
빅데이터 분석 15주차 실시간 세션

김 종 우



목차

- 감성분석
- 감성분석 실습
- 기말고사 안내

감성분석 목차

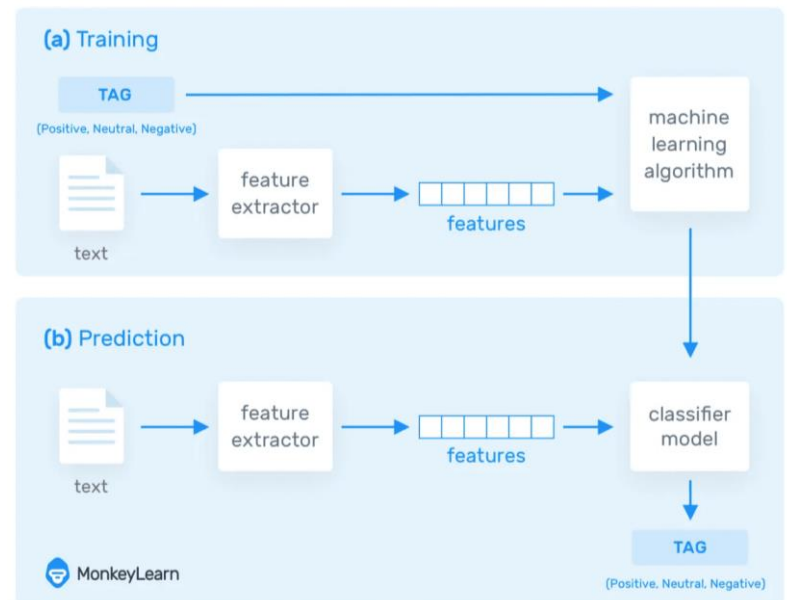
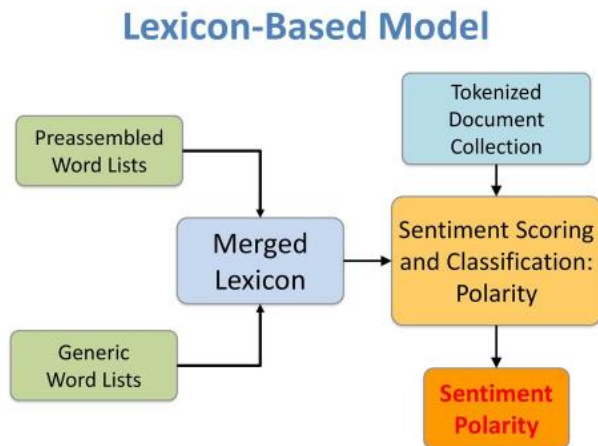
- 사전기반 감성 분석과 실습
 - VADER를 이용한 감성 분석
 - 로지스틱 회귀분석을 이용한 감성 분석
- 기계학습 기반 영어 감성 분석
 - CNN을 활용한 감성 분석
- BERT를 활용한 감성 분석

감성 분석

- 오피니언 마이닝
- 공개된 정보 원천들을 자동으로 읽어서 사람들이 어떤 것(브랜드, 제품, 연애인, 주가 등)에 대하여 어떻게 느끼고 있는지를 판단
- 활용
 - 상품과 콘텐츠 감성 분석
 - 정책 여론 분석
 - 영화 감성 분석을 활용한 흥행 예측
 - 기업 뉴스를 활용한 주가 예측
 - 기업 및 브랜드 감성 분석

감성 분석 방법

- 사전 기반 감성 분석
 - Lexicon-based 접근법
- 기계학습 기반 감성 분석
 - 나이브 베이지안, 로지스틱 회귀분석, SVM, 의사결정나무, Random Forest, CNN, RNN
 - BERT, GPT-2



다범주 감성 분석



POSITIVE



NEUTRAL



NEGATIVE



JOY



SURPRISE



ANGER



DISGUST



FEAR



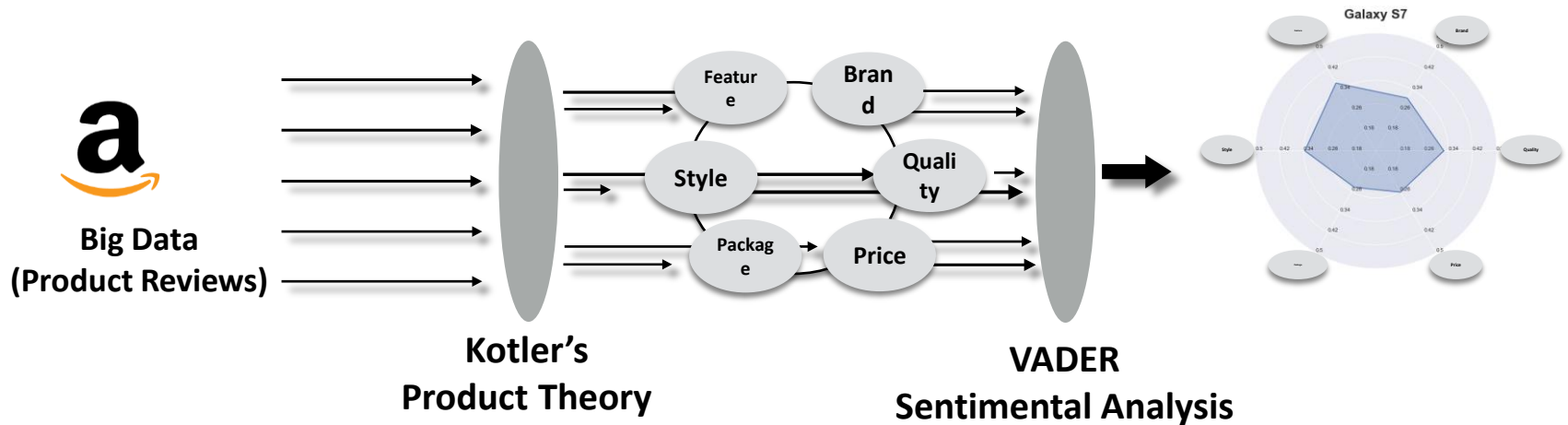
SADNESS

SENTIMENT

EMOTION

다차원 감성 분석

- Multi-Aspect Sentiment Analysis



1. Data Crawling

- Python Beautiful Soup

2. Data Preprocessing

- Word Tokenization
- Sentence Tokenization
- Stemming, Stopwords, etc.

3. Data Classification

- Using WordNet Synsets
- Keep tracking index

4. VADER Sentimental Analysis

- { score | $-1.0 \leq \text{score} \leq 1.0$ }
- To each sentence of each category

VADER와 로지스틱 회귀분석을 이용한 감성 분석

- 데이터 읽기와 전처리

```
import pandas as pd
```

```
review_df = pd.read_csv('labeledTrainData.tsv', header=0, sep="\t", quoting=3)  
review_df.head(3)
```

```
print(review_df['review'][0])
```

```
import re
```

```
# <br> html 태그는 replace 함수로 공백으로 변환
```

```
review_df['review'] = review_df['review'].str.replace('<br />', ' ')
```

```
review_df['review'] = review_df['review'].apply( lambda x : re.sub("[^a-zA-Z]", " ", x) )
```


VADER와 로지스틱 회귀분석을 이용한 감성 분석

- 성능 평가 함수 만들기

```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score
from sklearn.metrics import recall_score, f1_score, roc_auc_score
```

```
def get_clf_eval(y_test=None, pred=None):
    confusion = confusion_matrix( y_test, pred)
    accuracy = accuracy_score(y_test , pred)
    precision = precision_score(y_test , pred)
    recall = recall_score(y_test , pred)
    f1 = f1_score(y_test,pred)
    # ROC-AUC 추가
    roc_auc = roc_auc_score(y_test, pred)
    print('오차 행렬')
    print(confusion)
    # ROC-AUC print 추가
    print('정확도: {0:.4f}, 정밀도: {1:.4f}, 재현율: {2:.4f},\
F1: {3:.4f}, AUC:{4:.4f}'.format(accuracy, precision, recall, f1, roc_auc))
```

VADER와 로지스틱 회귀분석을 이용한 감성 분석

- VADER를 사용한 감성 분석

```
import nltk  
nltk.download('all')
```

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
senti_analyzer = SentimentIntensityAnalyzer()  
senti_scores = senti_analyzer.polarity_scores(review_df['review'][0])  
print(senti_scores)
```

```
def vader_polarity(review, threshold=0.1):  
    analyzer = SentimentIntensityAnalyzer()  
    scores = analyzer.polarity_scores(review)  
  
    agg_score = scores['compound']  
    final_sentiment = 1 if agg_score >= threshold else 0  
    return final_sentiment
```

VADER와 로지스틱 회귀분석을 이용한 감성 분석

- VADER를 사용한 감성 분석

```
review_df['vader_preds'] = review_df['review'].apply( lambda x : vader_polarity(x, 0.1) )  
y_target = review_df['sentiment'].values  
vader_preds = review_df['vader_preds'].values
```

```
print('#### VADER 예측 성능 평가 ####')  
get_clf_eval(y_target, vader_preds)
```

오차 행렬

```
[[ 6729  5771]  
 [ 1858 10642]]
```

정확도: 0.6948, 정밀도: 0.6484, 재현율: 0.8514, F1: 0.7361, AUC:0.6948

VADER와 로지스틱 회귀분석을 이용한 감성 분석

- 훈련 데이터와 테스트 데이터 분리

```
from sklearn.model_selection import train_test_split
```

```
class_df = review_df['sentiment']
```

```
feature_df = review_df.drop(['id','sentiment','vader_preds'], axis=1, inplace=False)
```

```
X_train, X_test, y_train, y_test= train_test_split(feature_df, class_df, test_size=0.3, \
                                                    random_state=156)
```

```
X_train.shape, X_test.shape
```

VADER와 로지스틱 회귀분석을 이용한 감성 분석

- 로지스틱 회귀분석

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
```

```
pipeline = Pipeline([
    ('cnt_vect', CountVectorizer(stop_words='english', ngram_range=(1,2) )),
    ('lr_clf', LogisticRegression(C=10))])
```

```
pipeline.fit(X_train['review'], y_train)
pred = pipeline.predict(X_test['review'])
```

```
print('##### 로지스틱 회귀 감성 분류 예측 성능 평가 #####')
get_clf_eval(y_test, pred)
```

정확도: 0.8860, 정밀도: 0.8876, 재현율: 0.8887, F1: 0.8882, AUC:0.8859

감성 분석 Lab-1

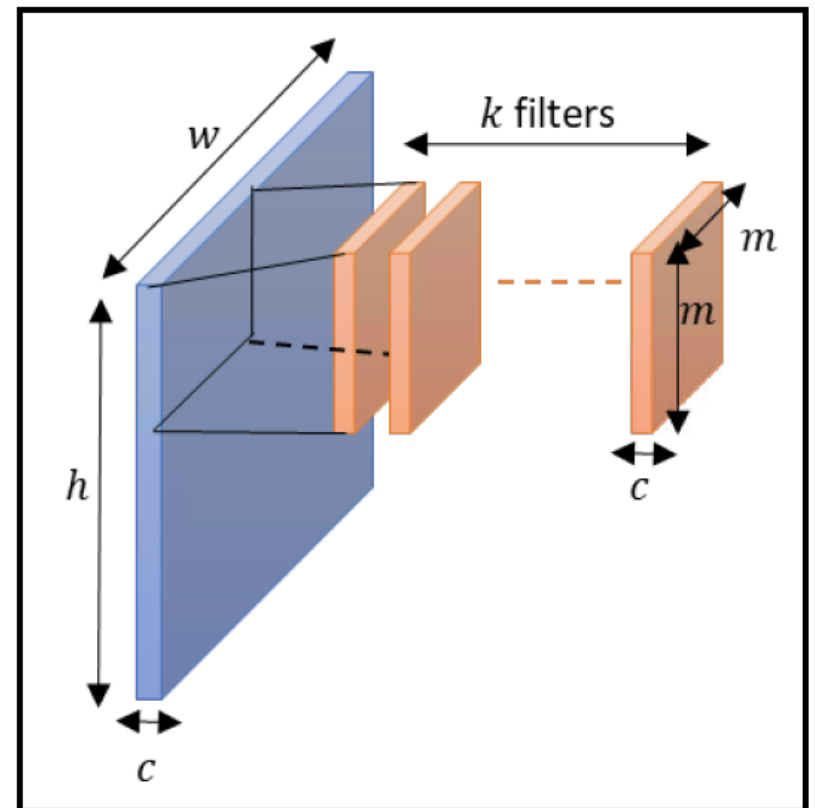
- amazon_mobile_reviews.csv 파일은 모바일 단말기에 대한 아마존 리뷰를 수집한 데이터이다. 이를 이용해서 다음을 수행하시오.
- 데이터파일을 읽어들여서, review_df에 저장하시오.
- review_df 내에 'body' 컬럼이 null인 행들을 삭제하시오.
- 'rating' 컬럼 값이 3인 행들을 삭제하시오.
- 'rating' 값이 4,5인 경우는 1, 1,2인 경우는 0를 가지는 컬럼 'sentiment'를 생성하시오.
- 인덱스를 재설정하시오(Hint. reset_index() 함수 사용).

감성 분석 Lab-1

- 데이터를 전처리하시오.
- 전처리된 데이터를 이용하여 VADER 패키지를 활용하여 감성 분석을 수행하고 성능을 평가하시오.
- 훈련 데이터와 테스트 데이터를 7대 3의 비율로 나누고 로지스틱 회귀분석을 이용하여 감성 분석을 하고 성능을 평가하시오.

CNN

- Two-dimensional (2D) => 이미지
- One-dimensional (1D) => 순차 입력
- Convolution Layer
- Pooling Layer



Convolutional neural networks

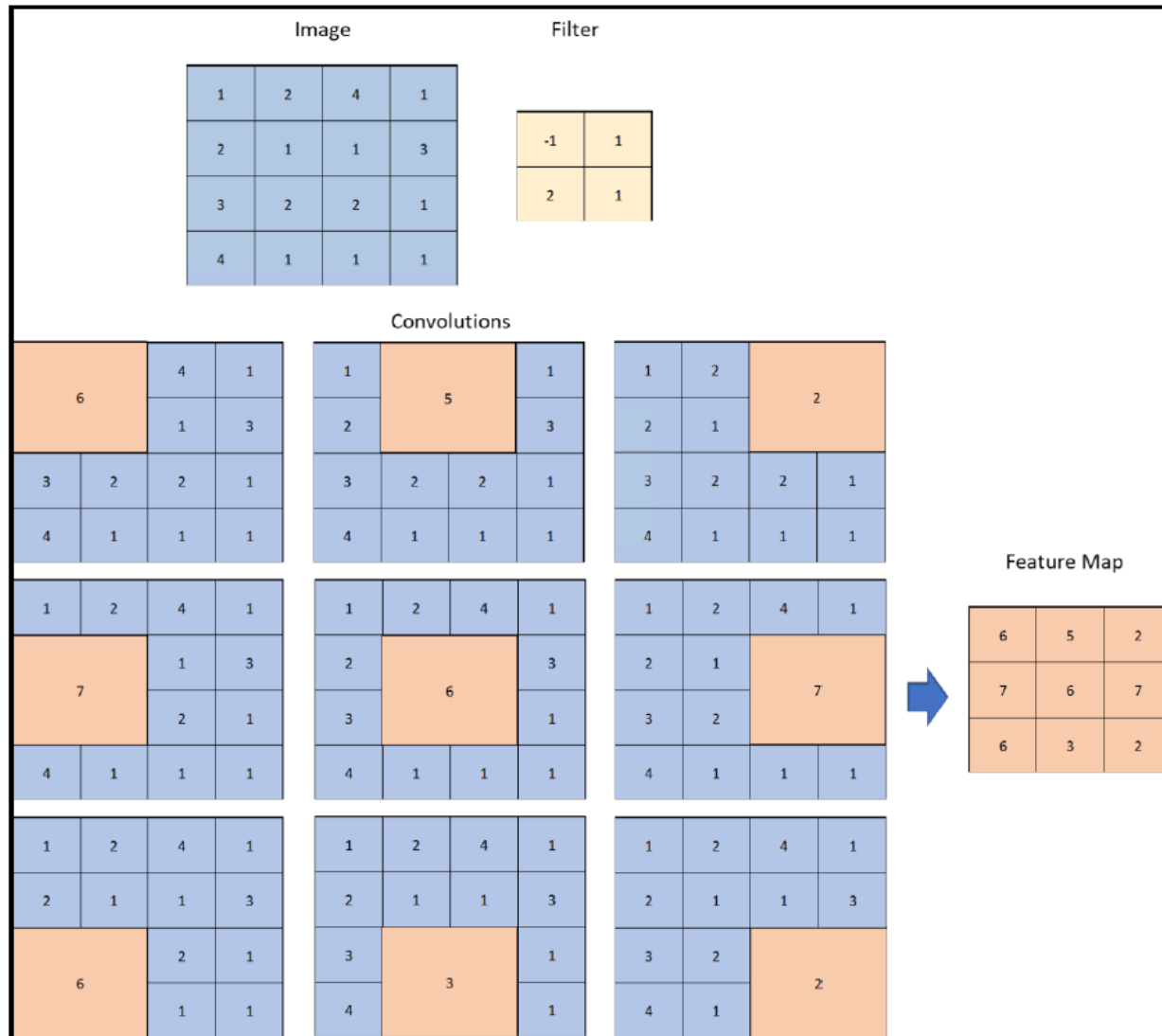
Image

1	2	4	1
2	1	1	3
3	2	2	1
4	1	1	1

Filter

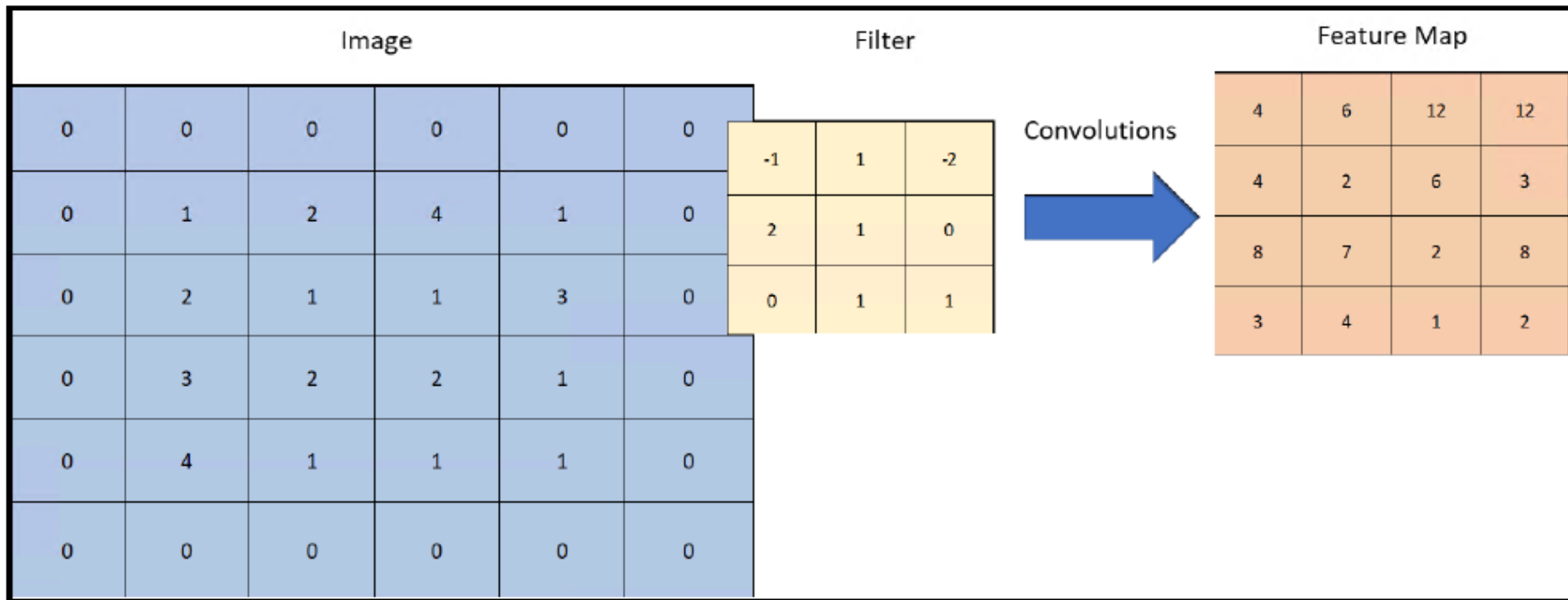
-1	1
2	1

Convolutional neural networks



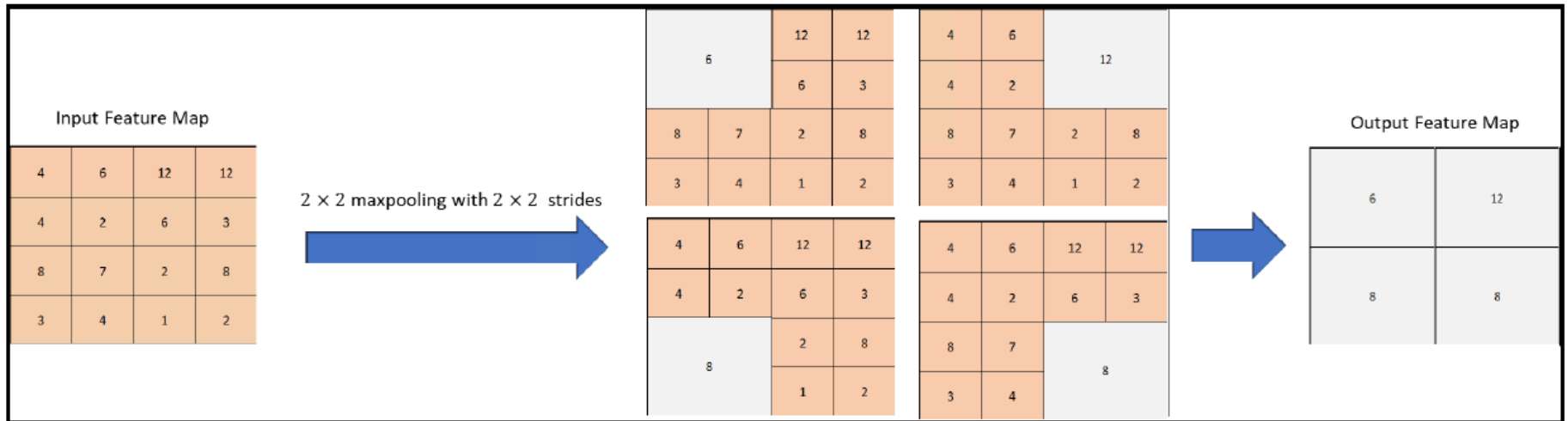
Convolutional neural networks

- Zero-padding



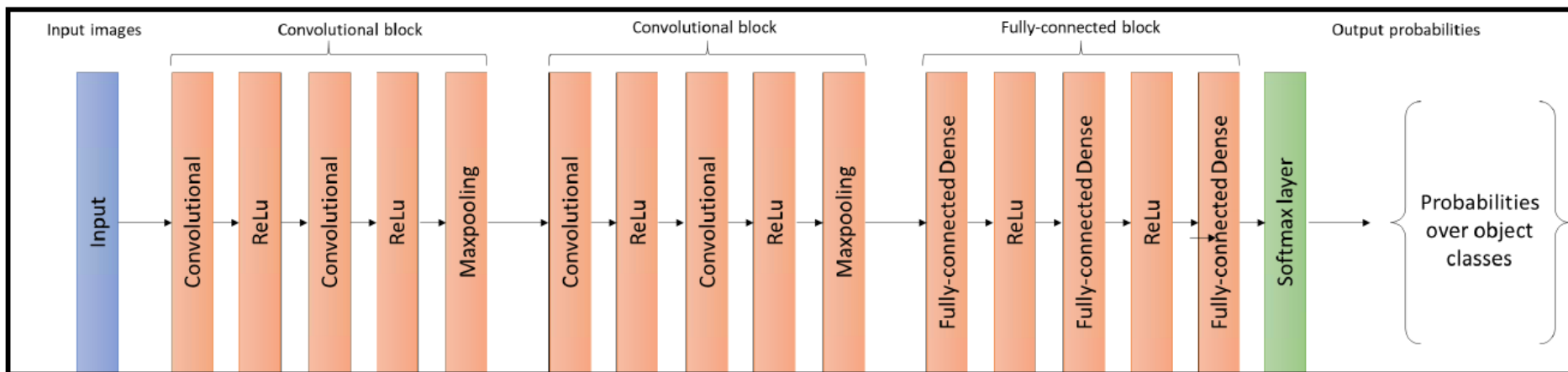
Convolutional neural networks

- Pooling

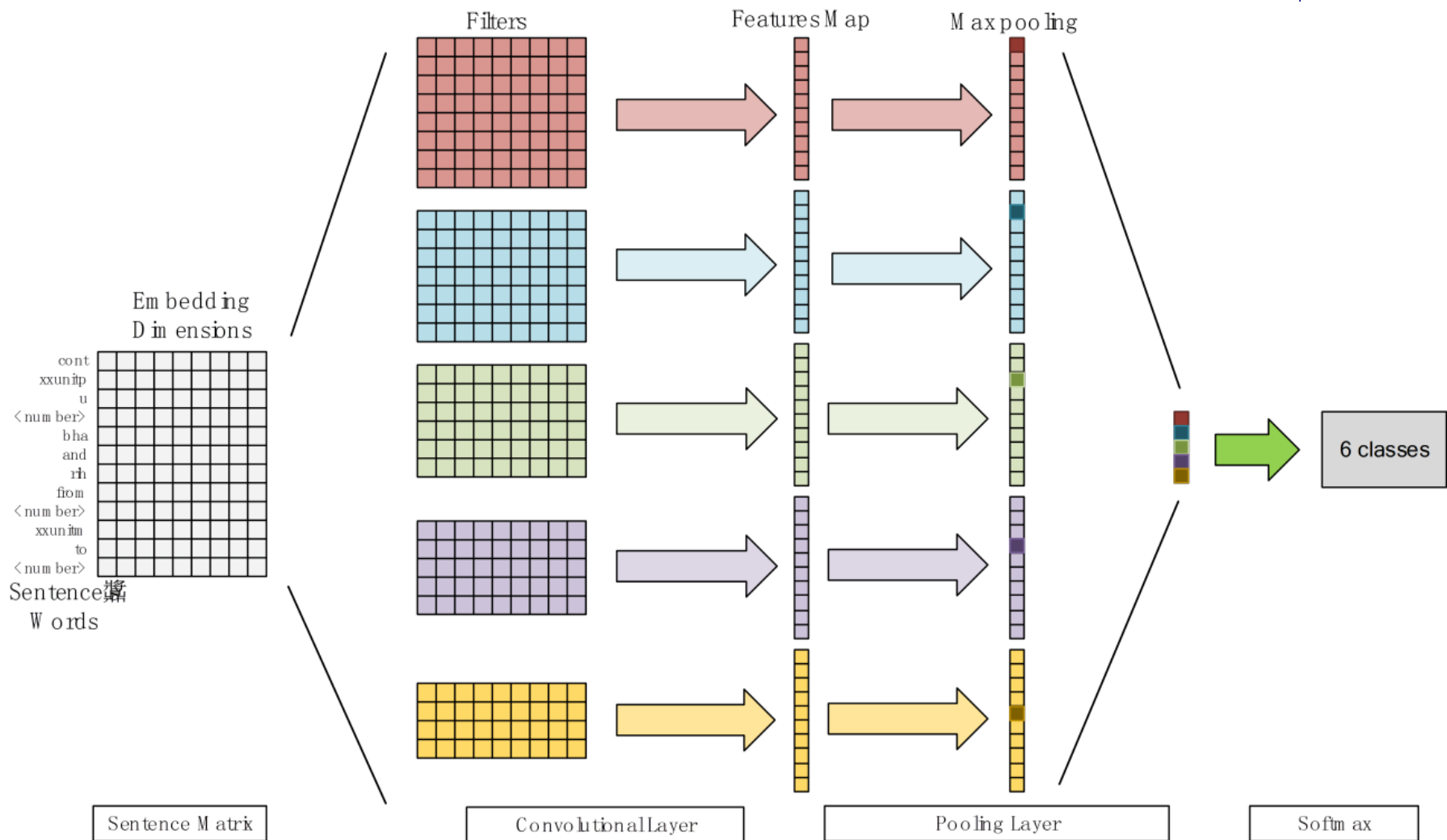


Convolutional neural networks

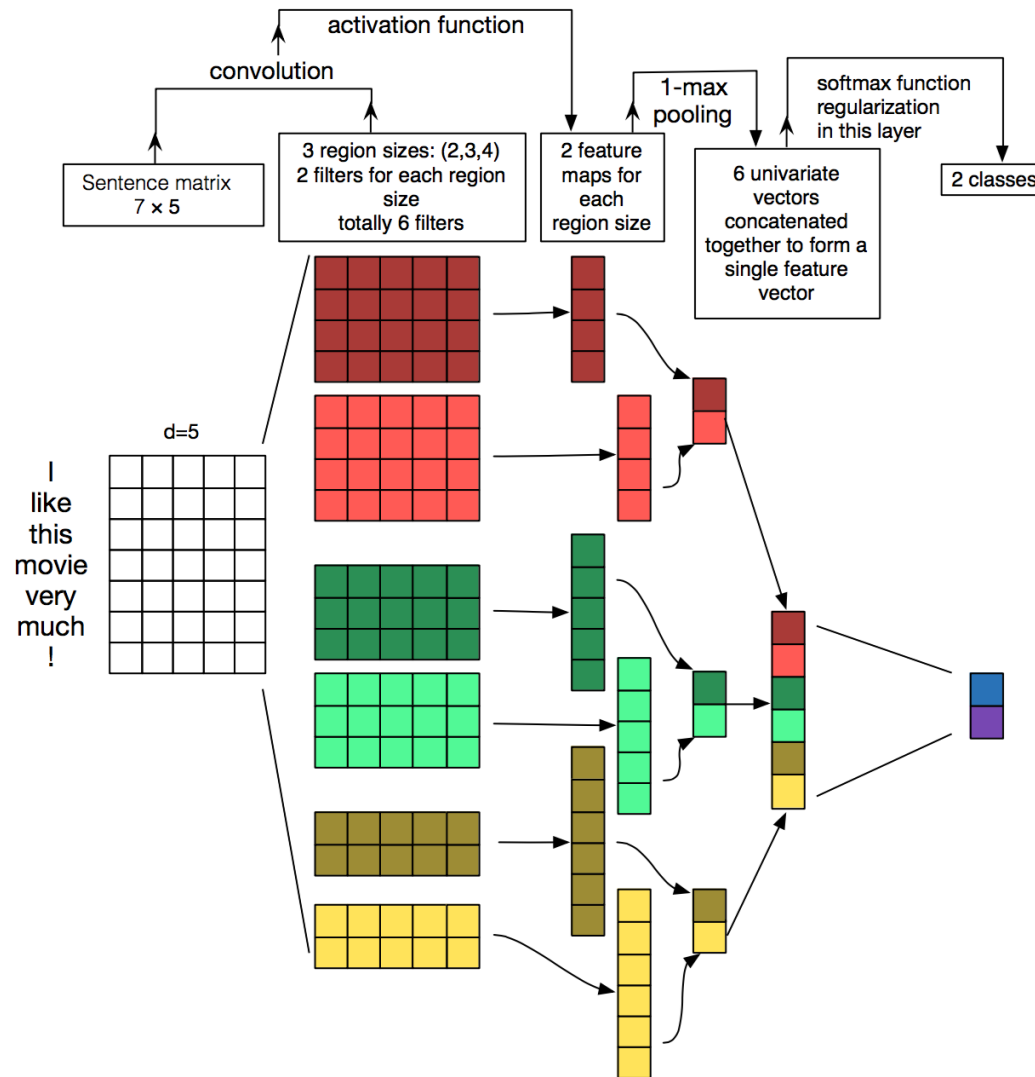
- Architecture of CNN for image classification



텍스트 분석을 위한 CNN 구조



텍스트 분석을 위한 CNN 구조



CNN 감성 분석 실습

- 데이터 읽어오기

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import seaborn as sns
train_data = pd.read_csv('labeledTrainData.tsv', header = 0,
delimiter = '\t', quoting = 3)
train_data.head()
```


CNN 감성 분석 실습

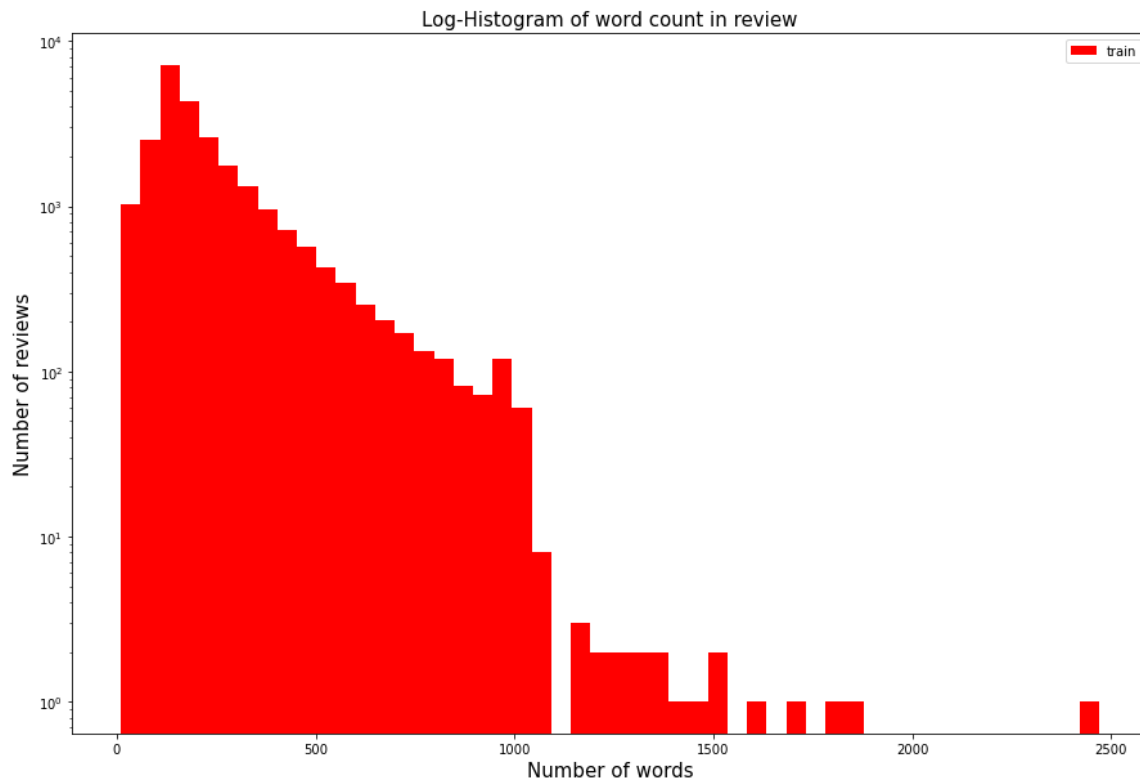
- 리뷰 단어 개수 분포 확인하기

```
temp=train_data['review'][0].split(' ')
train_word_counts = train_data['review'].apply(lambda x:len(x.split(' ')))
```

```
plt.figure(figsize=(15, 10))
plt.hist(train_word_counts, bins=50, facecolor='r', label='train')
plt.title('Log-Histogram of word count in review', fontsize=15)
plt.yscale('log', nonposy='clip')
plt.legend()
plt.xlabel('Number of words', fontsize=15)
plt.ylabel('Number of reviews', fontsize=15)
```

CNN 감성 분석 실습

- 리뷰 단어 개수 분포 확인하기



CNN 감성 분석 실습

- 리뷰 단어 개수 분포 확인하기

```
print('리뷰 단어 개수 최대 값: {}'.format(np.max(train_word_counts)))
print('리뷰 단어 개수 최소 값: {}'.format(np.min(train_word_counts)))
print('리뷰 단어 개수 평균 값:
{:.2f}'.format(np.mean(train_word_counts)))
print('리뷰 단어 개수 표준편차:
{:.2f}'.format(np.std(train_word_counts)))
print('리뷰 단어 개수 중간 값: {}'.format(np.median(train_word_counts)))
# 사분위의 대한 경우는 0~100 스케일로 되어있음
print('리뷰 단어 개수 제 1 사분위:
{}'.format(np.percentile(train_word_counts, 25)))
print('리뷰 단어 개수 제 3 사분위:
{}'.format(np.percentile(train_word_counts, 75)))
```

CNN 감성 분석 실습

- 데이터 전처리

```
import re
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
from tensorflow.python.keras.preprocessing.sequence import
pad_sequences
from tensorflow.python.keras.preprocessing.text import Tokenizer
```

CNN 감성 분석 실습

- 데이터 전처리

```
def preprocessing( review, remove_stopwords = False ):
```

```
    # 불용어 제거는 옵션으로 선택 가능하다.
```

```
    # 1. HTML 태그 제거
```

```
    review_text = BeautifulSoup(review, "html5lib").get_text()
```

```
    # 2. 영어가 아닌 특수문자들을 공백(" ")으로 바꾸기
```

```
    review_text = re.sub("[^a-zA-Z]", " ", review_text)
```

```
    # 3. 대문자들을 소문자로 바꾸고 공백단위로 텍스트들 나눠서 리스트로 만든다.
```

```
    words = review_text.lower().split()
```

CNN 감성 분석 실습

- 데이터 전처리

```
if remove_stopwords:
```

```
    # 4. 불용어들을 제거
```

```
    #영어에 관련된 불용어 불러오기
```

```
    stops = set(stopwords.words("english"))
```

```
    # 불용어가 아닌 단어들로 이루어진 새로운 리스트 생성
```

```
    words = [w for w in words if not w in stops]
```

```
    # 5. 단어 리스트를 공백을 넣어서 하나의 글로 합친다.
```

```
    clean_review = ' '.join(words)
```

```
else: # 불용어 제거하지 않을 때
```

```
    clean_review = ' '.join(words)
```

```
return clean_review
```

CNN 감성 분석 실습

- 데이터 전처리

```
clean_train_reviews = []  
for review in train_data['review']:  
    clean_train_reviews.append(preprocessing(review,  
remove_stopwords = True))
```

```
# 전처리한 데이터 출력  
clean_train_reviews[0]
```

```
clean_train_df = pd.DataFrame({'review':  
clean_train_reviews, 'sentiment': train_data['sentiment']})
```

CNN 감성 분석 실습

- 훈련 데이터와 테스트 데이터 분할

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test= train_test_split(clean_train_df['review'],  
clean_train_df['sentiment'], test_size=0.3, random_state=156)
```

```
X_train.shape, X_test.shape
```


CNN 감성 분석 실습

- 데이터 준비

```
tokenizer = Tokenizer()  
tokenizer.fit_on_texts(X_train)  
text_sequences =  
tokenizer.texts_to_sequences(X_train)  
print(text_sequences[0])
```

```
word_vocab = tokenizer.word_index  
word_vocab["<PAD>"] = 0
```

```
print("전체 단어 개수: ", len(word_vocab))
```

CNN 감성 분석 실습

- 데이터 준비

데이터 패딩

```
MAX_SEQUENCE_LENGTH = 174
```

```
train_inputs = pad_sequences(text_sequences,  
maxlen=MAX_SEQUENCE_LENGTH, padding='post')  
print('Shape of train data: ', train_inputs.shape)
```

레이블 정보를 numpy 배열로

```
train_labels = np.array(y_train)  
print('Shape of label tensor:', train_labels.shape)
```

CNN 감성 분석 실습

- 테스트 데이터 준비

test data encoding

```
text_sequences = tokenizer.texts_to_sequences(X_test)
test_inputs = pad_sequences(text_sequences,
                             maxlen=MAX_SEQUENCE_LENGTH, padding='post')
print('Shape of test data: ', test_inputs.shape)
```

레이블 정보를 numpy 배열로

```
test_labels = np.array(y_test)
print('Shape of label tensor:', test_labels.shape)
```

CNN 감성 분석 실습

- 필요한 패키지 импорт

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence
import pad_sequences
from tensorflow.keras.callbacks import
EarlyStopping, ModelCheckpoint
from tensorflow.keras import layers
```

CNN 감성 분석 실습

- 하이퍼파라미터 정의

```
model_name = 'cnn_classifier_en'  
BATCH_SIZE = 512  
NUM_EPOCHS = 10  
VALID_SPLIT = 0.1  
MAX_LEN = train_inputs.shape[1]
```

```
kargs = {'model_name': model_name,  
        'vocab_size': len(word_vocab),  
        'embedding_size': 128,  
        'num_filters': 100,  
        'dropout_rate': 0.5,  
        'hidden_dimension': 250,  
        'output_dimension': 1}
```

CNN 감성 분석 실습

- CNN Classifier 정의

```
class CNNClassifier(tf.keras.Model):
```

```
def __init__(self, **kwargs):
```

```
    super(CNNClassifier, self).__init__(name=kwargs['model_name'])
```

```
    self.embedding = layers.Embedding(input_dim=kwargs['vocab_size'],  
                                       output_dim=kwargs['embedding_size'])
```

```
    self.conv_list = [layers.Conv1D(filters=kwargs['num_filters'],  
                                    kernel_size=kernel_size,  
                                    padding='valid',  
                                    activation=tf.keras.activations.relu,  
                                    kernel_constraint=tf.keras.constraints.MaxNorm(max_value=3.))
```

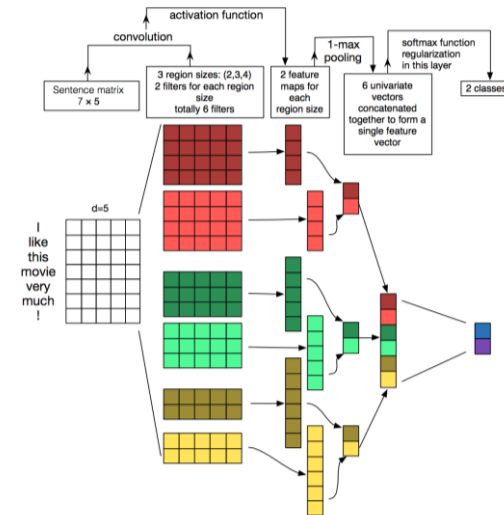
```
        for kernel_size in [3,4,5]]
```

```
    self.pooling = layers.GlobalMaxPooling1D()
```

```
    self.dropout = layers.Dropout(kwargs['dropout_rate'])
```

```
    self.fc1 = layers.Dense(units=kwargs['hidden_dimension'],  
                             activation=tf.keras.activations.relu,  
                             kernel_constraint=tf.keras.constraints.MaxNorm(max_value=3.))
```

```
    self.fc2 = layers.Dense(units=kwargs['output_dimension'],  
                             activation=tf.keras.activations.sigmoid,  
                             kernel_constraint=tf.keras.constraints.MaxNorm(max_value=3.))
```



CNN 감성 분석 실습

- CNN Classifier 정의

```
def call(self, x):  
    x = self.embedding(x)  
    x = self.dropout(x)  
    x = tf.concat([self.pooling(conv(x)) for conv in self.conv_list], axis=-1)  
    x = self.fc1(x)  
    x = self.fc2(x)  
  
    return x
```

CNN 감성 분석 실습

- 모형 생성과 학습 정의

```
model = CNNClassifier(**kargs)
model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=[tf.keras.metrics.BinaryAccuracy(name='accuracy')])
```


CNN 감성 분석 실습

- 체크포인트 정의

```
earlystop_callback = EarlyStopping(monitor='val_accuracy', \
                                    min_delta=0.0001,patience=2)
```

```
checkpoint_path = DATA_OUT_PATH + model_name + '/weights.h5'  
checkpoint_dir = os.path.dirname(checkpoint_path)
```

```
if os.path.exists(checkpoint_dir):  
    print("{} -- Folder already exists \n".format(checkpoint_dir))  
else:  
    os.makedirs(checkpoint_dir, exist_ok=True)  
    print("{} -- Folder create complete \n".format(checkpoint_dir))
```

CNN 감성 분석 실습

- 학습 정의와 학습

```
cp_callback = ModelCheckpoint(  
    checkpoint_path, monitor='val_accuracy', verbose=1, \  
    save_best_only=True, save_weights_only=True)
```

```
# 모델 학습
```

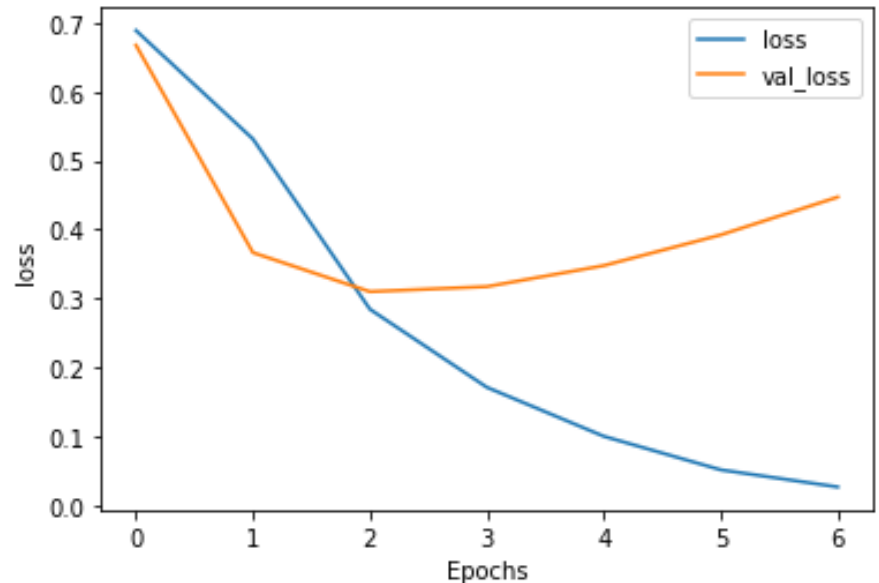
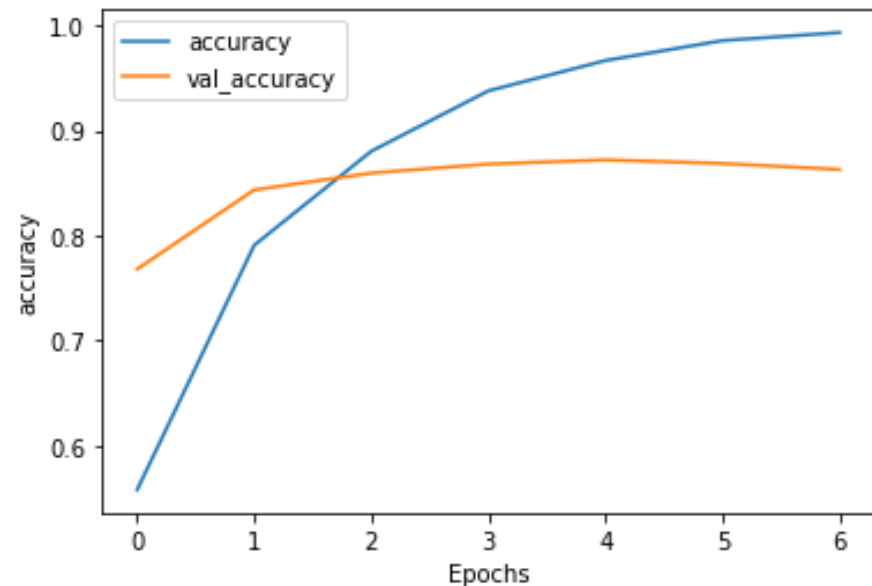
```
history = model.fit(train_input, train_label, batch_size=BATCH_SIZE,  
                    epochs=NUM_EPOCHS, validation_split=VALID_SPLIT,  
                    callbacks=[earlystop_callback, cp_callback])
```

CNN 감성 분석 실습

- 정확도와 손실함수 그리기

```
plot_graphs(history, 'accuracy')
```

```
plot_graphs(history, 'loss')
```



CNN 감성 분석 실습

- 베스트 모델 불러오기와 예측

```
SAVE_FILE_NM = 'weights.h5'
```

```
model.load_weights(os.path.join(DATA_OUT_PATH, model_name,  
SAVE_FILE_NM))
```

```
# 예측과 성능 평가
```

```
predictions = model.predict(test_inputs, batch_size=BATCH_SIZE)  
predictions = predictions.squeeze(-1)
```

```
predictions
```

```
pred=[1 if predictions[i]> 0.5 else 0 for i in range(0,predictions.shape[0])]
```

```
get_clf_eval(y_test, pred)
```

CNN 감성 분석 실습

- 성능

오차 행렬

[[3228 452]

[550 3270]]

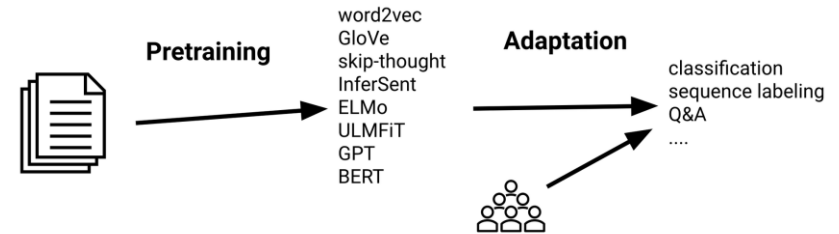
정확도: 0.8664, 정밀도: 0.8786, 재현율: 0.8560, F1: 0.8671, AUC:0.8666

BERT를 활용한 감성 분석

- 전이 학습

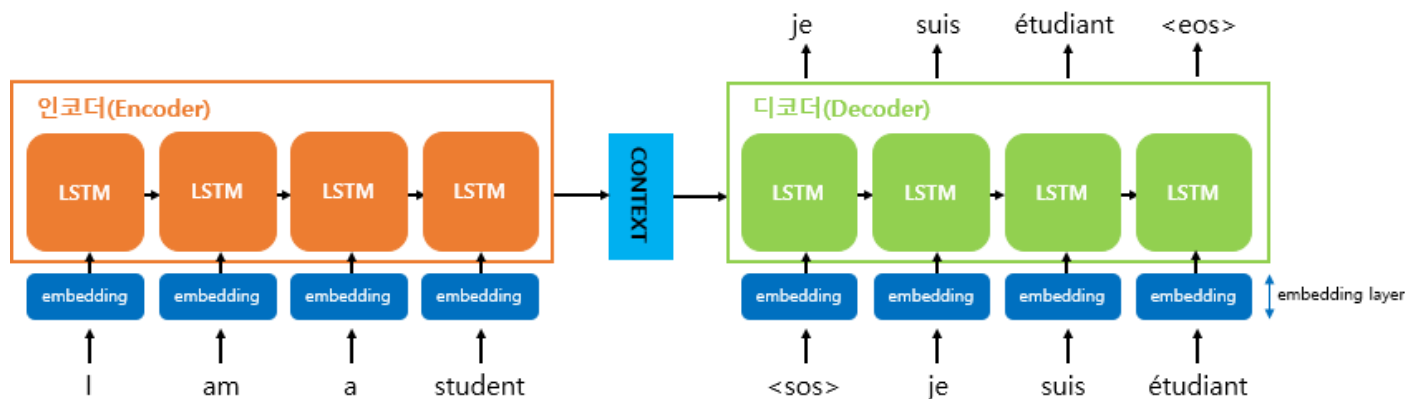
- Transfer Learning

- 훈련 데이터가 적은 경우
- 범용적인 문장들을 비지도 학습(사전 모델) –
> 특정 문제에 적합하게 추가 지도 학습
- BERT(Pre-training of Deep Bidirectional Transformers for Language Understanding)(2018)
- GPT-3(Generative Pre-Training-3, 2020)



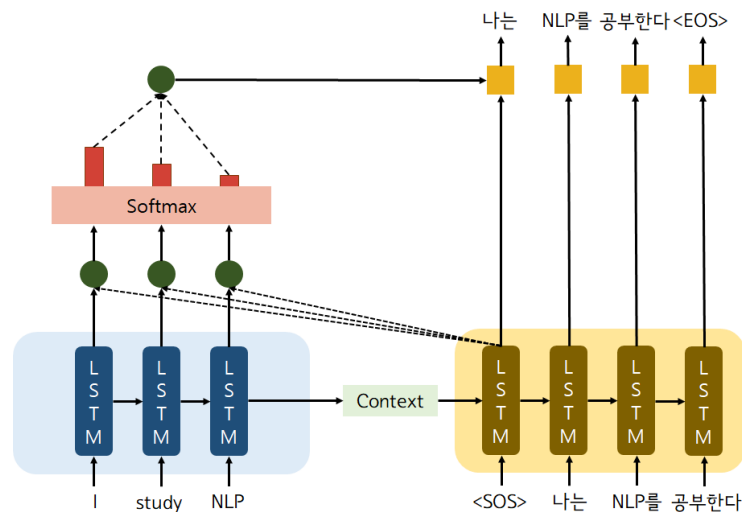
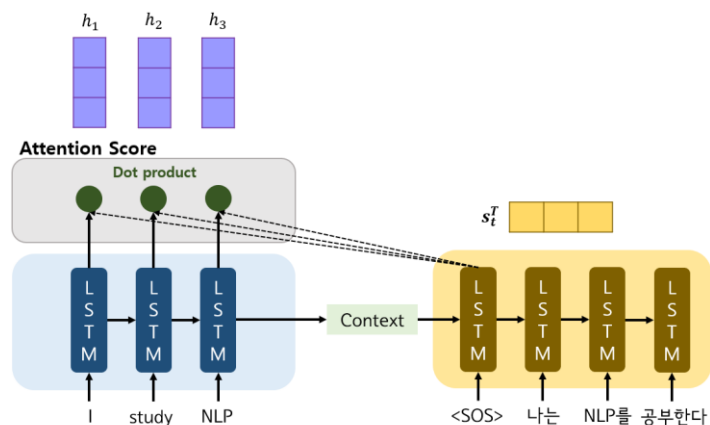
어텐션 메커니즘

- Seq2seq 모델의 한계
 - 입력 스퀀스가 **컨텍스트 벡터**라는 하나의 고정된 크기의 벡터 표현에 압축
 - 정보 손실 발생
 - 입력 문장이 길어지면 번역 품질 저하



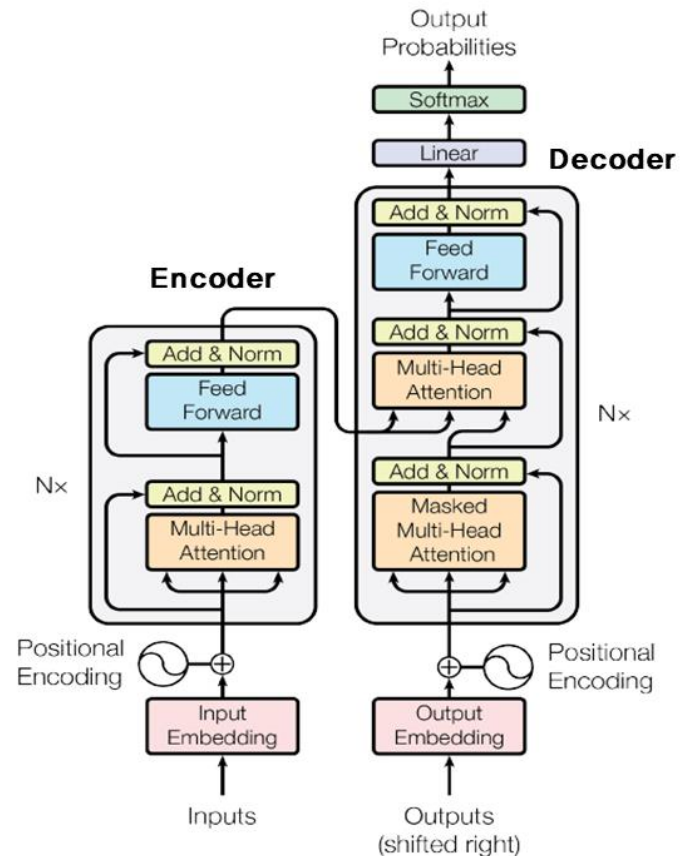
어텐션 메커니즘

- 디코더가 출력 단어를 예측하는 매 시점(time step)마다, 인코드에서 전체 입력 문장을 다시 한 번 참고
- 해당 시점에서 예측해야 할 단어와 연관이 있는 입력 단어 부분을 좀 더 **집중(attention)**해서 봄



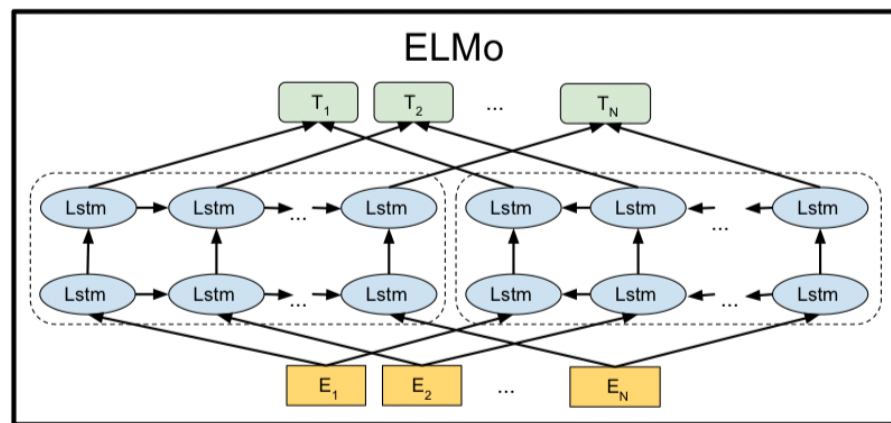
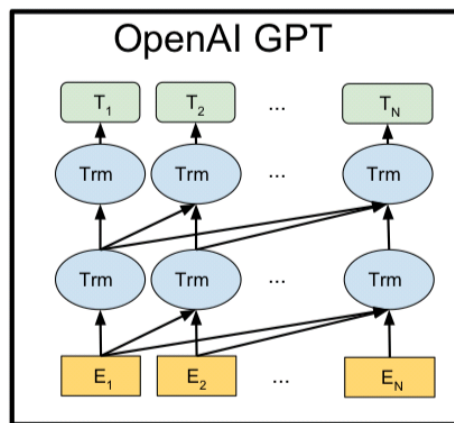
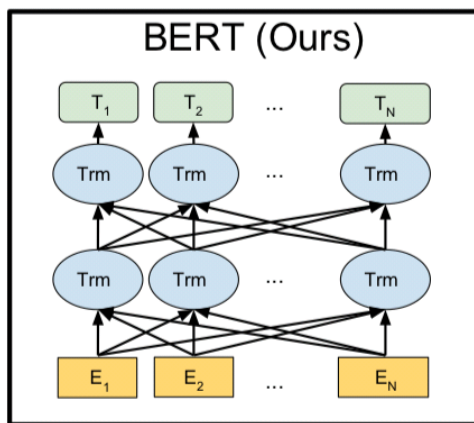
Transformer

- Attention Is All You Need(구글, 2017)

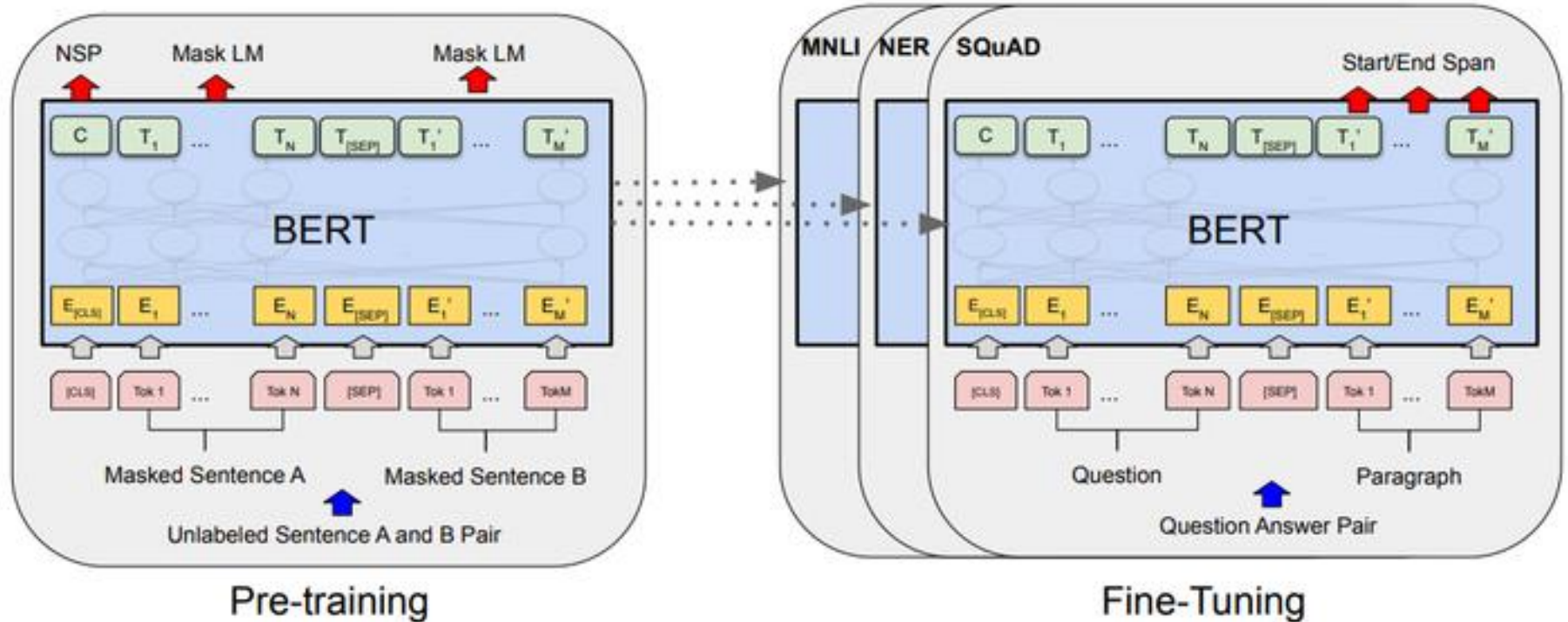


BERT

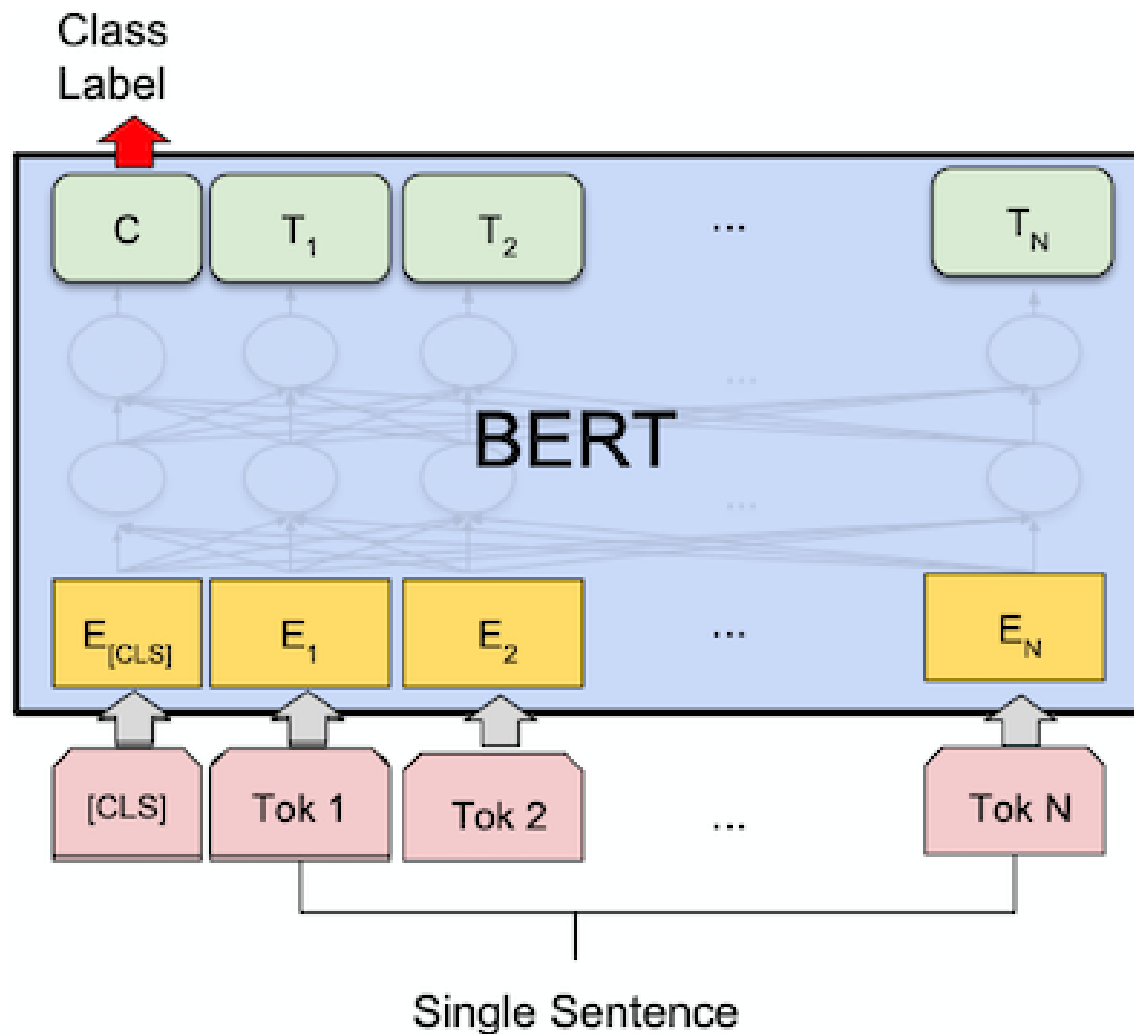
- Bidirectional Transformer
- Left-to-right Transformer
- Left-to-right, right-to-left를 독립적으로 훈련한 후에 concatenation



BERT



BERT 기반 Classification



BERT를 활용한 감성 분석

- 설치와 tensorflow импорт

```
#pip install transformers
#pip install sentencepiece
import os
import re
import numpy as np
from tqdm import tqdm
```

```
import tensorflow as tf
from transformers import *
```

```
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

BERT를 활용한 감성 분석

- 파라미터 지정

```
#random seed 고정  
tf.random.set_seed(1234)  
np.random.seed(1234)
```

```
BATCH_SIZE = 32  
NUM_EPOCHS = 3  
VALID_SPLIT = 0.2  
MAX_LEN = 39 # EDA에서 추출된 Max Length  
DATA_IN_PATH = 'data_in/BERT'  
DATA_OUT_PATH = 'data_out/BERT'
```

BERT를 활용한 감성 분석

- 사전 모델 가져오기

```
tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-cased", cache_dir='bert_ckpt', do_lower_case=False)
```

BERT를 활용한 감성 분석

- 문장 인코딩

```
test_sentence = "안녕하세요, 반갑습니다."
```

```
encode = tokenizer.encode(test_sentence)
```

```
token_print = [tokenizer.decode(token) for token in encode]
```

```
print(encode)
```

```
print(token_print)
```

```
[101, 9521, 118741, 35506, 24982, 48549, 117, 9321, 118610, 119081, 48345, 119, 102]
```

```
['[CLS]', '안', '##녕', '##하', '##세', '##요', ',', '반', '##갑', '##습', '##니다', ',', '[SEP]']
```


BERT를 활용한 감성 분석

- 문장 인코딩

```
kor_encode = tokenizer.encode("안녕하세요, 반갑습니다")  
eng_encode = tokenizer.encode("Hello world")  
kor_decode = tokenizer.decode(kor_encode)  
eng_decode = tokenizer.decode(eng_encode)
```

```
print(kor_encode)  
print(eng_encode)  
print(kor_decode)  
print(eng_decode)
```

[101, 9521, 118741, 35506, 24982, 48549, 117, 9321, 118610, 119081, 48345, 102]

[101, 31178, 11356, 102]

[CLS] 안녕하세요, 반갑습니다 [SEP]

[CLS] Hello world [SEP]

BERT를 활용한 감성 분석

- 스페셜 토큰들

```
print(tokenizer.all_special_tokens, "\n",  
tokenizer.all_special_ids)
```

모르는 단어 문장 종결 패딩 문장시작 마스크 토큰, 사전 학습에서 사용

```
['[UNK]', '[SEP]', '[PAD]', '[CLS]', '[MASK]']  
[100, 102, 0, 101, 103]
```

```
[101, 9521, 118741, 35506, 24982, 48549, 117, 9321, 118610, 119081, 48345, 102]  
[101, 31178, 11356, 102]  
[CLS] 안녕하세요, 반갑습니다 [SEP]  
[CLS] Hello world [SEP]
```

BERT를 활용한 감성 분석

- 버트에 들어갈 입력 만들기

```
def bert_tokenizer(sent, MAX_LEN):
```

```
    encoded_dict = tokenizer.encode_plus(  
        text = sent,  
        add_special_tokens = True, # Add '[CLS]' and '[SEP]'  
        max_length = MAX_LEN,      # Pad & truncate all sentences.  
        pad_to_max_length = True,  
        return_attention_mask = True # Construct attn. masks.  
    )
```

```
    input_id = encoded_dict['input_ids']  
    attention_mask = encoded_dict['attention_mask']  
        # And its attention mask (simply differentiates padding
```

```
from non-padding).
```

```
    token_type_id = encoded_dict['token_type_ids'] # differentiate two sentences
```

```
    return input_id, attention_mask, token_type_id
```

BERT를 활용한 감성 분석

- 데이터 전처리

```
input_ids = []
attention_masks = []
token_type_ids = []
train_data_labels = []

for train_sent, train_label in tqdm(zip(X_train, y_train), total=len(X_train)):
    try:
        input_id, attention_mask, token_type_id = bert_tokenizer(train_sent,
                                                                    MAX_LEN)

        input_ids.append(input_id)
        attention_masks.append(attention_mask)
        token_type_ids.append(token_type_id)
        train_data_labels.append(train_label)

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

BERT를 활용한 감성 분석

- 데이터 전처리

```
train_input_ids = np.array(input_ids, dtype=int)
train_attention_masks = np.array(attention_masks, dtype=int)
train_type_ids = np.array(token_type_ids, dtype=int)
train_inputs = (train_input_ids, train_attention_masks, train_type_ids)

train_data_labels = np.asarray(train_data_labels, dtype=np.int32) #레이블 토큰나이징 리스트

print("# sents: {}, # labels: {}".format(len(train_input_ids), len(train_data_labels)))
```

```
# sents: 17500, # labels: 17500
```

BERT를 활용한 감성 분석

- 전처리된 데이터 확인하기

```
input_id = train_input_ids[1]
attention_mask = train_attention_masks[1]
token_type_id = train_type_ids[1]
```

```
print(input_id)
print(attention_mask)
print(token_type_id)
print(tokenizer.decode(input_id))
```

BERT를 활용한 감성 분석

- BERTClassifier 정의

```
class TFBertClassifier(tf.keras.Model):  
    def __init__(self, model_name, dir_path, num_class):  
        super(TFBertClassifier, self).__init__()  
  
        self.bert = TFBertModel.from_pretrained(model_name,  
cache_dir=dir_path)  
        self.dropout =  
tf.keras.layers.Dropout(self.bert.config.hidden_dropout_prob)  
        self.classifier = tf.keras.layers.Dense(num_class,  
kernel_initializer=tf.keras.initializers.TruncatedNormal(self.bert.config.initializer_range), name="classifier")
```

BERT를 활용한 감성 분석

- BERTClassifier 정의

```
def call(self, inputs, attention_mask=None, token_type_ids=None,
training=False):

    #outputs 값: # sequence_output, pooled_output, (hidden_states),
    (attentions)
    outputs = self.bert(inputs, attention_mask=attention_mask,
token_type_ids=token_type_ids)
    pooled_output = outputs[1]
    pooled_output = self.dropout(pooled_output, training=training)
    logits = self.classifier(pooled_output)

    return logits
```


BERT를 활용한 감성 분석

- BERTClassifier 인스턴스 생성

```
cls_model = TFBertClassifier(model_name='bert-base-multilingual-cased',  
                             dir_path='bert_ckpt',  
                             num_class=2)
```

BERT를 활용한 감성 분석

- 학습 준비 하기

```
optimizer = tf.keras.optimizers.Adam(3e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
cls_model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```

BERT를 활용한 감성 분석

- Early Stopping 정의

```
model_name = "tf2_bert_review"
```

```
# overfitting을 막기 위한 earlrystop 추가
```

```
earlystop_callback = EarlyStopping(monitor='val_accuracy', min_delta=0.0001,patience=2)
```

```
# min_delta: the threshold that triggers the termination (acc should at least improve 0.0001)
```

```
# patience: no improvment epochs (patience = 1, 1번 이상 상승이 없으면 종료)\
```

```
checkpoint_path = os.path.join(DATA_OUT_PATH, model_name, 'weights.h5')
```

```
checkpoint_dir = os.path.dirname(checkpoint_path)
```

BERT를 활용한 감성 분석

- Early Stopping 정의

```
# Create path if exists
if os.path.exists(checkpoint_dir):
    print("{} -- Folder already exists \n".format(checkpoint_dir))
else:
    os.makedirs(checkpoint_dir, exist_ok=True)
    print("{} -- Folder create complete \n".format(checkpoint_dir))

cp_callback = ModelCheckpoint(
    checkpoint_path, monitor='val_accuracy', verbose=1,
    save_best_only=True, save_weights_only=True)
```

BERT를 활용한 감성 분석

- 학습과 평가

```
history = cls_model.fit(train_inputs, train_data_labels, epochs=NUM_EPOCHS,  
                        batch_size=BATCH_SIZE,  
                        validation_split = VALID_SPLIT, callbacks=[earlystop_callback, cp_callback])
```

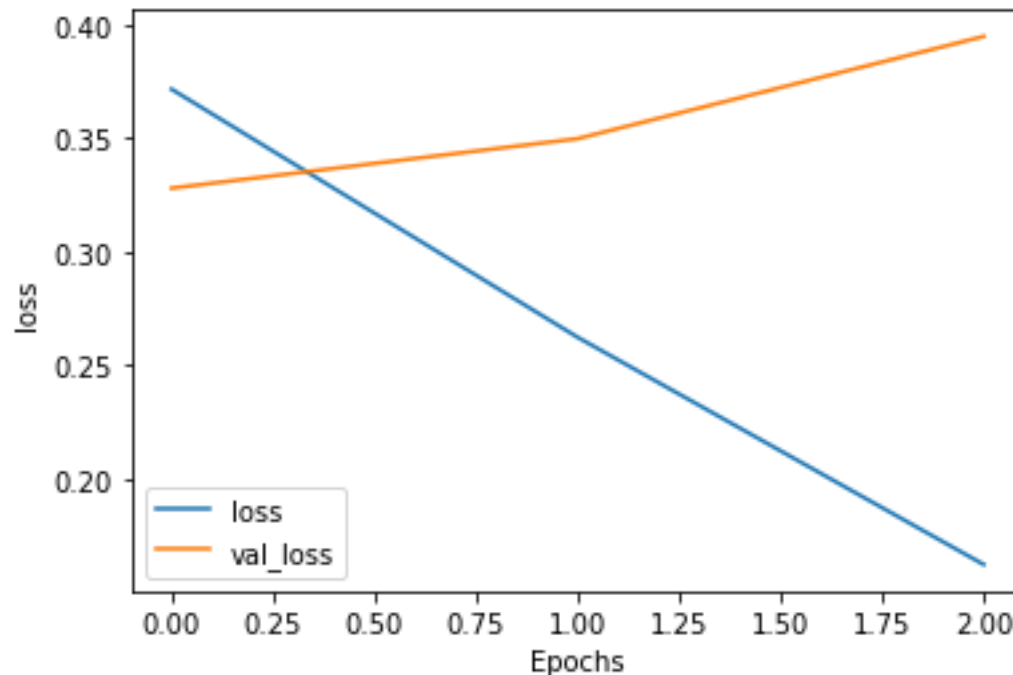
```
#steps_for_epoch
```

```
print(history.history)
```

BERT를 활용한 감성 분석

- 그래프 그리기

`plot_graphs(history, 'loss')`



BERT를 활용한 감성 분석

- 테스트 데이터 전처리

```
input_ids = []
attention_masks = []
token_type_ids = []
test_data_labels = []

for test_sent, test_label in tqdm(zip(X_test, y_test)):
    try:
        input_id, attention_mask, token_type_id = bert_tokenizer(test_sent, MAX_LEN)
        input_ids.append(input_id)
        attention_masks.append(attention_mask)
        token_type_ids.append(token_type_id)
        test_data_labels.append(test_label)
    except Exception as e:
        print(e)
        print(test_sent)
        pass
```

BERT를 활용한 감성 분석

- 테스트 데이터 전처리

```
test_input_ids = np.array(input_ids, dtype=int)
test_attention_masks = np.array(attention_masks, dtype=int)
test_type_ids = np.array(token_type_ids, dtype=int)
test_inputs = (test_input_ids, test_attention_masks, test_type_ids)

test_data_labels = np.asarray(test_data_labels, dtype=np.int32) #레이블 토큰나이징
리스트

print("num sents, labels {}, {}".format(len(test_input_ids), len(test_data_labels)))
```


BERT를 활용한 감성 분석

- 성능 평가

```
results = cls_model.evaluate(test_inputs, test_data_labels,  
batch_size=1024)  
print("test loss, test acc: ", results)
```

test loss, test acc: [0.39382049441337585, 0.8493333458900452]

BERT를 활용한 감성 분석

- 예측

```
pred=cls_model.predict(test_inputs)
print(pred)
class_pred=[1 if pred[i][1]> pred[i][0] else 0 for i in range(0,pred.shape[0])]
get_clf_eval(y_test, class_pred)
```

오차 행렬

[[2935 745]

[385 3435]]

정확도: 0.8493, 정밀도: 0.8218, 재현율: 0.8992,

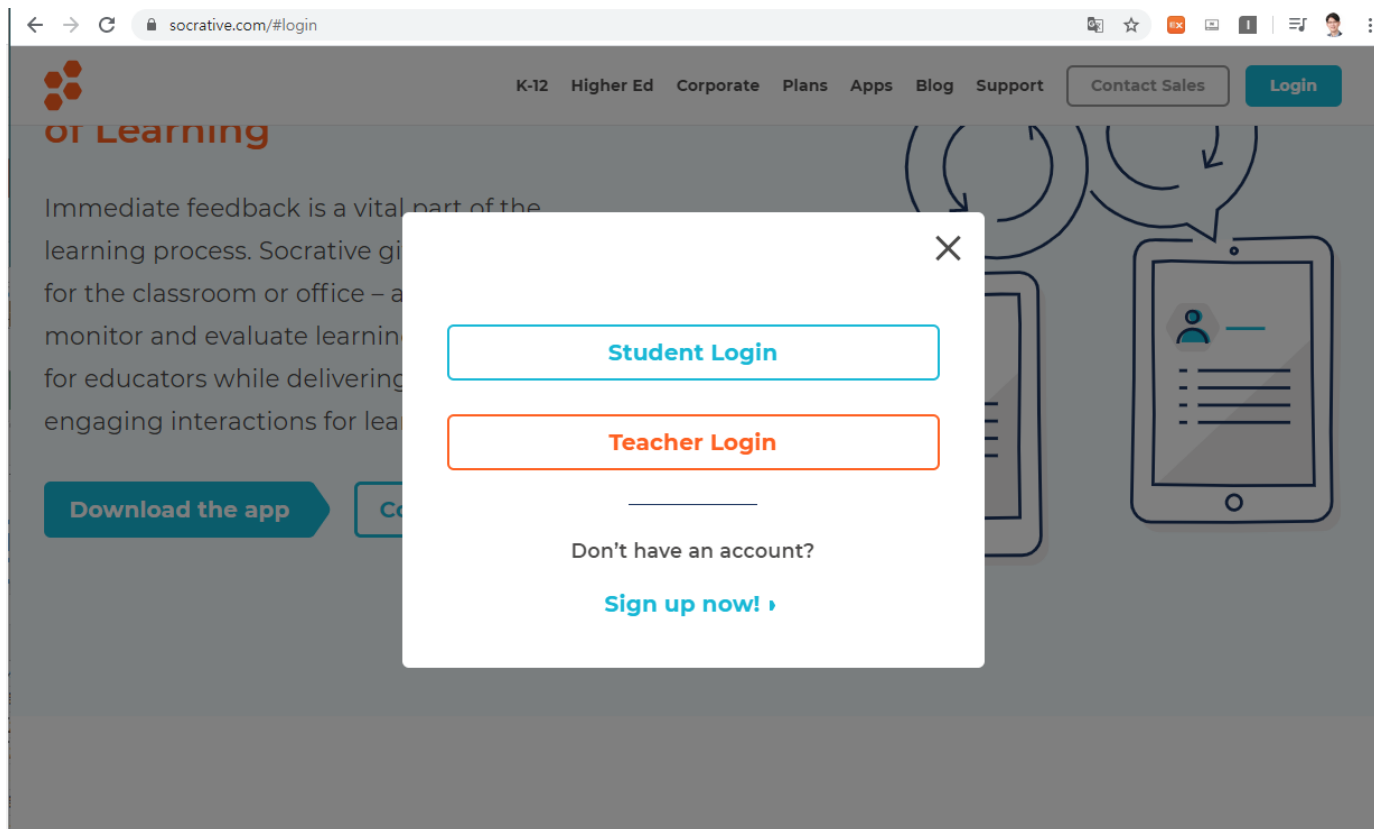
F1: 0.8588, AUC:0.8484

기말고사 안내

- 12월 20일(월) 10시부터 12시 50분까지
- 전체 온라인 시험
- PC에서 **Socrative SW** 사용
 - Socrative.com
 - Room: COE8043
 - Login: 자신의 학번
- 핸드폰에서 **LMS 16주차 기말고사** 세션(zoom) 사용
 - 줌에는 시험 10분 전에 입장할 것

PC 웹 브라우저에서


- 10시 정각에 socrative.com 접속후 - Login 클릭



PC 웹 브라우저에서

회의 정보 - Zoom x | Home - Socrative x | Socrative x +

← → ↻ 🔒 b.socrative.com/login/student/ 🔍 ☆ EX 📺 📄 ⚙️ 👤 ⋮


 socrative

Student Login

Room Name

COE8043

JOIN

 English ▾

Socrative

PC 웹 브라우저에서



Student Login

Room Name

COE8043

[Change Room](#)



This room requires a student ID. Please enter your student ID to continue.

Student ID

SUBMIT

자신의 학번 입력할 것

장애 발생 시 이메일:
김종우 교수
kju@hanyang.ac.kr
성민 조교
min91155@gmail.com

PC 웹 브라우저에서

- 시험 코딩 중 필요한 데이터 집합은 learning.hanyang.ac.kr에 16주차 강의자료로 제공 예정

202120HY33752_빅데이터분석 > 202120HY33752_빅데이터분석

2021년 2학기

홈

사용자 및 그룹

수업 계획서

공지

문의게시판

강의콘텐츠

과제 및 평가

시험 및 설문

토론

강의자료실

출결/학습 현황

성적

종합성적부

학습설계진단

ClassMix

파일

학습 분석

설정

모든 주차 보기

페이지 분할/병합 모드

OFF

주차 일정 일괄 변경

기존화면으로 전환

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16

Term Project 최종보고서 제출

시작 12월 6일 00:00 - 마감 12월 12일 23:59

D-0

열람기한 12.31 23:59

10 점

Term Project 동료 평가 결과 제출

시작 12월 6일 00:00 - 마감 12월 12일 23:59

D-0

열람기한 12.31 23:59

10 점

프로젝트발표검토의견양식

시작 12월 6일 00:00 - 마감 12월 12일 23:59

열람기한 12.31 23:59

2차시 / Lesson-2 실시간화상강의

→ 내 콘텐츠 가져오기

↑ 새 콘텐츠

학습활동

빅데이터분석 15주차 실시간 수업

시작 12월 13일 00:00

진행시간 1시간 20분

16주차 / 기말고사

1차시 / 기말고사

→ 내 콘텐츠 가져오기

↑ 새 콘텐츠

학습활동

빅데이터분석 기말고사

시작 12월 20일 10:00

진행시간 2시간 50분

2차시 / Lesson-2

→ 내 콘텐츠 가져오기

↑ 새 콘텐츠

학습활동

학습 요소를 추가해주세요.

핸드폰에서

- 시험 10분전 접속
- learning.hanyang.ac.kr

The screenshot shows the web interface of learning.hanyang.ac.kr. The browser address bar displays 'learning.hanyang.ac.kr/courses/75292/external_tools/1'. The page title is '202120HY33752_빅데이터분석 > 202120HY33752_빅데이터분석'. A horizontal row of 16 numbered boxes (01 to 16) is visible, with box 08 circled in red. Below this, the section '08 | 8주차 / 중간고사' is expanded, showing '1차시 / 중간고사'. A red oval highlights a 'Live Lecture' icon and the text '빅데이터분석 중간고사 세션' with a subtitle '학생 강의 | 시작일: 10월 25일 오전 10:00 | 진행시간: 2시간 40분'. The page also shows navigation links like '홈', '사용자 및 그룹', '수업 계획서', '공지', '문의게시판', '강의콘텐츠', '과제 및 평가', '시험 및 설문', '토론', '강의자료실', '출결/학습 현황', '성적', '종합성적부', '학습설계진단', 'ClassMix', '파일', '학습 분석', and '설정'.

핸드폰에서

learning.hanyang.ac.kr/courses/75292/external_tools/1

202120HY33752_빅데이터분석 > 202120HY33752_빅데이터분석

2021년 2학기

계정
대시보드
과목
캘린더
메시지함
이용안내

홈
사용자 및 그룹
수업 계획서
공지
문의게시판
강의콘텐츠
과제 및 평가
시험 및 실문
토론
강의자료실
출결/학습 현황
성적
종합성적부
학습설계진단
ClassMix
파일
학습 분석
설정

목록으로 돌아가기

목차 < 이전 학습 학습 종료

1페이지 페이지 제목 수정

빅데이터분석 중간고사 세션

시작일 10월 25일 오전 10:00 | 진행시간 2시간 40분

화상 강의 시작하기

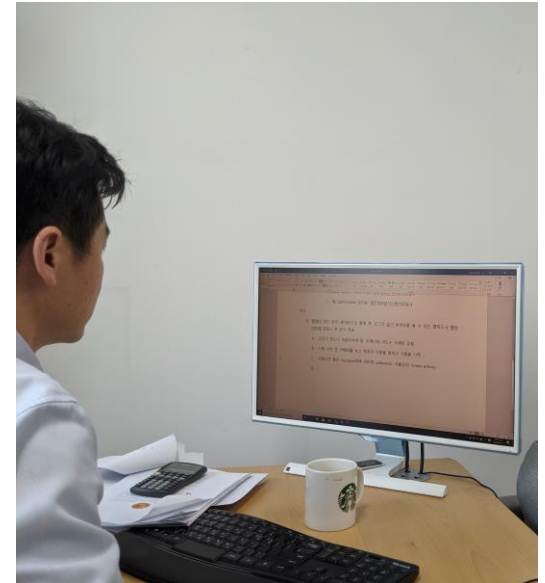
출결 기록 확인

* 화상 강의를 진행한 후에 출결 기록을 확인할 수 있습니다.
(화상 강의 종료 후 30분이 지나야 출결 기록을 확인할 수 있습니다. 경우에 따라 일정 시간이 더 소요될 수 있습니다.)

< 이전 학습 학습 종료

핸드폰에서

- 회의 입장 후 오디오연결-인터넷 전화 선택
- ‘비디오 시작’ 클릭
- 오디오 켜기
- 시험 시작 전에 조교가 학번, 이름, 얼굴을 확인합니다.
- 얼굴과 화면이 함께 보이도록 핸드폰을 설치하고 시험을 봅니다
- 시험 중에는 조교의 지시에 따르도록 합니다.



시험 중 핸드폰 설치 예