

KUBIG Contest

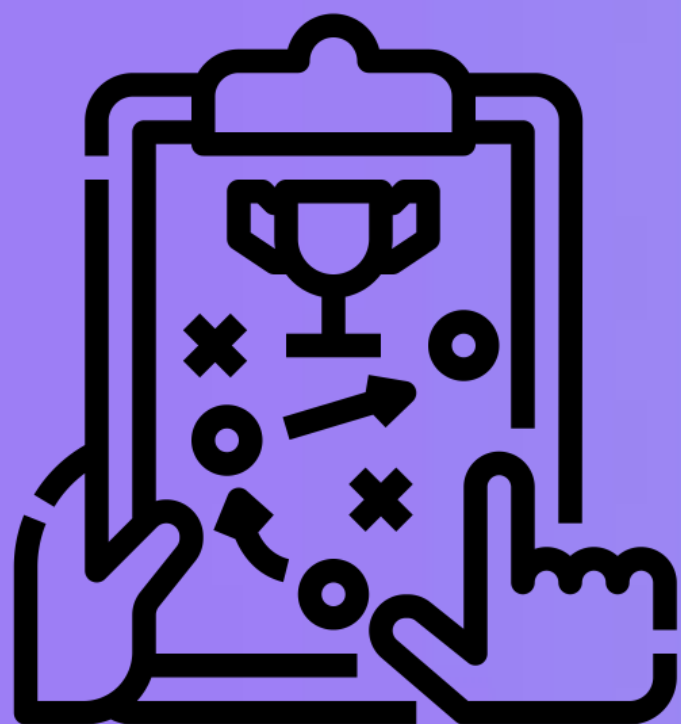
# 월간 뉴스 토픽 분류 AI 경진대회

With LSTM , Klue/Roberta-Large

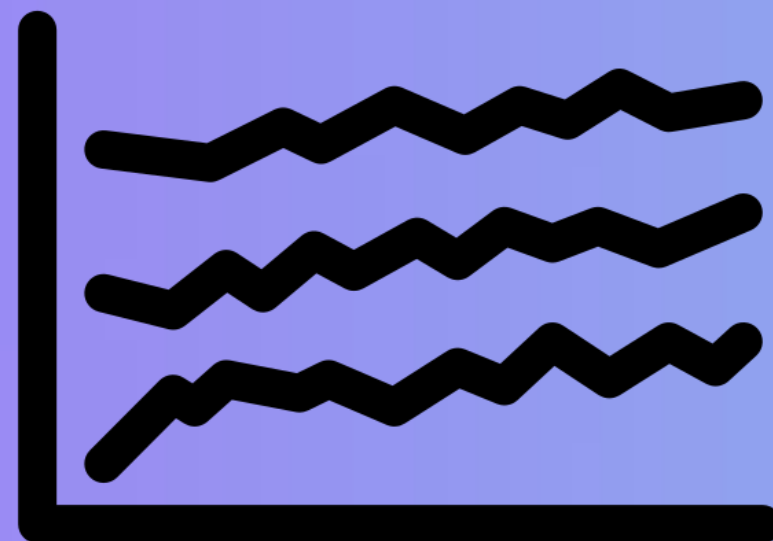


DL 2팀

원준혁 /우다경 /18기 정해원



분석 배경 및 목표



데이터 전처리



모델링



결과 / 해석

# 분석 배경 및 목표

# 텍스트 주제 추론

= 언어 이해 시스템이 보유해야 할 핵심 기능

한국어 자연어 데이터를 바탕으로  
주제를 추론/분류하는 알고리즘의 개발 필요

# Deep Learning을 기반으로 NLP 알고리즘을 개발

YNAT (주제 분류를 위한 연합뉴스 헤드라인) 데이터 세트를 활용

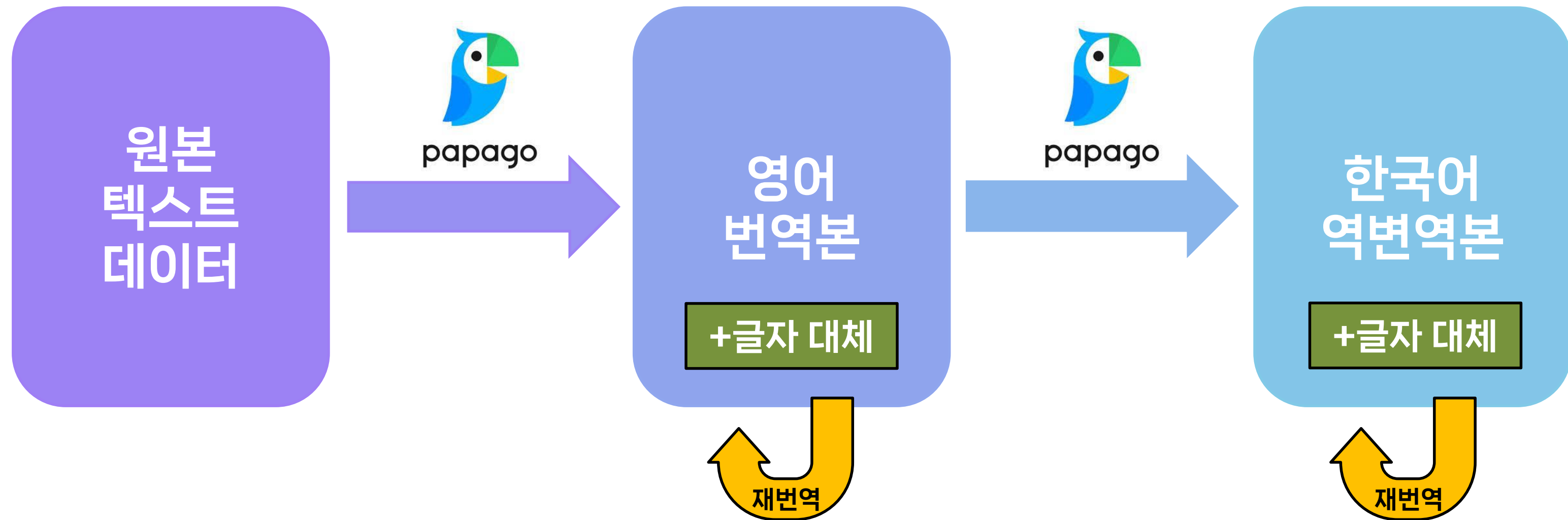
KLUE (Korean Language Understanding Evaluation) 데이터 세트를 이용  
다양한 언어 모델의 성능을 비교

한국어 뉴스 헤드라인을 이용하여,  
뉴스의 주제를 분류하는 알고리즘 개발

# 데이터 전처리

## 02 데이터 전처리

각 단계에서 네이버 파파고를 이용해서 번역



%, 한자 등의 특수 문자 → 한글/영문으로 대체

각 단계에서 결측치 발생시, **번역 시간을 늘려서 재번역**

## 02 데이터 전처리

```
[ ] 1 # Crawling 한->영 번역
2 def kor_to_trans(text_data, trans_lang,start_index,final_index):
3
4     target_present = EC.presence_of_element_located((By.XPATH, '//*[@id="txtTarget"]'))
5
6     for i in tqdm(range(start_index,final_index)):
7
8         if (i!=0)&(i%99==0):
9             time.sleep(0.05)
10            np.save(data_path+'kor_to_eng_train_{}_{}.npz'.format(start_index,final_index),trans_list)
11        elif i==(final_index) :
12            time.sleep(0.05)
13            np.save(data_path+'kor_to_eng_train_{}_{}.npz'.format(start_index,final_index),trans_list)
14
15        try:
16            driver.get('https://papago.naver.com/?sk=ko&tk='+trans_lang+'&st='+text_data[i])
17            time.sleep(1.5)
18            element=WebDriverWait(driver, 10).until(target_present)
19            time.sleep(0.1)
20            backtrans = element.text
21
22            if (backtrans=='')|(backtrans==' '):
23                element=WebDriverWait(driver, 20).until(target_present)
24                backtrans = element.text
25                trans_list.append(backtrans)
26            else:
27                trans_list.append(backtrans)
28
29        except:
30            trans_list.append('')
```

```
▶ 1 a=train.shape[0]
2 trans_list=[]
3 kor_to_trans(train['title'], 'en',0,a)
4 np.save(data_path+'kor_to_eng_train.npz',trans_list)
5 # 실제 데이터는 너무 커서 10000개씩 구간을 나눠서 번역 후 결합
6 # time sleep이나 wait time이 짧아 결측치가 생기는 문제가 발생하여 그 시간을 늘림
```

번역 : KOR → ENG

```
▶ 1 def en_to_trans(text_data, trans_lang,start_index,final_index):
2
3     target_present = EC.presence_of_element_located((By.XPATH, '//*[@id="txtTarget"]'))
4
5     for i in tqdm(range(start_index,final_index)):
6
7         if (i!=0)&(i%99==0):
8             time.sleep(0.05)
9             np.save(data_path+'eng_to_kor_train_{}_{}.npz'.format(start_index,final_index),trans_list)
10        elif i==(final_index) :
11            time.sleep(0.05)
12            np.save(data_path+'eng_to_kor_train_{}_{}.npz'.format(start_index,final_index),trans_list)
13
14        try:
15            driver.get('https://papago.naver.com/?sk=en&tk='+trans_lang+'&st='+text_data[i])
16            time.sleep(1.5)
17            element=WebDriverWait(driver, 10).until(target_present)
18            time.sleep(0.1)
19            backtrans = element.text
20
21            if (backtrans=='')|(backtrans==' '):
22                element=WebDriverWait(driver, 20).until(target_present)
23                backtrans = element.text
24                trans_list.append(backtrans)
25            else:
26                trans_list.append(backtrans)
27
28        except:
29            trans_list.append('')
```

```
[ ] 1 a=train_eng.shape[0]
2 trans_list=[]
3 en_to_trans(train_eng['title_en'], 'ko',0,a)
4 np.save(data_path+'eng_to_kor_train.npz',trans_list)
```

역번역 : ENG → KOR



## 02 데이터 전처리

```
22126th : Criticism of God's abandonment to Prague, which has severed its sisterhood relationship with Beijing
22425th : Hyundai Opens Premium Outlet in Namyangju in 2019
22724th : Although he caught a monster, IS forces were reinforced by Trump's withdrawal
23023th :
23322th :
23621th : FIFA expanded to 48 participating countries from the 2026 World Cup.....
23920th : Let's compete for coverage, not KT 5G subsidies...A goal of 1.5 million people at the end of the year
24219th :
24518th : It's raining... Cheonan-Asan Spring Flower Scenery Minister Next Month
24817th : U.S. teen iPhone preference intensifies
25116th :
25415th : Minister Yoo Young-min emphasizes the need to cut telecommunication costs to KT Chairman Hwang Chang-kyu
25714th :
26013th : Weekend N trip Gangwon-do Province's leisure sports stunt that will blow away the late heat
```

**Q) 번역 돌리고 중간에 발생한 결측치의 처리?**

```
15     try:
16         driver.get('https://papago.naver.com/?sk=ko&tk='+trans_lang+'&st='+text_data[i])
17         time.sleep(1.5)
18         element=WebDriverWait(driver, 10).until(target_present)
19         time.sleep(0.1)
20         backtrans = element.text
21
```

**번역 시간을 늘린다(10초)**

## 02 데이터 전처리

```
▶ 1 ### 역번역 자료에서 한자 및 특정 문자 변환
2 train_final["title_kor"] = train_final["title_kor"].apply(ch2_eda)
3 train_final["title_kor"] = train_final["title_kor"].apply(per2_eda)
```

```
[ ] 1 ### 특수문자, 한자 등 없애기
2 import re
3
4 def remove_chr(str):
5
6     new = re.sub(r'[^a-zA-Z가-힣0-9#s]', '', str) #
7     return new
8
9 # train_final 데이터에 적용
10 train_final1 = train_final
11 temp = train_final1['title_kor'].apply(remove_chr)
12 temp2 = train_final1['title_en'].apply(remove_chr)
13 temp3 = train_final1['title'].apply(remove_chr)
14 train_final1['title'] = temp3
15 train_final1['title_kor'] = temp
16 train_final1['title_en'] = temp2
17
18 train_final1
```

```
[ ] 1 def ch_eda(title) :
2     title = title.replace("英", "Great Britain")
3     title = title.replace("美", "the United States")
4     title = title.replace("伊", "Italy")
5     title = title.replace("與", "the rulling party")
6     title = title.replace("野", "the opposition party")
7     title = title.replace("獨", "Germany")
8     title = title.replace("女", "woman")
9     title = title.replace("親", "friendly")
10    title = title.replace("亞", "Asia")
11    title = title.replace("反", "opposing")
12    title = title.replace("印", "India")
13    title = title.replace("檢", "the prosecution")
14    title = title.replace("佛", "France")
15    return title
```

```
▶ 1 # 영-한 역번역 시 한자의 경우 제대로 번역이 되지 않는 문제가 발생
2 # 한자가 포함된 데이터 확인 후 ch_eda 적용 후 재번역 실시
3 def contains_hanja(text):
4     hanja_range = (0x4E00, 0x9FFF) # 한자 유니코드 범위
5     for char in text:
6         if hanja_range[0] <= ord(char) <= hanja_range[1]:
7             return True
8     else:
9         return False
10
11 kor_data2_hanja = train_eng[train_eng['title_en'].apply(contains_hanja)]
```

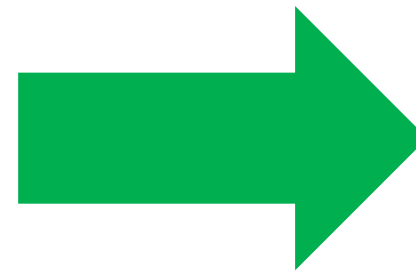
```
[ ] 1 # 한자어 포함 데이터프레임 추출
2 example_list = ["英", "美", "伊", "獨", "野", "與", "女", "親", "亞", "反", "印", "檢", "佛"]
3 test = '|'.join(example_list)
4 train_eng_c = train_eng[train_eng["title_en"].str.contains(test)]
```

역번역 과정에서 번역되지 않은 한자들을 인위적으로 번역하여 전처리

## 02 데이터 전처리

```
1 # 전처리
2 def word_delete(title):
3     delete_word = [
4         "1보",
5         "2보",
6         "3보",
7         "4보",
8         "5보",
9         "6보",
10        "게시판",
11        "신간",
12        "그래픽",
13        "속보",
14        "영상",
15        "...",
16        "주말 N",
17        "QA",
18        "동정",
19        "위클리",
20        "주간 화제의 뉴스",
21        "카드뉴스",
22        "팩트체크",
23        ##### 스포츠 10대 뉴스, 관련주 등 키워드 추론 가능 내용은 제거하지 않음
24    ]
25    for i in delete_word:
26        if title.endswith(i) or title.startswith(i):
27            title = title.replace(i, "")
28        if title.endswith("종합"):
29            title = title[:-2]
30        title = title.replace("↑", " 증가")
31        title = title.replace("↓", " 감소")
32        title = title.replace("→", "에서 ")
33        title = title.replace("~", "에서 ")
34        title = title.replace("·", " 그리고 ")
35        title = title.replace(":", " 대 ")
36        title = title.replace("%", "퍼센트")
37        title=title.replace("...", " ")
38        title=title.replace(".", " ")
39        title=title.replace("&", "그리고")
40    return title
```

불용어 제거



title\_kor

인천발 핀란드 항공기 결항 명절 여행객 분노

실리콘밸리 넘어 구글 **15조원** 들여 미국 글로벌 허브로  
도약

이란 외무부 긴장완화 해법은 미국이 경제전쟁을 멈추  
는 것입니다

**NYT** 클린턴 한국 기업 최측근 특수관계 부각 공과 연계

시진핑은 가능한 한 빨리 중미 무역 협상을 타결하기

...

**KB금융** 미국 **IB** 스티펠과 제휴 선진시장 공략

서울시교육청 코로나**19** 확산에 개 그리고 폐교 연기 검  
토

키움증권 **2020** 키움영웅대회 실물투자대회

배기동 국립중앙박물관장 답변

다음달 **1일 2020** 한국인터넷기자대상 시상식 특별상  
김성후

전처리 완료

# 모델링

## ① Bidirectional LSTM

# ❶ Bidirectional LSTM - Parameters

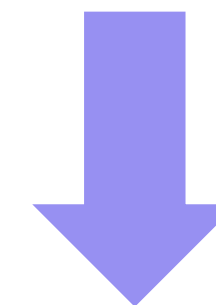
```
from keras.preprocessing.text import Tokenizer
vocab_size = 5000

tokenizer = Tokenizer(num_words = vocab_size)
# Tokenizer 는 데이터에 출현하는 모든 단어의 개수를 세고 빈도 수로 정렬해서
# num_words 에 지정된 만큼만 숫자로 반환하고, 나머지는 0 으로 반환합니다
tokenizer.fit_on_texts(X_train) # Tokenizer 에 데이터 실제로 입력
sequences_train = tokenizer.texts_to_sequences(X_train) # 문장 내 모든 단어를 시퀀스 번호로 변환
sequences_test = tokenizer.texts_to_sequences(X_test) # 문장 내 모든 단어를 시퀀스 번호로 변환

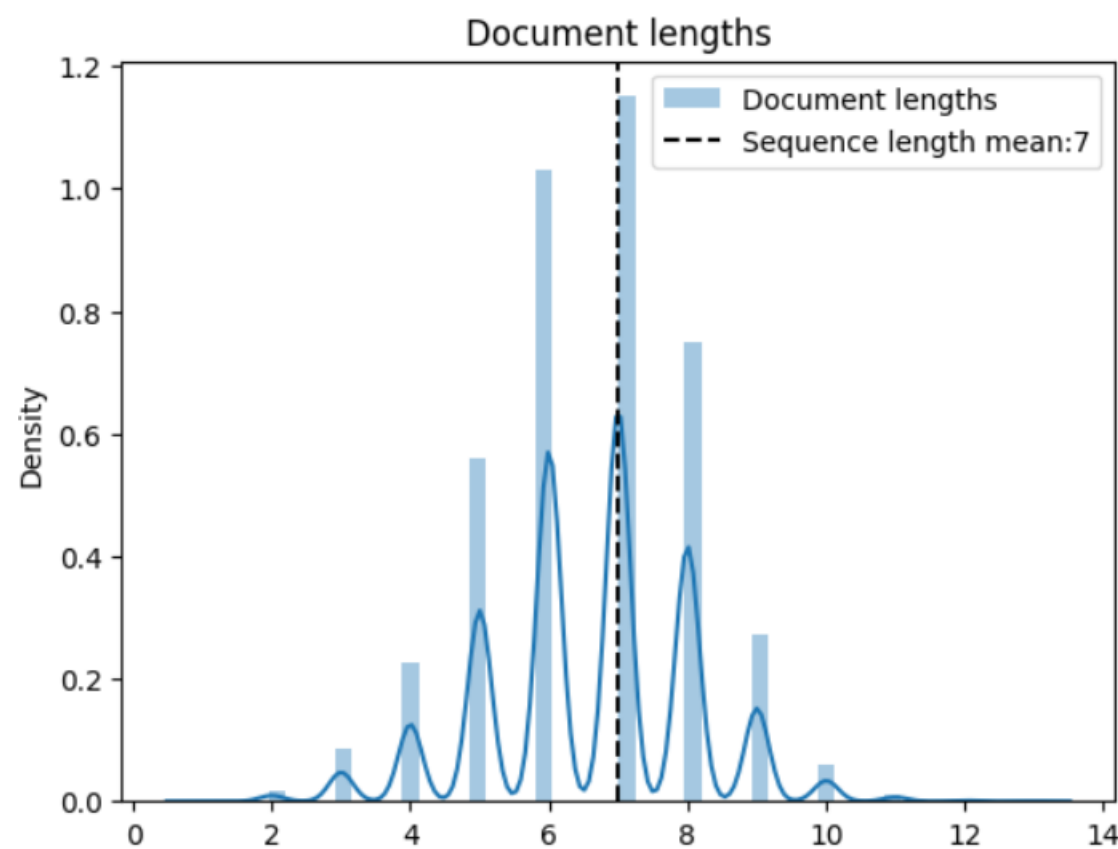
print(len(sequences_train), len(sequences_test))
```

45654 9131

Text Data



Tokenize



그래프를 통해 문장 내  
단어 수 분포 파악

최대 길이 : 14

# ❶ Bidirectional LSTM - Parameters

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences

## 문장의 길이기 제각각이기 때문에 벡터 크기 다 다름
## 그러므로 최대 시퀀스 길이 크기(211) 만큼 넉넉하게 늘리고
## 패딩(padding) 작업을 통해 나머지 빈 공간을 0으로 채움

max_length = 14 # 위에서 그래프 확인 후 정함
padding_type='post'

train_X = pad_sequences(sequences_train, padding='post', maxlen=max_length)
test_X = pad_sequences(sequences_test, padding=padding_type, maxlen=max_length)
```

```
from tensorflow.keras.utils import plot_model, to_categorical
from tensorflow.keras.optimizers import Adam

from keras.utils import np_utils

# 종속변수 데이터 전처리
train_Y = np_utils.to_categorical(Y_train) # Y_train 에 원-핫 인코딩
print(train_Y)
print(train_Y.shape)
```

```
[[0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]
 ...
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]]
(45654, 7)
```

Padding 완료 후  
Y\_train 데이터

# ❶ Bidirectional LSTM - Parameters

```
#파라미터 설정
vocab_size = 5000 # 제일 많이 사용하는 사이즈
embedding_dim = 200
max_length = 14 # 위에서 그래프 확인 후 정함
padding_type='post'
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, Dropout, Bidirectional

model3 = Sequential([Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(LSTM(units = 64, return_sequences = True)),
    tf.keras.layers.Bidirectional(LSTM(units = 64, return_sequences = True)),
    tf.keras.layers.Bidirectional(LSTM(units = 64)),
    Dense(7, activation='softmax') # 결과값이 0~4 이므로 Dense(5)
])

model3.compile(loss= 'categorical_crossentropy', #여러개 정답 중 하나 맞추는 문제이므로 손실 함수는 categorical_crossentropy
    optimizer= 'adam',
    metrics = ['accuracy'])
model3.summary()
```

Sequential 함수를 이용해 모델 설정

# ① Bidirectional LSTM – CV X

Train/Val Split : **O**  
Cross Validation : **X**  
Early Stopping : **X**



# 모델 실행해보기

```
history = model3.fit(train_X, train_Y, epochs=30, batch_size=128, validation_split= 0.2)  
# 양방향 LSTM 레이어에서는 batch size 를 100으로 잡고 30회 학습 해보았다.
```



Epoch 1/30

286/286 [=====] - 37s 84ms/step - loss: 0.9537 - accuracy: 0.6594 - val\_loss: 1.0427 - val\_accuracy: 0.6120

Epoch 2/30

286/286 [=====] - 7s 25ms/step - loss: 0.5104 - accuracy: 0.8317 - val\_loss: 0.9999 - val\_accuracy: 0.6371

Epoch 3/30

286/286 [=====] - 9s 31ms/step - loss: 0.3870 - accuracy: 0.8738 - val\_loss: 1.0684 - val\_accuracy: 0.6203

Epoch 4/30

286/286 [=====] - 5s 19ms/step - loss: 0.3186 - accuracy: 0.8956 - val\_loss: 1.2018 - val\_accuracy: 0.5988

Epoch 5/30

286/286 [=====] - 6s 20ms/step - loss: 0.2674 - accuracy: 0.9113 - val\_loss: 1.3868 - val\_accuracy: 0.5945

Epoch 6/30

286/286 [=====] - 5s 18ms/step - loss: 0.2270 - accuracy: 0.9231 - val\_loss: 1.3635 - val\_accuracy: 0.6131

Epoch 7/30

286/286 [=====] - 4s 16ms/step - loss: 0.1964 - accuracy: 0.9342 - val\_loss: 1.5381 - val\_accuracy: 0.5917

Epoch 8/30

286/286 [=====] - 6s 20ms/step - loss: 0.1665 - accuracy: 0.9437 - val\_loss: 1.7284 - val\_accuracy: 0.5936

Epoch 9/30

286/286 [=====] - 5s 18ms/step - loss: 0.1501 - accuracy: 0.9480 - val\_loss: 1.8077 - val\_accuracy: 0.5897

Epoch 10/30

286/286 [=====] - 5s 16ms/step - loss: 0.1309 - accuracy: 0.9551 - val\_loss: 1.9329 - val\_accuracy: 0.5972

**Relatively low Validation Accuracy**



# 1 Bidirectional LSTM – CV 0

```
#CV

from sklearn.metrics import accuracy_score, log_loss
from sklearn.model_selection import StratifiedKFold
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# 계층 교차 검증
n_fold = 5
seed = 42

cv = StratifiedKFold(n_splits = n_fold, shuffle=True, random_state=seed)

# 테스트데이터의 예측값 담을 곳 생성
test_Y = np.zeros((test_X.shape[0], 7))

# 조기 종료 옵션 추가
es = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=4,
                   verbose=1, mode='min', baseline=None, restore_best_weights=True)

for i, (i_trn, i_val) in enumerate(cv.split(train_X, Y_train), 1):
    print(f'training model for CV #{i}')

    model3.fit(train_X[i_trn],
               to_categorical(Y_train[i_trn]),
               validation_data=(train_X[i_val], to_categorical(Y_train[i_val])),
               epochs=20,
               batch_size=256,
               callbacks=[es])    # 조기 종료 옵션

    test_Y += model3.predict(test_X) / n_fold    # 나온 예측값들을 교차 검증 횟수로 나눈다
```

성능 저하 방지를 위해

학습 조기 종료 옵션 추가

# ① Bidirectional LSTM – CV 0

Train/Val Split : **O**  
Cross Validation : **O**  
Early Stopping : **O**

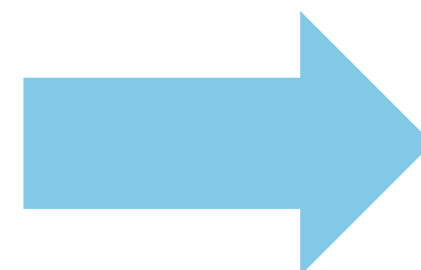
```
training model for CV #1
Epoch 1/20
143/143 [=====] - 19s 135ms/step - loss: 0.3461 - accuracy: 0.9098 - val_loss: 0.3079 - val_accuracy: 0.9138
Epoch 2/20
143/143 [=====] - 9s 58ms/step - loss: 0.1930 - accuracy: 0.9449 - val_loss: 0.3230 - val_accuracy: 0.9088
Epoch 3/20
143/143 [=====] - 5s 38ms/step - loss: 0.1391 - accuracy: 0.9599 - val_loss: 0.3558 - val_accuracy: 0.9015
Epoch 4/20
143/143 [=====] - 4s 27ms/step - loss: 0.1085 - accuracy: 0.9663 - val_loss: 0.3950 - val_accuracy: 0.8975
Epoch 5/20
143/143 [=====] - ETA: 0s - loss: 0.0906 - accuracy: 0.9705Restoring model weights from the end of the best epoch: 1.
143/143 [=====] - 4s 29ms/step - loss: 0.0906 - accuracy: 0.9705 - val_loss: 0.4366 - val_accuracy: 0.8921
Epoch 5: early stopping
286/286 [=====] - 4s 7ms/step
training model for CV #2
Epoch 1/20
143/143 [=====] - 4s 26ms/step - loss: 0.2268 - accuracy: 0.9355 - val_loss: 0.2004 - val_accuracy: 0.9428
Epoch 2/20
143/143 [=====] - 3s 19ms/step - loss: 0.1545 - accuracy: 0.9550 - val_loss: 0.2166 - val_accuracy: 0.9366
Epoch 3/20
143/143 [=====] - 3s 19ms/step - loss: 0.1192 - accuracy: 0.9645 - val_loss: 0.2364 - val_accuracy: 0.9316
Epoch 4/20
143/143 [=====] - 3s 22ms/step - loss: 0.0977 - accuracy: 0.9693 - val_loss: 0.2601 - val_accuracy: 0.9290
Epoch 5/20
142/143 [=====>. ] - ETA: 0s - loss: 0.0832 - accuracy: 0.9727Restoring model weights from the end of the best epoch: 1.
143/143 [=====] - 4s 23ms/step - loss: 0.0833 - accuracy: 0.9726 - val_loss: 0.2882 - val_accuracy: 0.9231
Epoch 5: early stopping
286/286 [=====] - 1s 5ms/step
```

**Relatively high Validation Accuracy**

# ① Bidirectional LSTM – CV 0

```
#각 topic으로 예측할 확률  
test_Y
```

```
array([[9.44389701e-01, 9.01007565e-03, 2.83175334e-02, ...,  
       1.38251950e-03, 1.66954952e-04, 5.99076988e-04],  
       [3.84923920e-04, 2.33899472e-04, 4.70364605e-03, ...,  
       4.80864146e-04, 1.67542895e-04, 1.22500940e-04],  
       [1.71786897e-03, 4.08202619e-03, 9.44459438e-01, ...,  
       1.27410095e-02, 1.29243880e-03, 2.70027912e-02],  
       ...,  
       [2.05261539e-02, 9.36366245e-01, 3.68497777e-02, ...,  
       1.42606560e-03, 1.85704723e-04, 4.29753281e-03],  
       [3.73810984e-03, 1.10083399e-01, 8.34659934e-01, ...,  
       1.99600916e-03, 3.63314377e-03, 4.31525263e-02],  
       [2.97852172e-03, 1.90633907e-02, 9.51867089e-01, ...,  
       1.18569049e-03, 1.12048228e-03, 2.18246346e-02]])
```



	index	topic_idx
0	45654	0
1	45655	3
2	45656	2
3	45657	2
4	45658	3
...	...	...
9126	54780	4
9127	54781	6
9128	54782	1
9129	54783	2
9130	54784	2

9131 rows × 2 columns

# 모델링

② Klue-Roberta/Large

## ② Klue-Roberta/Large – Import Module&Data

```
!pip install transformers
```

```
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import transformers
from transformers import AutoTokenizer, AdamW, RobertaForSequenceClassification

import torch
from torch.nn import functional as F
from torch.utils.data import DataLoader, Dataset

from tqdm.notebook import tqdm
```

```
a=pd.DataFrame()
a["title"]=back_train["title_kor"]
a["index"]=back_train["index"]
a["topic_idx"]=back_train["topic_idx"]
b=pd.DataFrame()
b["title"]=train["title"]
b["index"]=train["index"]
b["topic_idx"]=train["topic_idx"]
```

Transformer 모델 설치

Import Modules

Read Data

## ② Klue-Roberta/Large – Class

```
class NTDataset(Dataset):

    def __init__(self, csv_file):
        self.dataset = csv_file
        self.tokenizer = AutoTokenizer.from_pretrained("klue/roberta-large")

        print(self.dataset.describe())

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

    def __getitem__(self, idx):
        row = self.dataset.iloc[idx, 1:3].values
        text = row[0]
        y = row[1]
        inputs = self.tokenizer(
            text,
            return_tensors='pt',
            truncation=True,
            max_length=14,
            pad_to_max_length=True,
            add_special_tokens=True
        )

        input_ids = inputs['input_ids'][0]
        attention_mask = inputs['attention_mask'][0]

        return input_ids, attention_mask, y
```

Train / Validation

```
class NTDataset_test(Dataset):

    def __init__(self, csv_file):
        self.dataset = csv_file
        self.tokenizer = AutoTokenizer.from_pretrained("klue/roberta-large")

        print(self.dataset.describe())

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

    def __getitem__(self, idx):
        row = self.dataset.iloc[idx, 1:2].values
        text = row[0]
        inputs = self.tokenizer(
            text,
            return_tensors='pt',
            truncation=True,
            max_length=14,
            pad_to_max_length=True,
            add_special_tokens=True
        )

        input_ids = inputs['input_ids'][0]
        attention_mask = inputs['attention_mask'][0]

        return input_ids, attention_mask
```

Test

## ② Klue-Roberta/Large – Model Setting

```
model = RobertaForSequenceClassification.from_pretrained("klue/roberta-large", num_labels=7).to(device)
```

### RobertaForSequenceClassification 함수 이용

```
epochs = 1  
batch_size = 128
```

```
optimizer = AdamW(model.parameters(), lr=1e-5)  
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)  
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=True)  
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

### Optimizer : AdamW 이용

## ② Klue-Roberta/Large – Train

```
# train
losses = []
accuracies = []
total_loss = 0.0
correct = 0
total = 0

for i in range(epochs):

    model.train()

    for input_ids_batch, attention_masks_batch, y_batch in tqdm(train_loader):
        optimizer.zero_grad()
        y_batch = y_batch.to(device)
        y_pred = model(input_ids_batch.to(device), attention_mask=attention_masks_batch.to(device))[0]
        loss = F.cross_entropy(y_pred, y_batch)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        _, predicted = torch.max(y_pred, 1)
        correct += (predicted == y_batch).sum()
        total += len(y_batch)

    losses.append(total_loss)
    accuracies.append(correct.float() / total)
    print("Train Loss:", total_loss / total, "Accuracy:", correct.float() / total)
```

Train



## ② Klue-Roberta/Large – Validation&Test

```
# validation
model.eval()

pred = []
correct = 0
total = 0

for input_ids_batch, attention_masks_batch, y_batch in tqdm(val_loader):
    y_batch = y_batch.to(device)
    y_pred = model(input_ids_batch.to(device), attention_mask=attention_masks_batch.to(device))[0]
    _, predicted = torch.max(y_pred, 1)
    pred.append(predicted)
    correct += (predicted == y_batch).sum()
    total += len(y_batch)

print("val accuracy:", correct.float() / total)
```

Validation

```
# test
model.eval()

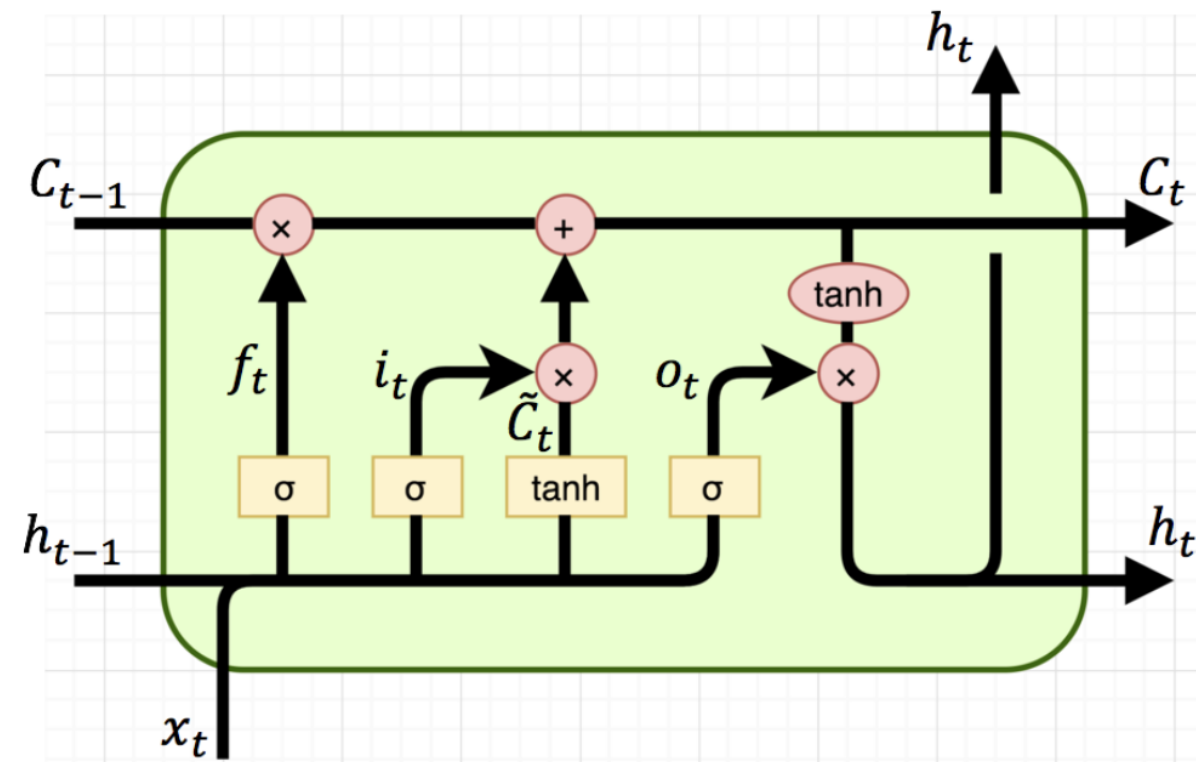
pred = []

for input_ids_batch, attention_masks_batch in tqdm(test_loader):
    y_pred = model(input_ids_batch.to(device), attention_mask=attention_masks_batch.to(device))[0]
    _, predicted = torch.max(y_pred, 1)
    pred.extend(predicted.tolist())
```

Test

# 결과 / 해석

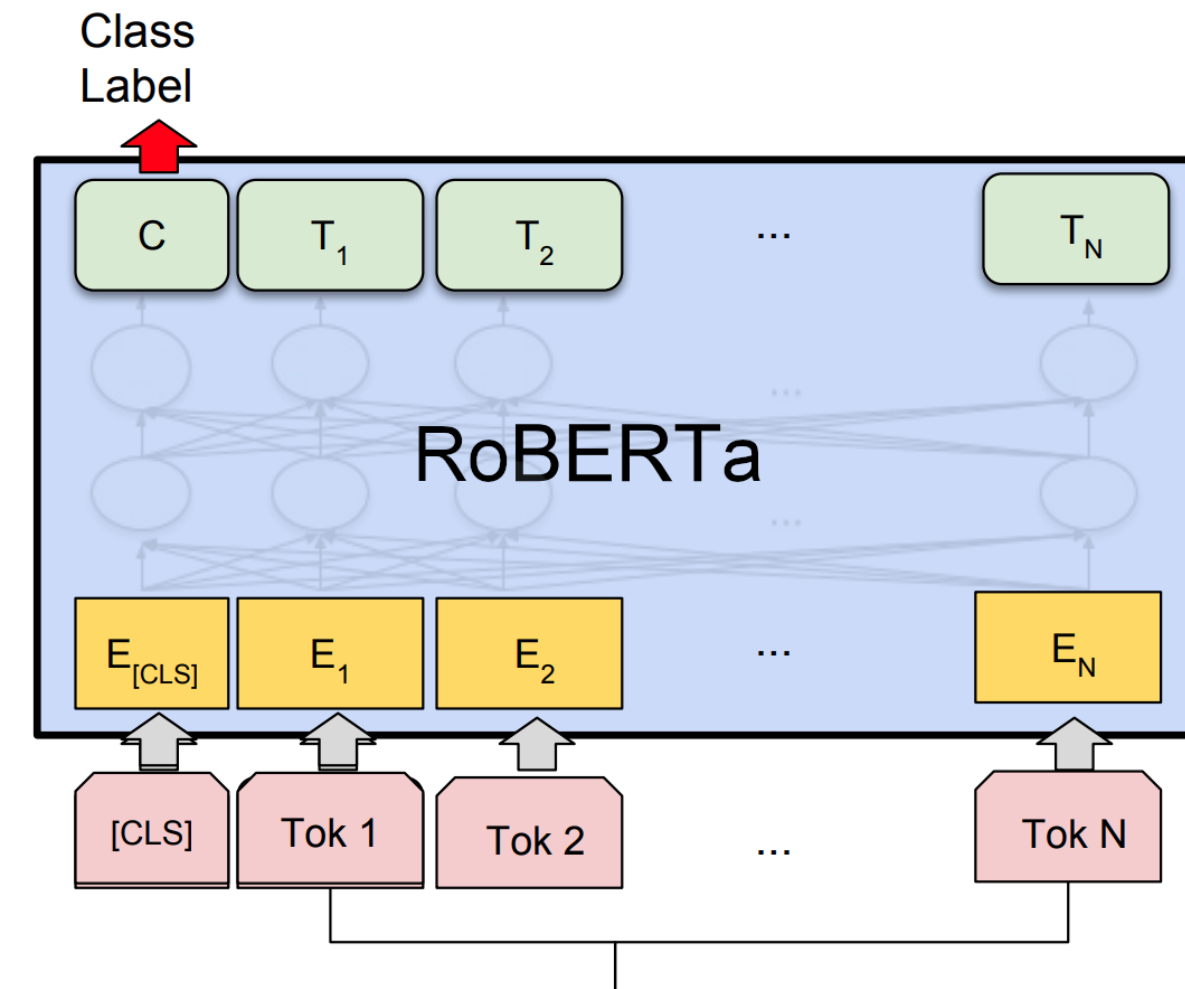
## 04 결과 및 해석



(a) Long Short-Term Memory

LSTM

Public : 0.658269  
Private : 0.651993



KLUE/Roberta-Large

Public : 0.756352  
Private : 0.746801

**감사합니다**