

GEN AI 인텐시브 과정

강사장철원

Section 0

코스소개

DAY1

LLM
Basic
Concept

DAY2

Transformers
paper
review

DAY3

Transformers
LangChain
LangGraph

DAY4

DAY5

DAY6

LLM
service
develop

DAY7

Final Project

DAY8

□ 트랜스포머 라이브러리 소개

□ 긍부정 분석

GEN AI 인텐시브 과정

Section 1. 트랜스포머

Section 1-1. 트랜스포머 라이브러리 소개

Transformers 라이브러리

- transformers 라이브러리는 Hugging Face에서 만든 LLM 활용시 필수 라이브러리
- 사전 학습된 모델(pretrained models)을 쉽게 불러와서 텍스트 생성, 분류, 번역, 요약 등 다양한 작업 가능

복잡한 NLP 모델을 쉽게 가져와서 바로 사용하자!

Section

트랜스포머라이브러리 소개

Hugging Face

The screenshot shows the homepage of the Hugging Face website (huggingface.co). The interface is dark-themed with white text and light-colored buttons.

Header: The top navigation bar includes links for Models, Datasets, Spaces, Posts, Docs, Enterprise, Pricing, and a user profile icon.

Sidebar: On the left, there's a sidebar with a "New" button, a user profile section for "Valentine87eve" (Profile, Inbox (0), Settings, Billing, Get Pro), and an "Organizations" section.

Following: The main content area starts with a "Following" section showing 0 items. Below it are filters for All, Models, Datasets, Spaces, Papers, Collections, Community, and Posts, along with Upvotes, Likes, and Articles.

Trending: To the right, a "Trending" section for the last 7 days is displayed, showing three projects:

- perplexity-ai/r1-1776**: Updated 3 days ago • ↓ 7.03k • ❤ 1.38k
- deepseek-ai/DeepSeek-R1**: Text Generation • Updated ... • ↓ 4.4M • ⚡ 9.91k
- microsoft/OmniParser-v2.0**: Image-Text-to-Text • Updated 4 da... • ↓ 4.2k • ❤ 834

Section

트랜스포머라이브러리 소개

사용하고자 하는 모델 확인

The screenshot shows the Hugging Face platform interface for the model **koelectra-base-v3-finetuned-korquad**. The top navigation bar includes links for Models, Datasets, Spaces, Posts, Docs, Enterprise, Pricing, and a user profile icon. Below the navigation is a search bar and a sidebar with categories like Question Answering, Transformers, PyTorch, Safetensors, electra, and Inference Endpoints. The main content area displays the model card, which includes tabs for Model card, Files and versions (selected), and Community. It shows 2 contributors and a history of 7 commits. The 'Files and versions' tab lists the following files:

File	Size	Description	Last Commit
.gitattributes	399 Bytes	Adding `safetensors` variant of this model (#1)	over 1 year ago
config.json	591 Bytes	Update config.json	over 4 years ago
model.safetensors	449 MB (LFS)	Adding `safetensors` variant of this model (#1)	over 1 year ago
pytorch_model.bin	449 MB (LFS)	Update pytorch_model.bin	over 4 years ago
special_tokens_map.json	112 Bytes	Update special_tokens_map.json	over 4 years ago
tokenizer_config.json	111 Bytes	Update tokenizer_config.json	over 4 years ago
vocab.txt	263 kB	Update vocab.txt	over 4 years ago

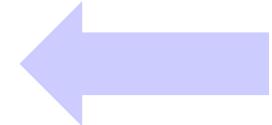
Section

트랜스포머라이브러리 소개

Transformers



Local



Model Download

The screenshot shows a web browser displaying the Hugging Face Model Hub at huggingface.co/monologg/koelectra-base-v3-finetuned-korquad/tree/main. The page title is "monologg/koelectra-base-v3-finetuned-korquad". The "Files and versions" tab is selected. The main content area shows a list of files in the "main" branch:

File	Type	Size	Last Commit
.gitattributes	Safe	399 Bytes	Adding 'safetensors' variant of this model (#1) ad4097b over 1 year ago
config.json	Safe	591 Bytes	Update config.json over 4 years ago
model.safetensors	Safe	449 MB LFS	Adding 'safetensors' variant of this model (#1) over 1 year ago
pytorch_model.bin	Safe	449 MB LFS	Update pytorch_model.bin over 4 years ago
special_tokens_map.json	Safe	112 Bytes	Update special_tokens_map.json over 4 years ago
tokenizer_config.json	Safe	111 Bytes	Update tokenizer_config.json over 4 years ago
vocab.txt	Safe	263 kB	Update vocab.txt over 4 years ago

Hugging Face

GEN AI 인텐시브 과정

Section 1. 트랜스포머 라이브러리

Section 1-2. 파이썬 기초

딕셔너리 언패킹

```
hyper_parameter = {"a": 3, "b": "birthday", "c": [1, 2, 3, 4, 5]}
```

```
def explain1(a, b, c):
    res1 = a
    res2 = b
    res3 = c
    return res1, res2, res3
```

explain1(**hyper_parameter)

딕셔너리 언패킹(unpacking)

- 딕셔너리 내부의 key-value 쌍을 풀어서 함수에 전달

(3, 'birthday', [1, 2, 3, 4, 5])

```
def explain2(param):
    return {**param, "d": 30}
```

****param** , d 가

→ 기존 key-value 쌍에
새로운 key-value 추가

explain2(hyper_parameter)

{'a': 3, 'b': 'birthday', 'c': [1, 2, 3, 4, 5], 'd': 30}

Section

파이썬기초

EvalPrediction

```
import numpy as np
from transformers.trainer_utils import EvalPrediction
```

- transformers 라이브러리에서 모델 평가 시 사용되는 예측 결과와 실제 라벨을 담는 객체
- 이후 실습에서 허깅페이스에서 Trainer가 평가할 때 자동으로 이 함수를 호출하면서 EvalPrediction 객체로 만듬

```
def accuracy_score(predict):
    preds = np.argmax(predict.predictions, axis=1)
    acc = (preds == predict.label_ids).mean()
    return {"accuracy": acc}
```

```
y_hat = np.array([[0.2, 0.8], [0.4, 0.6], [0.1, 0.9]]) 예측값
y_true = np.array([1, 0, 1]) 실제값
```

```
pred = EvalPrediction(predictions=y_hat, label_ids=y_true)
print(pred.predictions)
print('-----')
print(pred.label_ids)
```

```
[[0.2 0.8]
 [0.4 0.6]
 [0.1 0.9]]
-----
[1 0 1]
```

정확도(accuracy)만드는 함수
- 입력값 predict는 EvalPrediction 객체에 해당

y_hat

```
array([[0, 1],
       [0, 1],
       [0, 1]])
```

```
np.argmax(y_hat, axis=1)
```

```
array([1, 1, 1], dtype=int64)
```

y_hat

```
array([[0.2, 0.8],
       [0.4, 0.6],
       [0.1, 0.9]])
```

```
np.argmax(y_hat, axis=0)
```

```
array([1, 2], dtype=int64)
```

Section

파이썬기초

EvalPrediction

```
print(vars(pred)) 객체 내부 정보 확인할 때 사용(1)
```

```
{'predictions': array([[0.2, 0.8],  
예측결과 [0.4, 0.6],  
[0.1, 0.9]]), 'label_ids': array([1, 0, 1]), 'inputs': None, 'losses': None, 'elements': (array([[0.2, 0.8],  
[0.4, 0.6], 실제 정답  
[0.1, 0.9]]), array([1, 0, 1]))}
```

```
print(pred.__dict__) 객체 내부 정보 확인할 때 사용(2)
```

```
{'predictions': array([[0.2, 0.8],  
[0.4, 0.6],  
[0.1, 0.9]]), 'label_ids': array([1, 0, 1]), 'inputs': None, 'losses': None, 'elements': (array([[0.2, 0.8],  
[0.4, 0.6],  
[0.1, 0.9]]), array([1, 0, 1]))}
```

```
accuracy_score(pred)
```

```
{'accuracy': 0.6666666666666666}
```

GEN AI 인텐시브 과정

Section 1. 트랜스포머 라이브러리

Section 1-3. 공부정 분석

import 트랜스포머 라이브러리

- 모델마다 적합한 토크나이저가 다를 수 있음
- AutoTokenizer를 사용하면 모델 이름이 주어졌을 경우 적절한 토크나이저 자동 선택

[6]:

```
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification
```



- transformers 라이브러리에서 제공하는 자동 모델로더 중 하나
- 분류(classification) 태스크를 수행하는 사전 훈련된 모델을 로드하는 역할

* 토크나이저: 모델이 문장을 이해할 수 있도록 텍스트를 숫자로 변환해 토큰으로 나타내는 역할

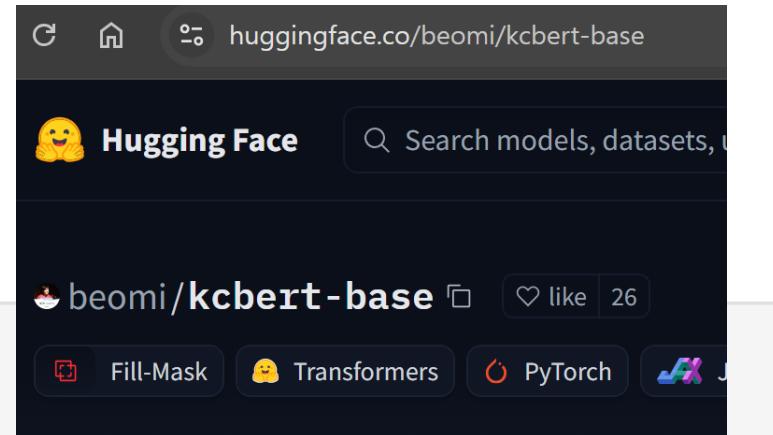
모델 불러오기

BERT 기반으로 한국어로 사전 학습된 모델

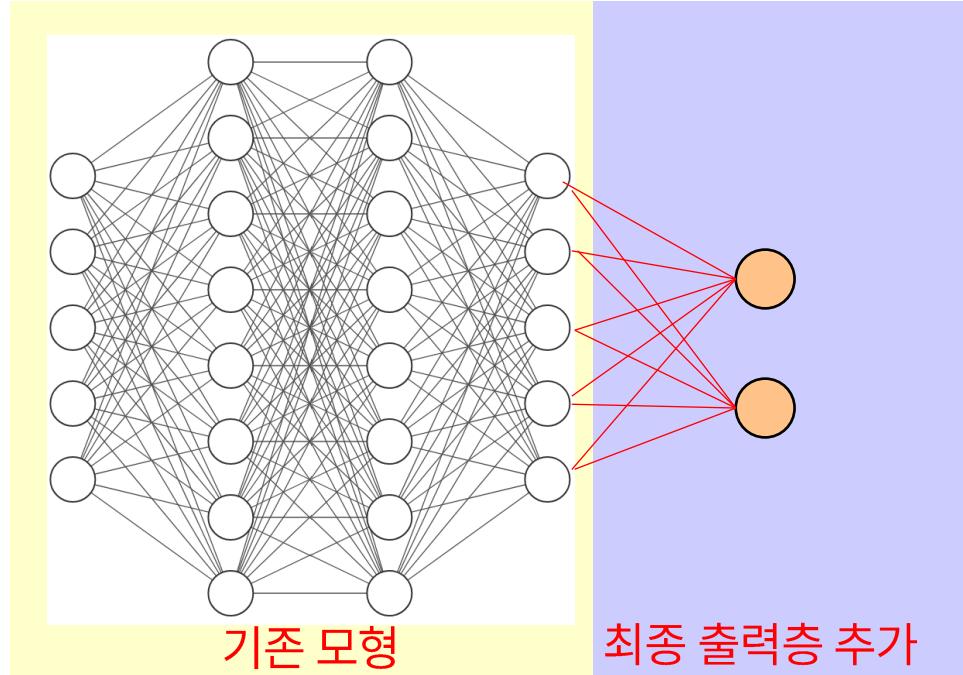
```
model_name = "beomi/kcbert-base"  
tokenizer = AutoTokenizer.from_pretrained(model_name)  
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)
```

Some weights of BertForSequenceClassification were not initialized from the model checkpoint at beomi/kcbert-base and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

→ 현재 불러온 모델 “beomi/kcbert-base”는 언어 모델이라 기본적인 문맥 이해 능력은 있으나
분류 모델이 아니므로 분류 레이어가 따로 없다는 뜻. 따라서 분류기로 사용하려면 파인튜닝 필요.
(다음장에 계속 설명)



모델 불러오기



- ```
model_name = "beomi/kcbert-base" → 사전 학습된(pretrained) 모델, 임베딩 레이어 포함
tokenizer = AutoTokenizer.from_pretrained(model_name) → 모델에 맞는 토크나이저 자동 설정
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)
```
- 분류 작업을 위해서 신경망에 마지막 출력층 추가(Adapter 아님)  
■ 이때 추가되는 출력층은 사전학습에 포함되어 있지 않으므로 가중치가 무작위로 부여됨  
■ 따라서 추후 추가학습 필요
- 긍정/부정 두개로 분류할 예정이므로 2로 설정

# 모델 저장 위치

사용자/.cache/huggingface/hub

The screenshot shows a Windows File Explorer window with the following directory path:

```
내 PC > 로컬 디스크 (C:) > 사용자 > stoic > .cache > huggingface > hub >
```

The main pane displays a list of folders and files in the 'hub' directory. The columns are '이름' (Name), '수정한 날짜' (Last modified), and '유형' (Type). The 'Name' column uses folder icons for directories and document icons for files.

| 이름                                                               | 수정한 날짜              | 유형     |
|------------------------------------------------------------------|---------------------|--------|
| .locks                                                           | 2025-04-15 오후 5:19  | 파일 폴더  |
| models--ainize--kobart-news                                      | 2025-02-02 오후 3:44  | 파일 폴더  |
| models--beomi--kcbert-base                                       | 2025-02-02 오후 4:24  | 파일 폴더  |
| models--beomi--KoAlpaca-Polyglot-5.8B                            | 2025-02-02 오후 3:29  | 파일 폴더  |
| models--deepset--roberta-base-squad2                             | 2024-12-27 오후 12:46 | 파일 폴더  |
| models--distilbert-base-uncased-finetuned-sst-2-english          | 2024-11-17 오후 2:26  | 파일 폴더  |
| models--EleutherAI--polyglot-ko-1.3b                             | 2025-04-15 오후 5:24  | 파일 폴더  |
| models--gpt2                                                     | 2024-12-27 오후 12:15 | 파일 폴더  |
| models--jghan--ko-sroberta-multitask                             | 2025-04-08 오후 4:36  | 파일 폴더  |
| models--kykim--bert-kor-base                                     | 2025-02-02 오후 4:19  | 파일 폴더  |
| models--monologg--koelectra-base-v3-finetuned-korquad            | 2025-02-02 오후 4:45  | 파일 폴더  |
| models--sentence-transformers--all-MiniLM-L6-v2                  | 2024-12-27 오후 12:19 | 파일 폴더  |
| models--sentence-transformers--paraphrase-multilingual-MiniLM... | 2025-04-15 오후 1:09  | 파일 폴더  |
| models--skt--kogpt2-base-v2                                      | 2025-04-08 오후 2:10  | 파일 폴더  |
| version.txt                                                      | 2024-11-17 오후 2:23  | 텍스트 문서 |

# 분류할 텍스트 설정

```
text = "이 영화 너무 감동적이었어! 최고야!"
inputs = tokenizer(text, return_tensors="pt")
```

inputs

```
{'input_ids': tensor([[2, 2451, 9376, 8069, 13912, 8805, 11320, 5, 8619, 4144,
5, 3]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]),
'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}
```

inputs.input\_ids → 텍스트를 토큰화 한 후 각 토큰을 숫자 ID로 변환

```
tensor([[2, 2451, 9376, 8069, 13912, 8805, 11320, 5, 8619, 4144,
5, 3]])
```

inputs.token\_type\_ids → 문장 구분 정보, 따옴표 한 쌍당 문장 1개로 인식

```
tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

inputs.attention\_mask → 실제 입력 토큰은 1, 패딩 토큰은 0, 텐서 내부에서 주목(attention)해야 할 부분, 1이면 주목, 0이면 무시

```
tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
```

return\_tensors="pt" → pytorch  
return\_tensors="tf" → tensorflow  
return\_tensors="np" → numpy

주요 토큰

| 토큰 | 의미             |
|----|----------------|
| 1  | 패딩 토큰          |
| 2  | 문장의 시작         |
| 3  | 문장의 끝 혹은 문장 구분 |

## Section

### 긍부정분석

# 주론

model.eval() → 이 코드를 생략하는 경우도 많지만, 코드 안정성을 위해 쓰는게 좋음

```
with torch.no_grad():
 outputs = model(**inputs)
```

outputs

```
SequenceClassifierOutput(loss=None, logits=tensor([[0.1845, 0.4710]]), hidden_states=None, attentions=None)
```

outputs.logits

```
tensor([[0.1845, 0.4710]]) 1번째가 높으므로 1번째 결과로 출력할건데, 문제는 1번째가 긍정인지 부정인지 모르는 상태.
```

```
logits = outputs.logits
sentiment = torch.argmax(logits).item()
```

```
print("감성 분석 결과:", "긍정" if sentiment == 1 else "부정")
```

감성 분석 결과: 긍정

임의로 1을 긍정으로 설정

추가학습 필요

감사합니다.

# Q & A