

# 텍스트 마이닝과 데이터 마이닝

# Part 06. 자연어 처리와 실습

정 정 민

## Chapter 17. 딥러닝 모델을 활용한 문장 분류 실습

---

1. Hugging Face 사용
2. Yelp 데이터 감정 분석 with 딥러닝 모델

# Hugging Face 사용

# HuggingFace 모델 선택

- HuggingFace에는 다양한 연구 결과물들을 사용할 수 있음
- Models 페이지([링크](#))에서 알맞은 연구 성과를 선택
  - 문장 분류 (Text Classification) 선택!
- SNS 데이터로 학습한 감정 분석 모델을 사용 ([링크](#))
  - 2018.01 ~ 2021.12 까지 1.24억 개 tweet을 학습
  - 출력으로 긍정 / 부정 / 중립 중 하나를 출력
- 이 모델은
  - 문장 분류에 기본이 되는 BERT라는 딥러닝 모델을 개선한
  - RoBERTa 라는 모델을 기본으로 하고
  - Tweet 데이터로 추가 학습 한 모델
- 이번 실습에서는 학습을 따로 진행하지 않음

[cs.CL] 1 Apr 2022

## TimeLMs: Diachronic Language Models from Twitter

Daniel Loureiro<sup>\*✉</sup>, Francesco Barbieri<sup>\*✉</sup>,  
Leonardo Neves<sup>✉</sup>, Luis Espinosa Anke<sup>✉</sup>, Jose Camacho-Collados<sup>✉</sup>  
<sup>✉</sup> LIAAD - INESC TEC, University of Porto, Portugal  
<sup>✉</sup> Snap Inc., Santa Monica, California, USA  
<sup>✉</sup> Cardiff NLP, School of Computer Science and Informatics, Cardiff University, UK  
<sup>✉</sup> daniel.b.loureiro@inesctec.pt, <sup>✉</sup> {fbarbieri, lneves}@snap.com,  
<sup>✉</sup> {espinosa-ankel, camachocolladosj}@cardiff.ac.uk

### Abstract

Despite its importance, the *time* variable has been largely neglected in the NLP and language model literature. In this paper, we present TimeLMs, a set of language models specialized on diachronic Twitter data. We show that a continual learning strategy contributes to enhancing Twitter-based language models' capacity to deal with future and out-of-distribution tweets, while making them competitive with standardized and more monolithic benchmarks. We also perform a number of qualitative analyses showing how they cope with trends and peaks in activity involving specific named entities or concept drift. TimeLMs is available at <https://github.com/cardiffnlp/timelms>.

a few examples. The lack of *diachronic specialization* is especially concerning in contexts such as social media, where topics of discussion change often and rapidly (Del Tredici et al., 2019).

In this paper, we address this issue by sharing with the community a series of *time-specific LMs specialized to Twitter data* (TimeLMs). Our initiative goes beyond the initial release, analysis and experimental results reported in this paper, as models will periodically continue to be trained, improved and released.

### 2 Related Work

There exists a significant body of work on dealing with the time variable in NLP. For instance, by specializing language representations derived

# Tokenize

- 딥러닝 모델의 **전처리**는 **모델마다 상이함**
- HuggingFace는 특정 모델에 맞는 전처리 코드를 제공
- 목표하는 **모델 HF 페이지 이름을 제공**하면
- 그 모델의 **전처리 Tokenizer**를 제공
  - **Auto** : HF 페이지 이름만으로 목표하는 것들을 자동으로 가져올 수 있음
- Stop words, Stemming 사용하지 않음
- Tokenizer의 결과로
  - Token의 index와
  - 기타 모델 입력 데이터가 생성

```
from transformers import AutoTokenizer

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment-latest"
tokenizer = AutoTokenizer.from_pretrained(MODEL)

tokenizer(text)
```

# Token의 Index

---

- Token과 Embedding vector 사이를 이어주는 mapping value
- Token index를 사용하면 몇 장점이 있음
  - **메모리적 이득**
    - Token은 글의 형태이므로 메모리를 많이 차지함
    - 특정 Token이 몇 번째(Index) Token 인지만 약속해두면 (이 모델의 예)
    - 단순 정수인 Index 값만 넘겨주면 메모리 절약이 좋음
  - Embedding Vector를 모델에 귀속시켜 **관리 용이성** ↑
    - 딥러닝 모델은 Word2Vec이나 GloVe와 같이 독립적인 Embedding을 사용하지 않음
    - 모델의 특성에 맞는 Embedding 모델을 각자 설계
    - 따라서 모델 구조의 앞단에 Embedding을 삽입
- Index는 Embedding Vector가 쌓여 있는 순서를 의미함!

# Model 호출

- Auto 모델 로더를 바탕으로 HF 페이지 이름으로 간단하게 학습된 모델을 호출 가능

```
from transformers import AutoModelForSequenceClassification

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment-latest"
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

- 불러온 model을 print 하면 구조 파악 가능
- Embeddings 모듈만 추론하여 embedding 값 도출 가능

```
print(model)

RobertaForSequenceClassification(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(50265, 768, padding_idx=1)
      (position_embeddings): Embedding(514, 768, padding_idx=1)
      (token_type_embeddings): Embedding(1, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (classifier): RobertaClassificationHead(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (out_proj): Linear(in_features=768, out_features=3, bias=True)
    )
  )
)
```



# Model 사용

- 모델을 사용하기 위해서는 적절한 입력 인자를 넘겨주어야 함
- 이 입력 인자는 모델 연구자의 활용 사례 혹은 모델 설명을 통해 확인
  - 활용 사례 : [링크](#)
    - 대부분 Models 페이지에 존재
  - 모델 설명 : [링크](#)
    - Base model의 class를 확인 후
    - Documentation에서 확인

## Parameters

- **input\_ids** (torch.LongTensor of shape (batch\_size, sequence\_length)) — Indices of input sequence tokens in the vocabulary.

Indices can be obtained using [AutoTokenizer](#). See [PreTrainedTokenizer.encode\(\)](#) and [PreTrainedTokenizer.call\(\)](#) for details.

[What are input IDs?](#)

```
output = model(input_ids = tokenized['input_ids'])

# 혹은

output = model(**tokenized)
```

## 출력 확인 그리고 Config

- 모델의 출력은 각 감정 상태를 예측하는 점수(score)의 형태
- 점수 값은 해석이 어려움
- 이를 확률의 형태로 변환 (softmax 함수 활용)
- 결과적으로
  - 0번째 감정은 0.9%
  - 1번째 감정은 1.5%
  - 2번째 감정은 97.6%
- 각 위치의 감정을 알기 위해서는 **모델이 설정한 감정의 index**를 알아야 함
- 이런 정보는 **config**라는 변수에 저장 & Auto 로더로 호출
- 즉, Positive가 97.6 % 구나~!!

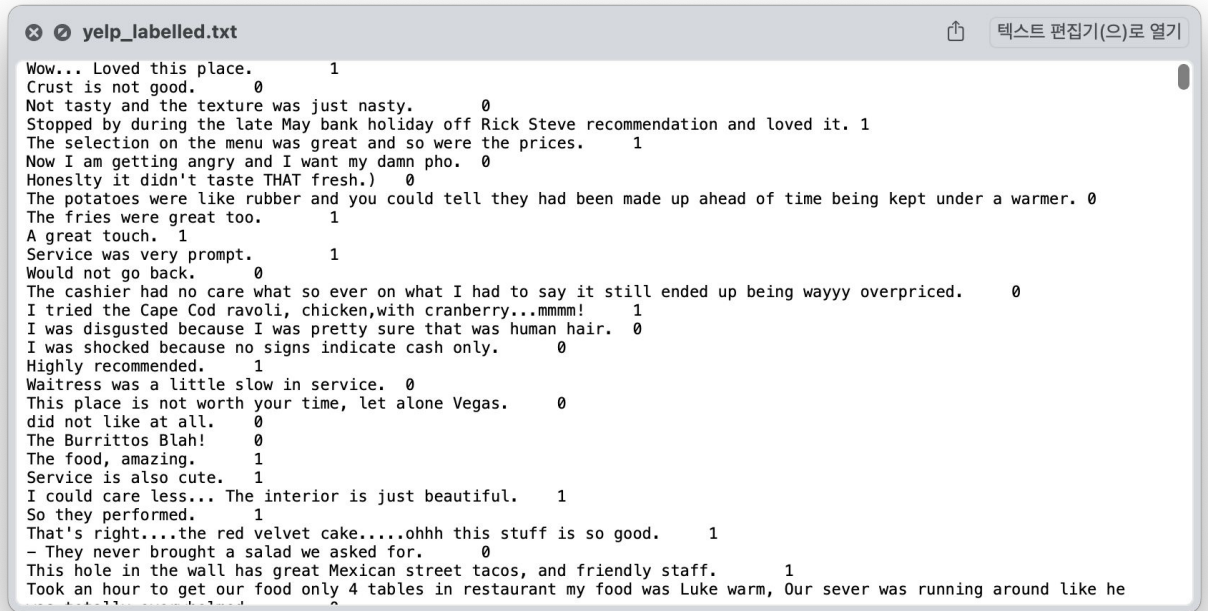
```
scores = output[0][0].detach().numpy()  
scores = softmax(scores)  
print(scores)  
  
# [0.00871163 0.01483237 0.97645605]
```

```
from transformers import AutoConfig  
config = AutoConfig.from_pretrained(MODEL)  
  
print(config)  
  
# ...  
# "id2label": {  
#   "0": "negative",  
#   "1": "neutral",  
#   "2": "positive"  
# }  
# ...
```

# Yelp 데이터 감정 분석 with 딥러닝 모델

# 사용 데이터

- Part 4의 Yelp 데이터를 사용
  - Kaggle 다운로드 : [링크](#)
- 감성 분석도 문장 분류의 한 종류
- 학습된 딥러닝 모델을 활용
- 과거 머신러닝 학습 결과와 비교
  - GloVe Embedding &
  - Logistic Regression



The screenshot shows a text editor window titled 'yelp\_labelled.txt'. The text inside consists of multiple lines of Yelp reviews, each followed by a sentiment label (0 for negative, 1 for positive). The reviews cover various topics like food quality, service, and atmosphere. The labels are placed at the end of each line, sometimes on the same line and sometimes on a new line.

```
Wow... Loved this place. 1
Crust is not good. 0
Not tasty and the texture was just nasty. 0
Stopped by during the late May bank holiday off Rick Steve recommendation and loved it. 1
The selection on the menu was great and so were the prices. 1
Now I am getting angry and I want my damn pho. 0
Honeslty it didn't taste THAT fresh.) 0
The potatoes were like rubber and you could tell they had been made up ahead of time being kept under a warmer. 0
The fries were great too. 1
A great touch. 1
Service was very prompt. 1
Would not go back. 0
The cashier had no care what so ever on what I had to say it still ended up being wayyy overpriced. 0
I tried the Cape Cod ravioli, chicken,with cranberry...mmm! 1
I was disgusted because I was pretty sure that was human hair. 0
I was shocked because no signs indicate cash only. 0
Highly recommended. 1
Waitress was a little slow in service. 0
This place is not worth your time, let alone Vegas. 0
did not like at all. 0
The Burritos Blah! 0
The food, amazing. 1
Service is also cute. 1
I could care less... The interior is just beautiful. 1
So they performed. 1
That's right....the red velvet cake.....ohhh this stuff is so good. 1
- They never brought a salad we asked for. 0
This hole in the wall has great Mexican street tacos, and friendly staff. 1
Took an hour to get our food only 4 tables in restaurant my food was Luke warm, Our sever was running around like he
```

## 평가 데이터로 추론 진행

- 머신 러닝 학습 결과를 확인하기 위해 사용한 평가 데이터를 똑같이 준비
  - `random_state=42`를 통해 동일한 결과 도출
- 정답은 0(neg) 혹은 1(pos)이지만 딥러닝 모델의 결과는 0(neg), 1(neutral), 2(pos)
  - 딥러닝 모델이 2(pos)가 나온 경우만 1로 치환하고
  - 1(neutral)이 나왔다면 강제로 틀리도록 표시
  - 딥러닝 모델에게 더욱 어려운 조건

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42)

def convertResults(index):
    if index == 0 : return index
    elif index == 1 : return -1
    elif index == 2 : return 1

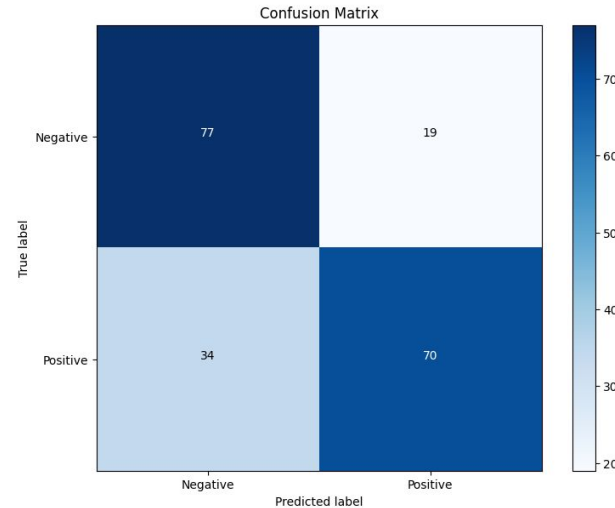
predictions = []

for test in X_test :
    res, idx = sentiment_analysis(test, tokenizer, model, config)
    idx = convertResults(idx)
    predictions.append(idx)

predictions = np.array(predictions)
```

## 결과 비교 분석

- 과거 머신러닝 모델의 confusion matrix와 수치 결과값과
- 딥러닝 모델의 confusion matrix와 수치 결과값 비교
- Neutral 클래스의 존재로 잘못된 예측이 있지만
- 전체적인 성능은 높음
- 특히, Pos와 Neg 사이의 혼동 거의 x

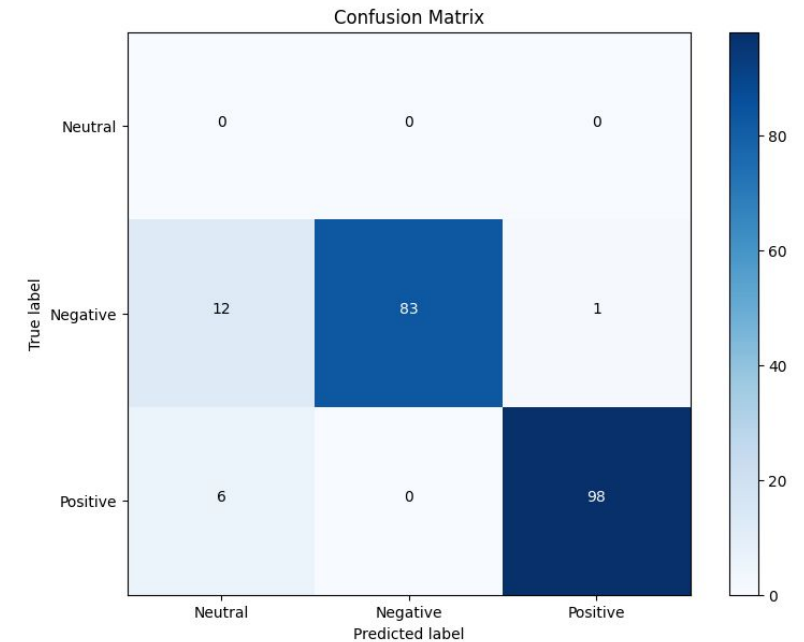


Accuracy: 0.73

Precision: 0.79

Recall: 0.67

F1-Score: 0.73



Accuracy: 0.91

Precision: 0.91

Recall: 0.91

F1-Score: 0.91

# 가상 데이터 테스트

- 10개의 가상 테스트 데이터에 대해 정답

```
for idx, exp in enumerate(examples) :  
    res, index = sentiment_analysis(exp, tokenizer, model, config)  
    origin_sent = '긍정적' if idx % 2 == 0 else '부정적'  
    if index == 0 : pred_sent = '부정적'  
    elif index == 1 : pred_sent = '중립'  
    else : pred_sent = '긍정적'  
  
    print(f"문장: {exp} \n원래 감정 : {origin_sent} / 예측 : {pred_sent}", end='\n\n')
```

**E.O.D**