


딥러닝 분반 NLP 소대회: DACON 뉴스 토픽 분류 AI 경진대회

13기 김창현
13기 박무성
13기 배은지
13기 전보민



목차

1. 대회소개
 - 뉴스 토픽 분류 AI 경진대회
 - 데이터 설명
2. 데이터EDA 및 전처리
3. 사용모델
 - BiLSTM
 - KoBERT
4. 결과

대회소개: 뉴스 토픽 분류 AI 경진대회

주제:

한국어 뉴스 헤드라인을 이용하여, 뉴스의 주제를 분류하는 알고리즘 개발

배경:

YNAT(주제 분류를 위한 연합뉴스 헤드라인) 데이터 세트를 활용해 주제 분류 알고리즘을 개발. 국내 최초 오픈 데이터 세트인 KLUE (Korean Language Understanding Evaluation) 데이터 세트를 이용하여 다양한 언어 모델의 성능을 비교해 한국어 자연어처리 분야의 발전에 기여할 것으로 예상

뉴스 토픽 분류 AI 경진대회

월간 데이콘 17 | 자연어 | 분류 | KLUE | Accuracy

🏆 상금 : 500,000 D-point

🕒 2021.06.30 ~ 2021.08.09 17:59 [+ Google Calendar](#)

👤 669명 📅 마감

[참여](#)

대회소개: 데이터 설명

Train Data:

Title: 기사 제목

Topic_idx: 분야별로 분류된 기사의 index

<input type="checkbox"/>	index	title	topic_idx
1	29708	웹툰 나이트에 참가한 작가들	3
2	1189	북한인권법 4일 시행...북한인권재단 다음주 출범	6
3	40319	與 검찰개혁 강공 드라이브...한국당 검찰 검박 홍위...	6
4	11860	이라크중리 민생고 시위 100여명 사망 진상조사 지시	4
5	15621	그래픽 기업경기실사지수 낙폭 메르스 이후 최대	1
6	30193	민주 당권주자 제주시 유세대결...1강2중 관측 속 경...	6
7	6567	카메라 4개 달린 스마트폰 지오니 S10	0
8	34239	내년 첫 5G폰 평균가 80만원 육박...2023년 60만원...	0
9	40441	IBS 천진우 나노의학연구단장팀	0
10	21726	고발장 접수하는 김진태 의원실 관계자	6
11	8858	아시안피스컵 남북 男배구 열전 펼쳐...북한팀 32 승...	5

Topic Index:

IT과학, 경제, 사회, 생활문화, 세계, 스포츠, 정치 등 7개의 분야로 나뉜진 topic index. 다중 클래스 분류 문제 (Multi-class Classification)

<input type="checkbox"/>	topic	topic_idx
1	IT과학	0
2	경제	1
3	사회	2
4	생활문화	3
5	세계	4
6	스포츠	5
7	정치	6

데이터 EDA 및 전처리

훈련 데이터 순서 섞기 (Shuffle)

```
indices = np.arange(data.shape[0])
np.random.seed(5)
np.random.shuffle(indices)
print(indices)
```

```
[15780 1035 28358 ... 20463 18638 35683]
```

```
data = data.iloc[indices].reset_index(drop=True)
```

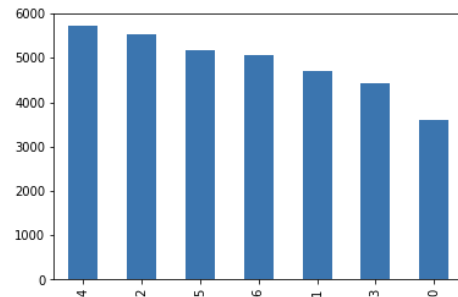
index	title	topic_idx
15780	U20월드컵 이탈리아 우승후보 프랑스 꺾고 8강 진출...	5
1035	카카오프렌즈가 나간다...프렌즈 게임 퍼블리싱 본격화	0
28358	종근당홀딩스 2분기 영업이익 60억원...527% ↑	1
37717	광명전기 181억원 규모 원전 전동기제어반 공급계약	1
38990	일하다 죽지 않게 차별받지 않게...이달 22일 희망버스종합	2
...
5520	주간 화제의 뉴스 국정농단 재판 이국종 등 관심	2
35814	임종석 문재인정부서 사찰행위는 존재하지 않는다종합	6
20463	대한항공-현대캐피탈 FA 대어 정치석·문성민 잡는다	5
18638	정부 北 8·25 합의 되새기고 핵개발·대남도발 중단해야	6
35683	베트남 3대 대도시 주민 복수비자 발급대상 영주권자로 제한	4

Train/test 데이터 분리

```
train_data, test_data = train_test_split(data, test_size = 0.25, random_state = 42)
print('훈련용 리뷰의 개수 :', len(train_data))
print('테스트용 리뷰의 개수 :', len(test_data))
```

훈련용 리뷰의 개수 : 34240
테스트용 리뷰의 개수 : 11414

레이블 분포 확인



불균형 데이터 처리는 필요하지 않다고 판단!

데이터 EDA 및 전처리

데이터 정제하기

- 한글, 영어, 한자, 숫자를 제외한 나머지 제거
- 제거 후 공백이 된 sample이 있다면 제거

```
def elimCharacter(data, mode='Train'):  
    data['title'] = data['title'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣a-zA-Z0-9\u2013\u2018\u2019]", "")  
    if mode=='Train':  
        data.drop_duplicates(subset = ['title'], inplace=True) # 중복 제거  
        data['title'].replace('', np.nan, inplace=True) # 공백은 NaN값으로 변환  
        data = data.dropna(how='any') # 결측치 제거  
        print('전처리 후 샘플의 개수 : ',len(data))  
        print('ㄷ----- Eliminated unmeaningful characters -----ㄷ')  
  
    return data
```

숫자를 제거하지 않은 이유

숫자를 제거한 후 토큰화를 진행할 경우,
토큰화가 잘못 진행되는 경우가 발생

北 美 김정은 제재반발...뉴욕 북미접촉 통로 완전차단보

➡ [北, 美, 김정은, 제재, 반발, 뉴욕, 북, 미접, 촉, 통로, 완전, 차, 단보]

토큰화 진행

- 뉴스 제목 특성상, 사람 이름 또는 지명이 많이 등장
- 그냥 형태소 분석 시 고유명사가 잘못 분석되는 경우 발생!

예시

조명균천해성윤영찬 등 고위급회담 대표단 명단 北에 통지

➡ [조명균천해성윤영, 차, ㄴ, 등, 고위급, 회담, 대표단, 명단, 北, 에, 통지]

페루 아르헨 팬 무서워FIFA에 월드컵예선 경기장 변...

➡ [페루, 아르, 헤, ㄴ, 팬, 무섭, 어, FIFA, 에, 월드컵, 예선, 경기장...]

데이터 EDA 및 전처리



정의

한국어 형태소 분석기인 Kiwi(Korean Intelligent Word Identifier)의 Python 모듈

특징

- 코퍼스로부터 미등록 단어 추출 가능
- 사용자 사전 추가 기능
- 형태소 분석

1. 코퍼스로부터 미등록 단어 추출

```
kiwi.extract_words(texts, min_cnt, max_word_len, min_score)
```

- texts: 분석할 텍스트
- min_cnt: 추출할 단어가 입력 텍스트 내에서 최소 몇 번 이상 등장하는 지 결정
- max_word_len: 추출할 단어의 최대 길이
- min_score: 추출할 단어의 최소 단어 점수

2. 사용자 사전 추가

```
kiwi.add_user_word(word, pos, score)
```

- pos: 등록할 단어의 품사
- score: 등록할 단어의 점수

3. 형태소 분석

```
kiwi.analyze(text, top_n, match_option)
```

데이터 EDA 및 전처리

Kiwi 패키지 활용: 미등록 단어 추출

```
kiwi = Kiwi()
inputs = list(data['title'])
extracted_list = kiwi.extract_words(inputs, min_cnt=2, max_word_len=10, min_score=0.1)

('소상공인', 0.16344572603702545, 30, -2.8995964527130127),
('獨프랑크푸르트', 0.16318117082118988, 2, -0.4862860143184662),
('의원전년', 0.16293174028396606, 16, -2.9272992610931396),
('삼성라이온즈', 0.15813526511192322, 2, -2.229809522628784),
('00만원', 0.15753695368766785, 44, -2.990668535232544),
('6억원', 0.1566649228334427, 58, -2.7227892875671387),
('첨단기술', 0.15578074753284454, 7, -2.8458688259124756),
('아이폰XS', 0.15294866263866425, 10, -2.2486329078674316),
```

Kiwi 패키지 활용: 사전 등록

```
# 고유명사(NNP) 사전등록: '아르헨'을 제외하고는 단어점수를 높게 설정(score=3)
NNP_list = ['아가베즈', '정의용', '서훈', '김성균', '전혜성', '윤건영', '조명균', '윤영찬', '후안 포이스',
            '라오닝', '샌프란시스코', '우즈베크', '삼성라이온즈', '템파베이', 'IBK기업은행', '버라이즌', '아이폰XS', '메타세쿼이아']
for item in NNP_list:
    kiwi.add_user_word(item, 'NNP', 3)
kiwi.add_user_word('아르헨', 'NNP') # score = 0 (default)

# 일반명사(NNG) 사전등록: 전부 단어점수를 높게 설정(score=3)
NNG_list = ['현대미술', '소상공인', '현대무용', '임대주택', '사전예약', '4차산업혁명', '자율주행']
for item in NNG_list:
    kiwi.add_user_word(item, 'NNG', 3)
```

단어 점수 설정 기준

soynlp 패키지의 WordExtractor를 활용해 전체적인 단어 점수 분포 확인 후 적절한 값으로 결정

```
from soynlp.word import WordExtractor

word_extractor = WordExtractor(min_frequency=2,
                               min_cohesion_forward=0.05,
                               min_right_branching_entropy=0.0)
word_extractor.train(list(train_data['title']))
words = word_extractor.extract()
```

```
[4.133307123431565,
3.6152487617397835,
3.5318644485113526,
3.5117464019574167,
3.401187598831576,
3.3751059902115226,
3.160346163063064,
3.0273627004189465,
2.953571434439625,
2.84477024333441764
```


데이터 EDA 및 전처리

Kiwi 패키지 활용: 토큰화 진행 및 품사 기준 불용어 제거

```
pos_list = ['NNG', 'NNP', 'NNB', 'NR', 'VV', 'VA', 'VX', 'VCN', 'MM', 'MAG', 'IC', 'XPN', 'XR', 'SL', 'SH']

X = []
for result in tokenizer.analyze(list(data['title'])):
    result_sliced = result[0][0]
    temp = []
    for token in result_sliced:
        if token[1] in pos_list:
            temp.append(token[0])
    X.append(temp)

data['tokenized'] = X
```

[품사 태그 참고]

NNG 일반 명사	MM 관형사
NNP 고유 명사	MAG 일반 부사
NNB 의존 명사	IC 감탄사
NR 수사	XPN 체언 접두사
VV 동사	XR 어근
VA 형용사	SL 알파벳(A-Z a-z)
VX 보조 용언	SH 한자
VCN 부정 지시사	

[토큰화 예시]

index	title	topic_idx	tokenized
11329	조명균천해성윤영찬 등 고위급회담 대표단 명단 北에 통지	6	[조명균, 천해성, 윤영찬, 등, 고위급, 회담, 대표단, 명단, 北, 통지]

데이터 EDA 및 전처리

추가적인 불용어 지정 및 제거

- 각 토픽 별 빈도수 상위 200 단어 교집합 확인
- 교집합 단어 중 토픽 분류에 도움이 되지 않는다고 판단되는 단어들을 추가적인 불용어로 지정하여 제거

```
top200_words = []  
for words_cnt in words_cnt_list:  
    top200_words.append(set(list(map(lambda x: x[0], words_cnt.most_common(200)))))
```

```
top200_intersection = list(top200_words[0] & top200_words[1] & top200_words[2] &  
                             top200_words[3] & top200_words[4] & top200_words[5] & top200_words[6])  
print('모든 토픽의 빈도수 상위 200 공통 단어: ', top200_intersection)
```

모든 토픽의 빈도수 상위 200 공통 단어: ['월', '중', '개', '첫', '한', '보', '대', '원', '지', '년', '것', '명', '만', '전', '일', '등', '종합', '주', '한국', '위']

```
my_stopwords = ['종합', '년', '월', '일', '명', '대', '등', '만', '한', '것', '개', '지', '보', '위', '전', '첫', '새', '주', '원', '안', '없', '하']  
data['tokenized'] = data['tokenized'].apply(lambda x: [item for item in x if item not in set(my_stopwords)])
```

[최종 토큰화 예시]

title
조명균천해성윤영찬 등 고위급회담 대표단 명단 北에 통지
tokenized
[조명균, 천해성, 윤영찬, 고위급, 회담, 대표단, 명단, 北, 통지]

데이터 EDA 및 전처리

토픽 별 단어 빈도 분포 시각화

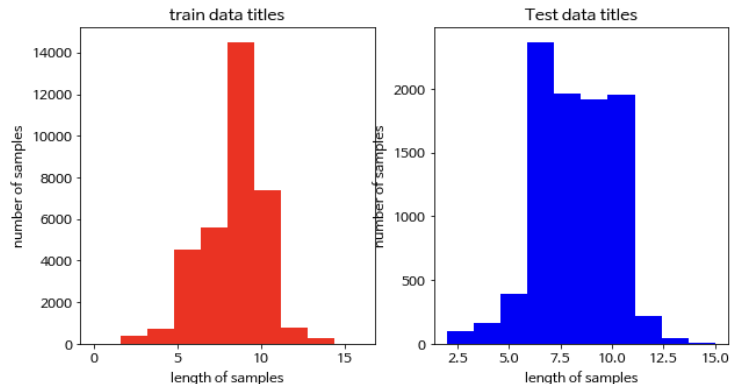


단어 길이 분포 시각화

train data title 평균 길이 : 8.230308519340891

Test data title 평균 길이 : 8.216515168108641

Words in texts



데이터 EDA 및 전처리

정수 인코딩

- 빈도수가 1회인 희귀 단어들의 수를 제외한 단어 개수를 단어 집합의 최대 크기로 제한
- 케라스 토큰라이저로 정수 인코딩 진행

```
vocab_size = total_cnt - rare_cnt + 2  
print('단어 집합의 크기 제한 :', vocab_size)
```

단어 집합의 크기 제한 : 14549

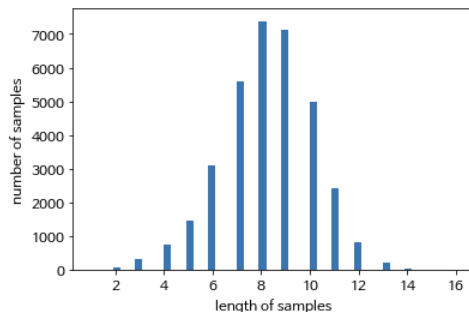
```
tokenizer = Tokenizer(vocab_size, oov_token = 'OOV')  
tokenizer.fit_on_texts(X_train)
```

```
X_train = tokenizer.texts_to_sequences(X_train)  
X_test = tokenizer.texts_to_sequences(X_test)  
X_Test = tokenizer.texts_to_sequences(X_Test)
```

패딩

- 서로 다른 길이의 샘플들의 길이를 동일하게 맞춰주기 위한 패딩 진행
- 훈련 데이터의 길이 분포 확인 후, 최대 길이로 패딩 진행

리뷰의 최대 길이 : 16
리뷰의 평균 길이 : 8.230548981797996



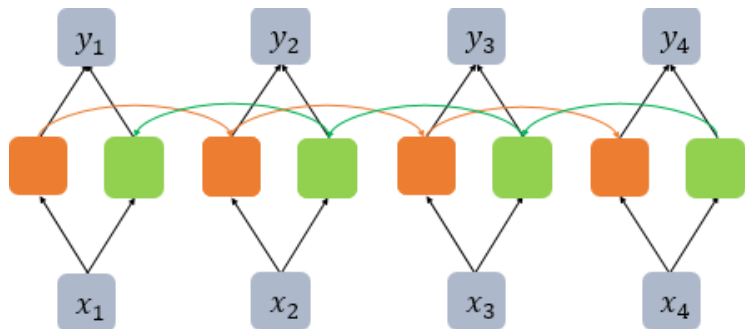
```
X_train = pad_sequences(X_train, maxlen = 16)  
X_test = pad_sequences(X_test, maxlen = 16)  
X_Test = pad_sequences(X_Test, maxlen = 16)
```

사용모델: BiLSTM

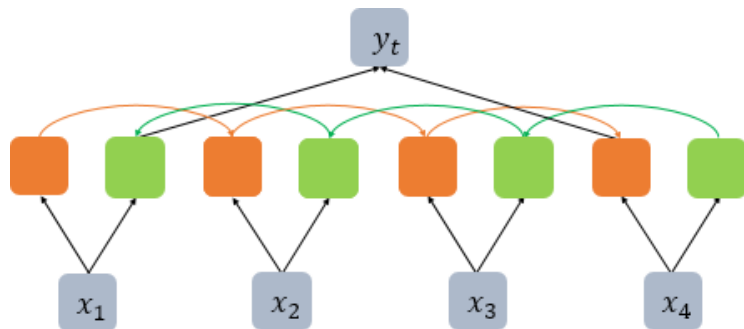
BiLSTM (양방향 LSTM)은 두 개의 독립적인 LSTM을 사용하는 구조.

아래 그림을 참고하면 주황색 LSTM 셀은 순차적으로 입력을 받음. 자연어 처리라고 하면 문장을 왼쪽 단어부터 순차적으로 읽는 셈. 양방향 LSTM은 뒤의 문맥까지 고려하기 위해 문장을 오른쪽에서 반대로 읽는 역방향의 LSTM 셀(초록색)을 함께 사용. 결과적으로 두 가지 정보를 모두 고려하여 출력층에서 모두 사용

다-대-다(Many to Many) 분류



다-대-일(Many to One) 분류



사용모델: BiLSTM

필요한 패키지 불러오기

```
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, GRU, Bidirectional
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model
from keras.layers import Dropout
```

모델 구축

```
model = Sequential()
model.add(Embedding(vocab_size, 200))
model.add(Bidirectional(LSTM(512, return_sequences=True)))
model.add(Dropout(0.3))
model.add(Bidirectional(LSTM(512, return_sequences=True)))
model.add(Dropout(0.3))
model.add(Bidirectional(LSTM(512)))
model.add(Dense(256))
model.add(Dropout(0.3))
model.add(Dense(7, activation='softmax'))
```

조기종료 조건 설정

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=4)
mc = ModelCheckpoint('best_model.h5', monitor='val_acc', mode='max', verbose=1, save_best_only=True)
```

모델 학습

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
history = model.fit(X_train, y_train, epochs=15, callbacks=[es, mc], batch_size=60, validation_split=0.2)
```

```
Epoch 1/15
457/457 [=====] - 17s 19ms/step - loss: 0.7815 - acc: 0.7237 - val_loss: 0.4735 - val_acc: 0.8382

Epoch 00001: val_acc improved from -inf to 0.83815, saving model to best_model.h5
Epoch 2/15
457/457 [=====] - 8s 18ms/step - loss: 0.2972 - acc: 0.9041 - val_loss: 0.5024 - val_acc: 0.8370

Epoch 00002: val_acc did not improve from 0.83815
Epoch 3/15
457/457 [=====] - 8s 17ms/step - loss: 0.1768 - acc: 0.9421 - val_loss: 0.5967 - val_acc: 0.8307

Epoch 00003: val_acc did not improve from 0.83815
Epoch 4/15
457/457 [=====] - 8s 17ms/step - loss: 0.1176 - acc: 0.9624 - val_loss: 0.6983 - val_acc: 0.8215

Epoch 00004: val_acc did not improve from 0.83815
Epoch 5/15
457/457 [=====] - 8s 17ms/step - loss: 0.0848 - acc: 0.9715 - val_loss: 0.8424 - val_acc: 0.8155

Epoch 00005: val_acc did not improve from 0.83815
Epoch 00005: early stopping
```

사용모델: KoBERT

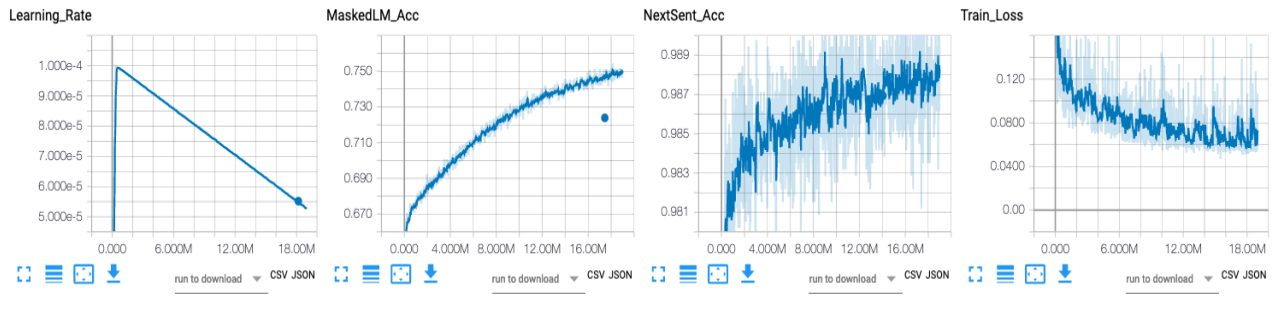
KoBERT는 SKT에서 BERT를 기반으로 만든 모델. 구글 BERT에서 한국어 성능의 한계를 극복하기 위해 만들어짐.

학습에선 한국어 위키에서 500만개의 문장과 5400만개의 단어를 사용함. 네이버 감성분석 데이터를 적용한 결과 BERT와 KoGPT보다 좋은 성능을 보여줌. 구글에서 한글 위키 기반으로 학습시킨 Sentencepiece tokenizer와 8000개의 단어가 들어있는 사전(Vocabulary)사용

Naver Sentiment Analysis

- Dataset : <https://github.com/e9t/nsmc>

Model	Accuracy
BERT base multilingual cased	0.875
KoBERT	0.901
KoGPT2	0.899



사용모델: KoBERT

KoBERT 입력 데이터로 만들기

```
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))
```

하이퍼파라미터 설정

```
# Setting parameters
max_len = 64
batch_size = 32
warmup_ratio = 0.1
num_epochs = 4
max_grad_norm = 1
log_interval = 200
learning_rate = 4e-5
```

KoBERT 학습 모델 구축하기

```
class BERTClassifier(nn.Module):
    def __init__(self,
                 bert,
                 hidden_size = 768,
                 num_classes=7,  ##7개 카테고리로 분류
                 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)
```

학습시키기

```
100% 1427/1427 [09:10<00:00, 2.85it/s]
epoch 4 batch id 1 loss 0.43105894327163696 train acc 0.90625
epoch 4 batch id 201 loss 0.259283954263305664 train acc 0.93578990009950248
epoch 4 batch id 401 loss 0.2646643817424774 train acc 0.9343827930174564
epoch 4 batch id 601 loss 0.2471175491809945 train acc 0.9364600665557404
epoch 4 batch id 801 loss 0.11763133853679335 train acc 0.9386704119850188
epoch 4 batch id 1001 loss 0.022024845704436302 train acc 0.9473963636463537
epoch 4 batch id 1201 loss 0.088623948395625223 train acc 0.9526436303080766
epoch 4 batch id 1401 loss 0.011783912777900696 train acc 0.9531807637401856
epoch 4 train acc 0.9538586194814296
```


결과

모델: BiLSTM

Public 점수
0.82256297921

Private 점수
0.7908453789

모델: KoBERT

Public 점수
0.8617743702

Private 점수
0.829391152

E.O.D