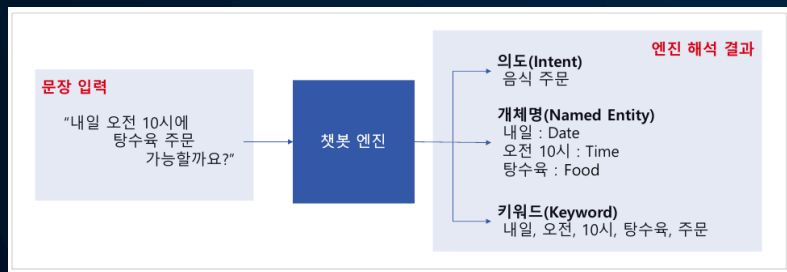
이름	성별	나이	전화번호	국적
김철수	남자	15	010-1234-5678	한국
홍길동	남자	35	010-2222-3333	한국
Jason	남자	21	010-1212-2323	미국
이영희	여자	22	010-9876-1234	한국
Emma	여자	27	010-4567-7890	스페인

챗봇 학습툴 만들기

MySQL로 챗봇의 학습 데이터를 관리하는 툴을 제작

학습데이터를 DB에 저장했을 때 실시간으로 챗봇 시스템에 적용될 수 있도록 구현

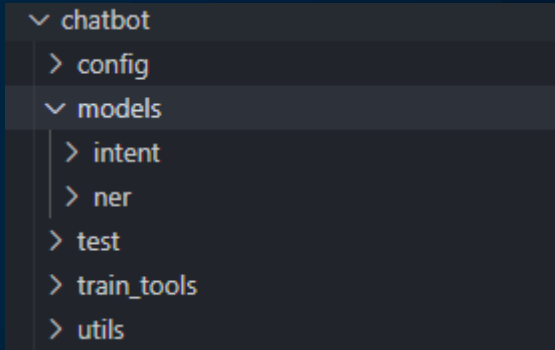
이를 위해 챗봇이 이해할 수 있는 질문 정보와 해당 답변 데이터를 관리하기 위한 툴이 필요.



폴더 구조

챗봇 구현 과정 시 파일 관리를 위해 아래와 같이 폴더를 생성

```
chatbot
├── config
├── models
│   ├── intent
│   └── ner
├── test
├── train_tools
└── utils
```



```
▼ chatbot
  > config
  ▼ models
    > intent
    > ner
  > test
  > train_tools
  > utils
```

A screenshot of a file explorer window showing the directory structure for a chatbot project. The 'chatbot' directory is expanded, revealing subdirectories: 'config', 'models' (which is further expanded to show 'intent' and 'ner'), 'test', 'train_tools', and 'utils'.

DatabaseConfig.py

```
DB_HOST = "127.0.0.1"
DB_USER = "root"
DB_PASSWORD = "1234"
DB_NAME = "homestead"

def DatabaseConfig():
    global DB_HOST, DB_USER, DB_PASSWORD, DB_NAME
```

코드 작성 후 /config 폴더 안에 저장

DB 접속 정보는 본인 환경에 맞춰서 작성

DB 설계, DB 생성

아래 표와 같은 형식의 테이블을 생성

테이블 명 : chatbot_train_data

컬럼	속성	설명
id	int primary key not null	학습 데이터 id
intent	varchar(45)	의도명, 의도가 없는 경우 null
ner	varchar(45)	개체명, 개체명이 없는 경우 null
query	text null	질문 텍스트
answer	text not null	답변 텍스트
answer_image	varchar(2048)	답변에 들어갈 이미지 URL, 이미지 URL을 사용하지 않을 경우 null

create_train_data_table.py

/train_tools/qna 폴더 생성 후, qna 폴더 안에 코드 파일 작성

```
import pymysql
import sys
sys.path.append('../..')
from config.DatabaseConfig import *

db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )
```

```
sql = '''
    CREATE TABLE IF NOT EXISTS `chatbot_train_data` (
        `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
        `intent` VARCHAR(45) NULL,
        `ner` VARCHAR(1024) NULL,
        `query` TEXT NULL,
        `answer` TEXT NOT NULL,
        `answer_image` VARCHAR(2048) NULL,
        PRIMARY KEY(`id`))
ENGINE = InnoDB DEFAULT CHARSET=utf8
'''

with db.cursor() as cursor:
    cursor.execute(sql)
db.commit()

except Exception as e:
    print(e)

finally:
    if db is not None:
        db.close()
```

코드 실행 시, 테이블 구조 생성

`show tables;`

DB안의 테이블 목록 출력

`desc chatbot_train_data;`

chatbot_train_data 테이블의 구조 출력

실행 성공 시,

테이블이 추가됨과 동시에

데이터를 담을 컬럼 구조가 생성된다.

```
mysql> show tables;
+-----+
| Tables_in_homestead |
+-----+
| chatbot_train_data  |
| tb_student          |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> desc chatbot_train_data;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+-----+
| id         | int unsigned  | NO   | PRI | NULL    | auto_increment |
| intent     | varchar(45)   | YES  |     | NULL    |                 |
| ner        | varchar(1024) | YES  |     | NULL    |                 |
| query      | text          | YES  |     | NULL    |                 |
| answer     | text          | NO   |     | NULL    |                 |
| answer_image | varchar(2048) | YES  |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

데이터 엑셀 파일 및 DB연동

	A	B	C	D	E
1	의도(Intent)	개체명(NER)	질문(Query)	답변(Answer)	답변 이미지
2	인사		안녕하세요	네 안녕하세요 :D 반갑습니다. 저는 챗봇입니다.	
3	인사		반가워요	네 안녕하세요 :D 반갑습니다. 저는 챗봇입니다.	
4					
5					
6					

학습툴의 UI를 대신하여 엑셀을 통해 학습 데이터를 추가 삭제

엑셀 파일을 학습툴에 입력해 DB 내용을 업데이트 하는 형태로 구현

create_train_data_table.py

/train_tools/qna 폴더 생성 후, qna 폴더 안에 코드 파일 작성

```
import pymysql
import openpyxl
import sys
sys.path.append('../..')
from config.DatabaseConfig import *

def all_clear_train_data(db):
    sql = '''
        delete from chatbot_train_data
    '''

    with db.cursor() as cursor:
        cursor.execute(sql)

    sql = '''
        ALTER TABLE chatbot_train_data AUTO_INCREMENT=1
    '''

    with db.cursor() as cursor:
        cursor.execute(sql)
```

```
def insert_data(db, xls_row):
    intent, ner, query, answer, answer_img_url = xls_row

    sql = '''
        INSERT chatbot_train_data(intent, ner, query, answer,
        answer_image)
        values(
            '%s', '%s', '%s', '%s', '%s'
        )
    ''' % (intent.value, ner.value, query.value, answer.value,
    answer_ima_url.value)

    sql = sql.replace("'None'", "null")

    with db.cursor() as cursor:
        cursor.execute(sql)
        print('{} 저장'.format(query.value))
        db.commit()
```

```
train_file = './train_data.xlsx'
db = None
try:
    db = pymysql.connect(
        host='127.0.0.1',
        user='root',
        passwd='1234',
        db='homestead',
        charset='utf8'
    )

    all_clear_train_data(db)

    wb = openpyxl.load_workbook(train_file)
    sheet = wb['Sheet1']
    for row in sheet.iter_rows(min_row=2):
        insert_data(db, row)

    wb.close()

except Exception as e:
    print(e)

finally:
    if db is not None:
        db.close()
```

코드 실행 시, 엑셀 파일의 데이터가

MySQL DB에 저장된 것을 확인할 수 있음.