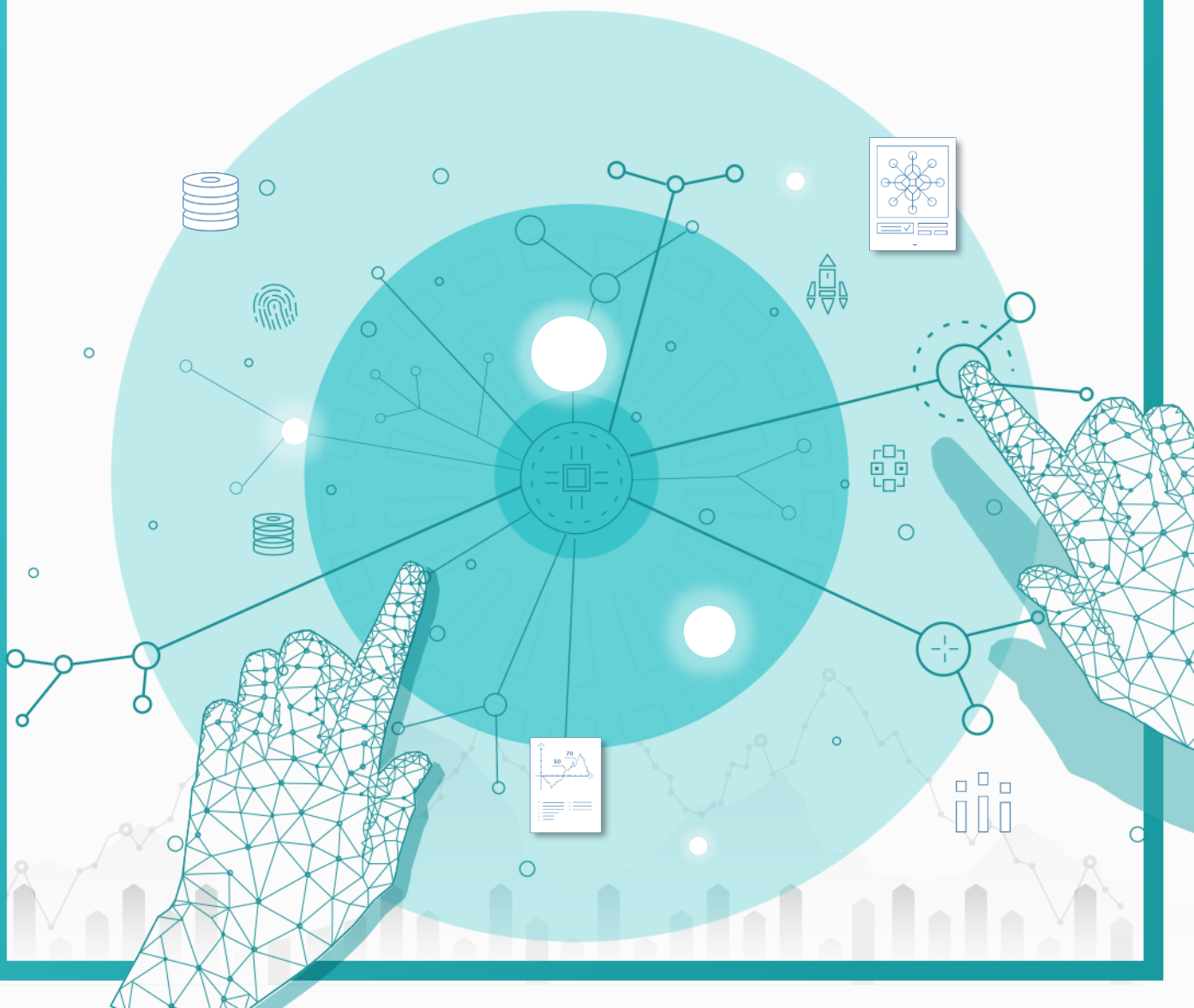




한국기술교육대학교
온라인평생교육원

파이썬을 활용한 인공지능 자연어 처리(실습)

자연어 처리를 위한
BERT 모델



자연어 처리를 위한 BERT 모델

학습 목표

1. BERT 모델을 Fine-tuning할 수 있다.
2. BERT 모델을 감성분석에 적용할 수 있다.

학습 내용

1. BERT 모델 Fine-tuning
2. BERT 모델을 감성분석

1. BERT 모델 Fine-tuning

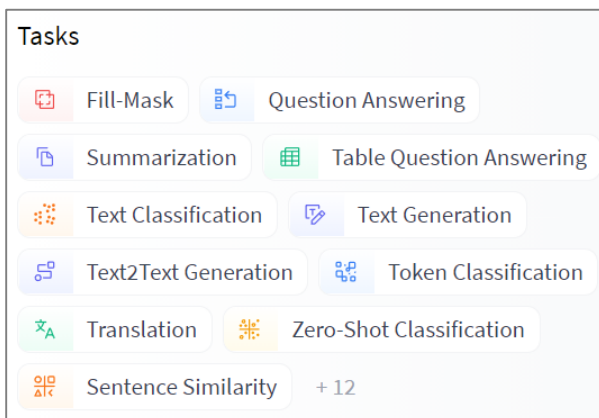
1) Huggingface

(1) Huggingface란?

Pre-trained 모델을 공유하기 위한 모델 공유 플랫폼(Model Hub)

→ 모델 다운로드 및 Fine-tuning을 이용하여 활용 가능

- 언어 모델 Fine-tuning 목적에 따라 모델 공유



※출처 : <https://huggingface.co/models>

- 딥러닝 프레임워크 라이브러리, 데이터 세트, 언어별 모델 공유



※출처 : <https://huggingface.co/models>

1. BERT 모델 Fine-tuning

1) Huggingface

(1) Huggingface란?

- 모델 검색 및 모델명을 이용한 다운로드

Models 18,256

Search Models

Sort: Most Downloads

bert-base-uncased
Fill-Mask • Updated May 19 • 16.2M • ♥ 43

roberta-large
Fill-Mask • Updated May 21 • 5.22M • ♥ 17

distilbert-base-uncased
Fill-Mask • Updated Aug 30 • 5.08M • ♥ 22

gpt2
Text Generation • Updated May 20 • 3.89M • ♥ 17

sentence-transformers/paraphrase-xlm-r-multilin...
Sentence Similarity • Updated Aug 5 • 3.54M • ♥ 18

hfl/chinese-roberta-wwm-ext
Fill-Mask • Updated May 20 • 2.84M • ♥ 8

distilbert-base-uncased-finetuned-sst-2-english
Text Classification • Updated Feb 9 • 2.81M • ♥ 14

xlm-roberta-base
Fill-Mask • Updated Sep 16 • 2.66M • ♥ 9

valhalla/distilbart-mnli-12-3
Zero-Shot Classification • Updated Jun 14 • 2.62M • ♥ 5

deepset/roberta-base-squad2
Question Answering • Updated Aug 2 • 2.32M • ♥ 14

roberta-base
Fill-Mask • Updated Jul 6 • 2.29M • ♥ 6

t5-base
Translation • Updated Jun 23 • 1.7M • ♥ 12

※출처 : <https://huggingface.co/models>

1. BERT 모델 Fine-tuning

2) Huggingface Transformer

(1) Huggingface Transformer란?

- ① 자연어 처리를 위한 범용 아키텍처 제공
- ② BERT 계열, GPT 계열 등의 다양한 언어 모델 아키텍처 제공
- ③ PyTorch, Tensorflow 등 다양한 프레임워크 지원 가능
- ④ Python 패키지로 제공되어 설치 및 사용 용이

1. BERT 모델 Fine-tuning

3) Huggingface와 Transformer

(1) Huggingface Transformer 설치

```
!pip install transformers
```

```
Collecting transformers
```

```
  Downloading transformers-4.11.3-py3-none-any.whl (2.9 MB)
```

```
  |
```

(2) Huggingface Transformer 모델 다운로드

- Down Stream TASK에 따른 클래스 import

```
from transformers import BertForSequenceClassification
```

- [참고] Down Stream TASK란?
 - 모델 Fine-tuning을 통해 수행하고자 하는 작업
(예) Text Classification, Question Answering,
Summarization 등

1. BERT 모델 Fine-tuning

3) Huggingface와 Transformer

(2) Huggingface Transformer 모델 다운로드

- 다운로드할 Pre-train 모델명 지정

```
bert_model_name = 'bert-base-uncased'
```

- from_pretrained()함수를 이용한 모델 다운로드

```
model = BertForSequenceClassification.from_pretrained(bert_model_name)
```

Downloading: 100%  570/570 [00:00<00:00,

Downloading: 100%  420M/420M [00:11<00

1. BERT 모델 Fine-tuning

3) Huggingface와 Transformer

(3) 다운로드 모델 구조

- 다운로드 모델 구조 확인

```
for name, param in model.bert.named_parameters():  
    print(name)
```

- BERT Embedding Layer

```
embeddings.word_embeddings.weight  
embeddings.position_embeddings.weight  
embeddings.token_type_embeddings.weight  
embeddings.LayerNorm.weight  
embeddings.LayerNorm.bias
```

- BERT Encoder X 12

```
encoder.layer.0.attention.self.query.weight  
encoder.layer.0.attention.self.query.bias  
encoder.layer.0.attention.self.key.weight  
encoder.layer.0.attention.self.key.bias  
encoder.layer.0.attention.self.value.weight  
encoder.layer.0.attention.self.value.bias  
encoder.layer.0.attention.output.dense.weight  
encoder.layer.0.attention.output.dense.bias  
encoder.layer.0.attention.output.LayerNorm.weight  
encoder.layer.0.attention.output.LayerNorm.bias  
encoder.layer.0.intermediate.dense.weight  
encoder.layer.0.intermediate.dense.bias  
encoder.layer.0.output.dense.weight  
encoder.layer.0.output.dense.bias
```


1. BERT 모델 Fine-tuning

3) Huggingface와 Transformer

(3) 다운로드 모델 구조

- Classification Layer

```
pooler.dense.weight  
pooler.dense.bias
```

- 다운로드 모델 파라미터

```
for name, param in model.bert.named_parameters():  
    print(name, param)
```

```
encoder.layer.23.output.LayerNorm.weight Parameter containing:  
tensor([0.7078, 0.7500, 0.7275, ..., 0.7114, 0.7231, 0.6928],  
       requires_grad=True)  
encoder.layer.23.output.LayerNorm.bias Parameter containing:  
tensor([ 0.0547, 0.0670, 0.0421, ..., -0.0172, -0.0762, -0.0607],  
       requires_grad=True)  
pooler.dense.weight Parameter containing:  
tensor([[ -0.0391,  0.0444, -0.0249, ...,  0.0010,  0.0244,  0.0079],  
        [ -0.0225,  0.0373, -0.0350, ..., -0.0122,  0.0424,  0.0095],  
        [ -0.0328, -0.0122, -0.0519, ..., -0.0761,  0.0065, -0.0129],  
        ...,  
        [ -0.0033,  0.0137,  0.0128, ...,  0.0554, -0.0628, -0.0112],  
        [ -0.0333,  0.0165, -0.0234, ..., -0.0384, -0.0343, -0.0629],  
        [  0.0137,  0.0049, -0.0149, ...,  0.0111,  0.0091,  0.0217]],  
       requires_grad=True)  
pooler.dense.bias Parameter containing:  
tensor([ -0.0365, -0.0285,  0.0037, ...,  0.0227,  0.0211, -0.0185],  
       requires_grad=True)
```

1. BERT 모델 Fine-tuning

3) Huggingface와 Transformer

(4) 모델 Fine-tuning 수행(Trainable/Non-trainable 설정)

- `requires_grad = True`
 - 해당 Layer의 파라미터들을 학습을 통해 변경
- `requires_grad = False`
 - 해당 Layer의 파라미터를 frozen, 학습을 통해 변경하지 않음

```
if tl_strategy == 1:
    for name, param in model.bert.named_parameters():
        param.requires_grad = False

elif tl_strategy == 2:
    for name, param in model.bert.named_parameters():
        if not name.startswith('pooler'):
            param.requires_grad = False

elif tl_strategy == 3:
    for name, param in model.bert.named_parameters():
        if ( not name.startswith('pooler') ) and "layer.23" not in name :
            param.requires_grad = False
```

2. BERT 모델 감성분석

1) 학습 및 평가 데이터 세트

(1) 데이터 세트 준비

- IMDB 데이터 세트 업로드 및 Pandas를 이용한 데이터 로드

```
import pandas as pd

dataset = pd.read_csv("/content/mnt/MyDrive/IMDB Dataset.csv",

dataset.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

2. BERT 모델 감성분석

1) 학습 및 평가 데이터 세트

(1) 데이터 세트 준비

- Scikit-learn을 이용한 Train(80%), Test(20%) 데이터 세트 분리
 - stratify 파라미터를 통한 층화 샘플링(Stratified Sampling)

```
from sklearn.model_selection import train_test_split

train_idx, test_idx, _, _ = train_test_split(dataset.index,
                                             test_size=0.2,
train_set = dataset.iloc[train_idx]
test_set = dataset.iloc[test_idx]
train_set.reset_index(drop=True, inplace=True)
```

- Train 데이터 세트를 이용하여 Train(70%), Validation(30%) 데이터 세트 분리
 - stratify 파라미터를 통한 층화 샘플링(Stratified Sampling)

```
train_idx, valid_idx, _, _ = train_test_split(train_set.index,
                                             test_size=0.3, s
valid_set = train_set.iloc[valid_idx]
train_set = train_set.iloc[train_idx]

train_set.shape, valid_set.shape, test_set.shape

((84000, 3), (36000, 3), (30000, 3))
```

2. BERT 모델 감성분석

1) 학습 및 평가 데이터 세트

(2) 데이터 세트 클래스 정의

`torch.utils.data.Dataset` 상속

`__init__()`, `__len__()`, `__getitem__()` 함수 재정의

`__init__()` Text/Label 설정, Tokenizer, 최대 Token 수 지정

`__len__()` 전체 데이터 세트의 개수 리턴

`__getitem__()` 특정 index에 해당하는 입력 Text ID, attention mask, label 리턴

2. BERT 모델 감성분석

2) Pre-trained 모델 다운로드 및 Fine-tuning 설정

(1) Pre-trained Tokenizer 다운로드

- Huggingface로부터 Pre-trained Tokenizer 다운로드

```
from transformers import BertTokenizerFast

bert_token_model = 'bert-base-uncased'
bert_model_name = bert_token_model #'bert-large-uncased'

tokenizer = BertTokenizerFast.from_pretrained(bert_token_model)
```

- 주의!

- Pre-trained 모델과 호환될 수 있는 Tokenizer 다운로드

- Train 데이터 세트와 Validation 데이터 세트를 이용하여 객체 생성

```
train_set_dataset = BertDataset(
    reviews    = train_set.review.tolist(),
    sentiments  = train_set.sentiment.tolist(),
    tokenizer   = tokenizer,
)

valid_set_dataset = BertDataset(
    reviews    = valid_set.review.tolist(),
    sentiments  = valid_set.sentiment.tolist(),
    tokenizer   = tokenizer,
)
```

2. BERT 모델 감성분석


2) Pre-trained 모델 다운로드 및 Fine-tuning 설정

(2) Pre-trained 모델 다운로드

- Huggingface로부터 'Bert-base-uncased' Pre-trained 모델 다운로드

```
from transformers import BertForSequenceClassification

model = BertForSequenceClassification.from_pretrained(bert_model_name)
```

Downloading: 100%  454M/454M [00:11<00:00]

(3) Fine-tuning 설정

- `tl_strategy`를 설정하여 `param.requires_grad = False` 범위 설정

```
tl_strategy = 3

if tl_strategy == 1:
    for name, param in model.bert.named_parameters():
        print(name)
        param.requires_grad = False

elif tl_strategy == 2:
    for name, param in model.bert.named_parameters():
        if not name.startswith('pooler'):
            param.requires_grad = False

elif tl_strategy == 3:
    for name, param in model.bert.named_parameters():
        if ( not name.startswith('pooler') ) and "layer.23" not in name :
            param.requires_grad = False
```


2. BERT 모델 감성분석

3) 모델 학습 및 평가

(1) Training을 위한 하이퍼 파라미터 설정

- 모델 Output 디렉토리, Epoch 수, Batch Size 등
하이퍼 파라미터 설정

```
training_args = TrainingArguments(  
    output_dir                = model_dir,  
    num_train_epochs          = 1,  
    per_device_train_batch_size = 128,  
    per_device_eval_batch_size = 64,  
    warmup_steps              = 500,  
    weight_decay               = 0.01,  
    save_strategy              = "epoch",  
    evaluation_strategy        = "steps"  
)
```

(2) Evaluation을 위한 Test 데이터 세트 준비 및 파라미터 설정

- Test 데이터 세트를 이용하여 객체 생성

```
test_set_dataset = BertDataset(  
    reviews      = test_set.review.tolist(),  
    sentiments    = test_set.sentiment.tolist(),  
    tokenizer     = tokenizer,  
)
```


2. BERT 모델 감성분석

3) 모델 학습 및 평가

(2) Evaluation을 위한 Test 데이터 세트 준비 및 파라미터 설정

- Predict를 위한 do_predict 등 파라미터 설정

```
training_args = TrainingArguments(  
    output_dir = "./model",  
    do_predict = True  
)
```

(3) Evaluation을 위한 Predict 수행

- trainer.predict() 함수를 이용하여 Predict 수행

```
trainer = Trainer(  
    model = model,  
    args = training_args,  
    compute_metrics = compute_metrics,  
)  
  
trainer.predict(test_set_dataset)
```