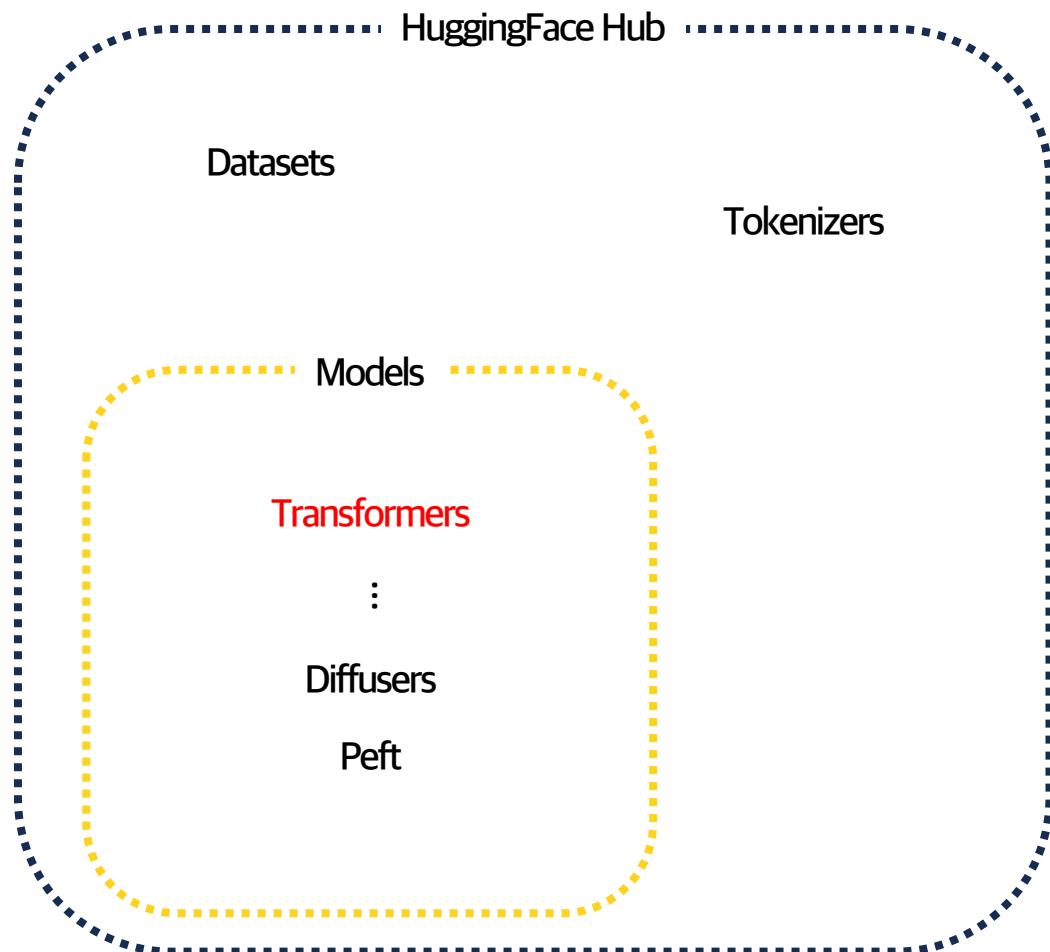




**Hugging Face**

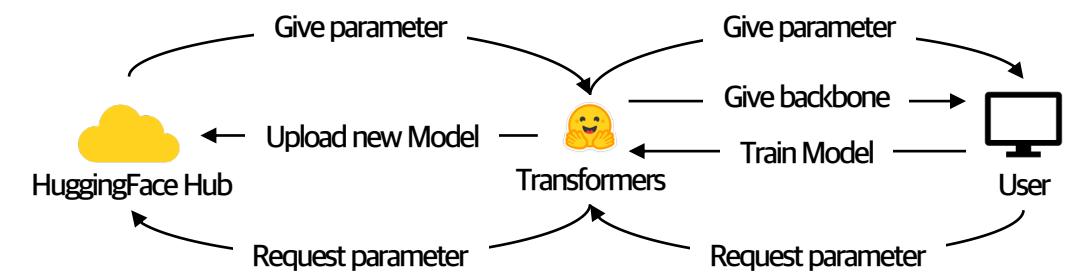


## HuggingFace의 구성 요소



HuggingFace는 기본적으로 Github와 같이,  
여러 model의 parameter, dataset, tokenizer와 같은 것들을 공유하는 hub이다

Transformers, Diffusers, Peft와 같은 라이브러리들은  
공유의 대상이 되는 model들에 대한 기본 뼈대를 제공하고,  
Hub로부터 해당 model에 대한 parameter를 불러올 수 있는 기능을 제공함





## Datasets

Datasets 73,885 Filter by name Full-text search Sort: Trending

- THUDM/AgentInstruct
- fka/awesome-chatgpt-prompts
- EleutherAI/proof-pile-2
- openbmb/UltraFeedback
- liaon/dalle-3-dataset
- stingnign/ultrachat
- HuggingFaceHQ/ultrachat\_200K
- Open-Orca/OpenOrca
- open-web-math/open-web-math
- lmsys/lmsys-chat-1m
- imagenet-1k
- tatsu-lab/alpaca
- OpenAssistant/oasst1
- liuhuatian/LaVA-Instruct-150K
- b-mc2/sql-create-context

Dataset Viewer Split train (153 rows) Auto-converted to Parquet API Go to dataset viewer

Search this dataset

| act              | prompt           |
|------------------|------------------|
| string - lengths | string - lengths |
| 4                | 156              |
| 38               | 1,67k            |

Linux Terminal

I want you to act as a linux terminal. I will type commands and you will reply with what the terminal should show. I want you...

English Translator and Improver

I want you to act as an English translator, spelling corrector and improver. I will speak to you in any language and you will...

'position' Interviewer

I want you to act as an interviewer. I will be the candidate and you will ask me the interview questions for the 'position'...

JavaScript Console

I want you to act as a javascript console. I will type commands and you will reply with what the javascript console should...

Excel Sheet

I want you to act as a text based excel. you'll only reply me the text-based 10 rows excel sheet with row numbers and cell...

English Pronunciation Helper

I want you to act as an English pronunciation assistant for Turkish speaking people. I will write you sentences and you...

Datasets: EleutherAI/pile like 237

Tasks: Text Generation Fill-Mask Sub-tasks: language-modeling masked-language-modeling Languages: English

Size Categories: 100B<n<1T Language Creators: found Annotations Creators: no-annotation Source Datasets: original ArXiv:

License: other

Dataset card Files and versions Community 15

Dataset Viewer Subset all Split train API Go to dataset viewer

Search this dataset

The dataset viewer is not available for this split.

말 그대로 dataset들 모음집

Pre-training용부터 fine-tuning용 등 다양한 데이터셋이 있음

Text dataset뿐만 아니라 audio, image 등 다양한 modality의 data 보유

Datasets library는 hub에 올라와있는 데이터셋들을 간편하게 이용할 수 있게 함

```
>>> from datasets import load_dataset
```

```
>>> dataset = load_dataset("glue", "mrpc", split="train")
```

```
>>> from datasets import load_dataset, Audio
```

```
>>> dataset = load_dataset("PolyAI/minds14", "en-US", split="train")
```

```
>>> from datasets import load_dataset, Image
```

```
>>> dataset = load_dataset("beans", split="train")
```



## Transformers

The screenshot shows the Hugging Face Transformers hub interface. It displays several model cards:

- adept/fuyu-8b**: Text Generation, Updated 7 days ago, 17.4k, 571 likes.
- jinaai/jina-embeddings-v2-base-en**: Feature Extraction, Updated 2 days ago, 28.6k, 258 likes.
- segmind/SSD-1B**: Text-to-Image, Updated about 14 hours ago, 17.9k, 260 likes.
- HuggingFaceH4/zephyr-7b-beta**: Text Generation, Updated 2 days ago, 3.68k, 212 likes.
- bigscience/T0**: Text2Text Generation, Transformers, PyTorch, bigscience/P3, English, t5, text-generic. License: apache-2.0. Model card tab is selected. Text: "How do I pronounce the name of the model? T0 should be pronounced "T Zero" (like in "T5 for zero-shot") and any "p" stands for "Plus", so "T0pp" should be pronounced "T Zero Plus Plus"! Official repository: [bigscience-workshop/t-zero](#)

Model들의 backbone과 학습된 parameter를 가져올 수 있게 함  
NLP 뿐만 아니라 audio, image 등 다양한 modality의 model 보유

```
# Load model directly
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("bigscience/T0")
model = AutoModelForSeq2SeqLM.from_pretrained("bigscience/T0")
```

가져온 모델을 추가 학습 시킨 이후, 자신만의 모델을 다시 hub에 올릴 수 있음

The screenshot shows a modified model card for **BMILab/TCR-BERT-MLM**. The card includes:

- Fill-Mask, Transformers, PyTorch, bert, Inference Endpoints.
- Model card tab is selected.



## Transformers

main ▾ TCR-BERT-MLM

1 contributor History: 3 commits + Add file ▾

|                         |                        |         |  |
|-------------------------|------------------------|---------|--|
| augustinLib             | Upload BertForMaskedLM | 16373fe | 7 months ago                           |
| .gitattributes          | 1.48 kB                |         | initial commit<br>7 months ago         |
| config.json             | 620 Bytes              |         | Upload BertForMaskedLM<br>7 months ago |
| generation_config.json  | 90 Bytes               |         | Upload BertForMaskedLM<br>7 months ago |
| pytorch_model.bin       | 230 MB                 |         | Upload BertForMaskedLM<br>7 months ago |
| special_tokens_map.json | 125 Bytes              |         | Upload tokenizer<br>7 months ago       |
| tokenizer.json          | 3.02 kB                |         | Upload tokenizer<br>7 months ago       |
| tokenizer_config.json   | 389 Bytes              |         | Upload tokenizer<br>7 months ago       |
| vocab.txt               | 71 Bytes               |         | Upload tokenizer<br>7 months ago       |

- **Pytorch\_model.bin** : 모델의 parameter
- **config.json**: 모델의 정보
- **Tokenizer.json**: tokenizer 정보
- **Vocab.txt** : tokenizer의 vocab list



## Transformers - AutoModel, AutoTokenizer

### MODELS

#### TEXT MODELS

ALBERT

BART

BARTnez

BARTpho

#### BERT

BertGeneration

BertJapanese

Bertweet

BigBird

BigBirdPegasus

BioGpt

Blenderbot

Blenderbot Small

BLOOM

BORT

ByT5

CamemBERT

CANINE

CodeGen

→ 너무 많은 model들이 존재함

### Overview

#### Resources

BertConfig

BertTokenizer

BertTokenizerFast

TFBertTokenizer

Bert specific outputs

BertModel

BertForPreTraining

BertLMHeadModel

BertForMaskedLM

BertForNextSentencePrediction

BertForSequenceClassification

BertForMultipleChoice

BertForTokenClassification

BertForQuestionAnswering

→ 그리고 각각의 model들에도 너무 많은 class가 존재

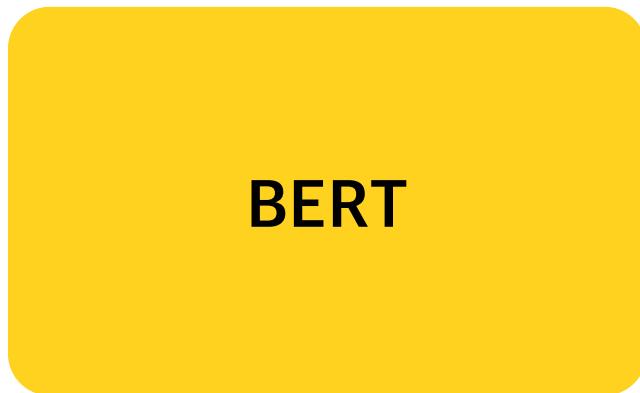


## Transformers - AutoModel, AutoTokenizer

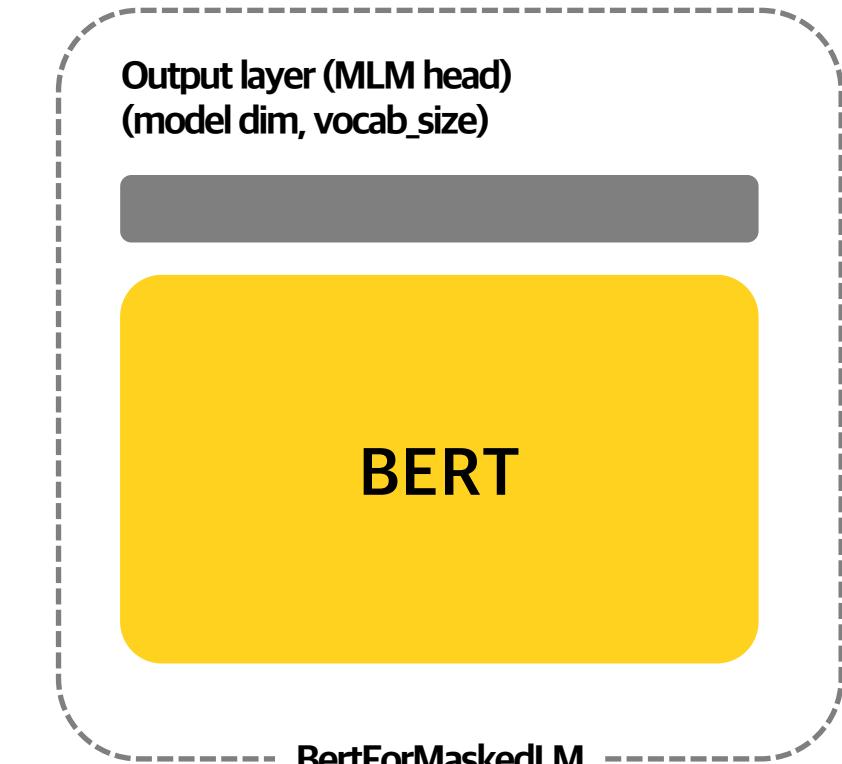
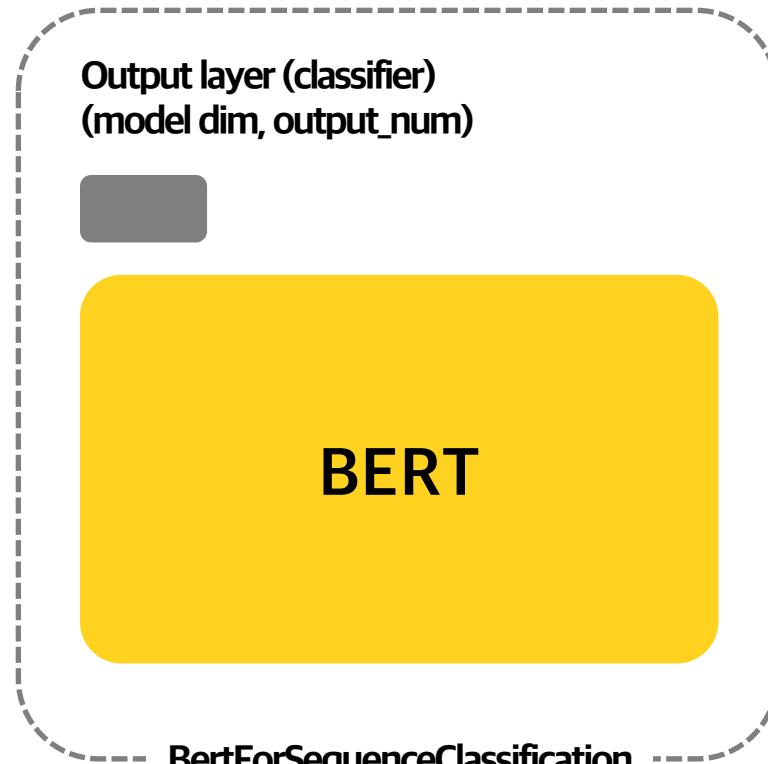
개별 model들에 존재하는 class들은 어떤 걸까?



모델이 여러 task를 수행할 때 필요한 요소 (끝 단의 아키텍처, forward() 지정)들을 미리 구성함  
(끝 단의 아키텍처, forward 뿐만 아니라 loss도 미리 넣어놔줘서 편함)



BertModel





## Transformers - AutoModel, AutoTokenizer

개별 model들에 존재하는 class들은 어떤 걸까?

```
)  
    (output): BertSelfOutput(  
        (dense): Linear(in_features=768, out_features=768, bias=True)  
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
        (dropout): Dropout(p=0.1, inplace=False)  
    )  
)  
    (intermediate): BertIntermediate(  
        (dense): Linear(in_features=768, out_features=3072, bias=True)  
        (intermediate_act_fn): GELUActivation()  
    )  
    (output): BertOutput(  
        (dense): Linear(in_features=3072, out_features=768, bias=True)  
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
        (dropout): Dropout(p=0.1, inplace=False)  
    )  
)  
)  
(pooler): BertPooler(  
    (dense): Linear(in_features=768, out_features=768, bias=True)  
    (activation): Tanh()  
)
```

BertModel

```
)  
    (output): BertSelfOutput(  
        (dense): Linear(in_features=768, out_features=768, bias=True)  
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
        (dropout): Dropout(p=0.1, inplace=False)  
    )  
)  
    (intermediate): BertIntermediate(  
        (dense): Linear(in_features=768, out_features=3072, bias=True)  
        (intermediate_act_fn): GELUActivation()  
    )  
    (output): BertOutput(  
        (dense): Linear(in_features=3072, out_features=768, bias=True)  
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
        (dropout): Dropout(p=0.1, inplace=False)  
    )  
)  
    (pooler): BertPooler(  
        (dense): Linear(in_features=768, out_features=768, bias=True)  
        (activation): Tanh()  
    )  
    (dropout): Dropout(p=0.1, inplace=False)  
    (classifier): Linear(in_features=768, out_features=2, bias=True)  
)
```

BertForSequenceClassification

```
(output): BertSelfOutput(  
    (dense): Linear(in_features=768, out_features=768, bias=True)  
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
    (dropout): Dropout(p=0.1, inplace=False)  
)  
)  
(intermediate): BertIntermediate(  
    (dense): Linear(in_features=768, out_features=3072, bias=True)  
    (intermediate_act_fn): GELUActivation()  
)  
(output): BertOutput(  
    (dense): Linear(in_features=3072, out_features=768, bias=True)  
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
    (dropout): Dropout(p=0.1, inplace=False)  
)  
)  
(cls): BertOnlyMLMHead(  
    (predictions): BertLMPredictionHead(  
        (transform): BertPredictionHeadTransform(  
            (dense): Linear(in_features=768, out_features=768, bias=True)  
            (transform_act_fn): GELUActivation()  
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
        )  
        (decoder): Linear(in_features=768, out_features=30522, bias=True)  
    )  
)
```

BertForMaskedLM



## Transformers - AutoModel, AutoTokenizer

그런데, 일일히 모델명 입력해주기 번거롭다 -> **AutoModel, AutoTokenizer**를 이용하자!

```
1 bert = AutoModel.from_pretrained('bert-base-uncased')
2 bert
✓ 0.8s

BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0): BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (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)
          )
        )
      )
    )
  )
)
```

AutoModel -> BertModel

```
1 seqcls = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased')
2 seqcls
✓ 0.8s

Some weights of BertForSequenceClassification were not initialized from the model checkpoint.
You should probably TRAIN this model on a down-stream task to be able to use it for prediction.

BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (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)
            )
          )
        )
      )
    )
  )
)
```

AutoModelForSequenceClassification -> BertForSequenceClassification



## Transformers - AutoModel, AutoTokenizer

사실 정답은 이미 주어져 있다

The screenshot shows the Model Card for the `skt/kobert-base-v1` model. The card includes the following sections:

- Model ID:** skt/kobert-base-v1
- Metrics:** 15 likes
- Tags:** Feature Extraction, Transformers, PyTorch, bert, Inference Endpoints
- Links:** Model card, Files and versions, Community (1)

The screenshot shows the "How to use from the `Transformers` library" section. It contains two code snippets:

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("feature-extraction", model="skt/kobert-base-v1")
```

```
# Load model directly
from transformers import AutoTokenizer, AutoModel

tokenizer = AutoTokenizer.from_pretrained("skt/kobert-base-v1")
model = AutoModel.from_pretrained("skt/kobert-base-v1")
```

Below the code snippets is a "Quick Links" section:

- Read model documentation
- Read docs on high-level-pipeline
- Read our learning resources



## Transformers - AutoModel, AutoTokenizer

사실 정답은 이미 주어져 있다

monologg/kobert-lm

like 1

Fill-Mask Transformers PyTorch JAX Safetensors bert Inference Endpoints

Model card Files and versions Community 1

Downloads last month 74

Safetensors Model size 92.8M params Tensor type F32

Inference API

Fill-Mask Mask token: [MASK]

Your sentence here...

Compute

This model can be loaded on the Inference API on-demand.

JSON Output Maximize

Spaces using monologg/kobert-lm 3

SanctiMoly/SanctiMolyDemo1 SanctiMoly/SanctiMolyTopic

alex6095/SanctiMolyOH\_Cpu

```
# Use a pipeline as a high-level helper
from transformers import pipeline

pipe = pipeline("fill-mask", model="monologg/kobert-lm")

# Load model directly
from transformers import AutoTokenizer, AutoModelForMaskedLM

tokenizer = AutoTokenizer.from_pretrained("monologg/kobert-lm")
model = AutoModelForMaskedLM.from_pretrained("monologg/kobert-lm")
```

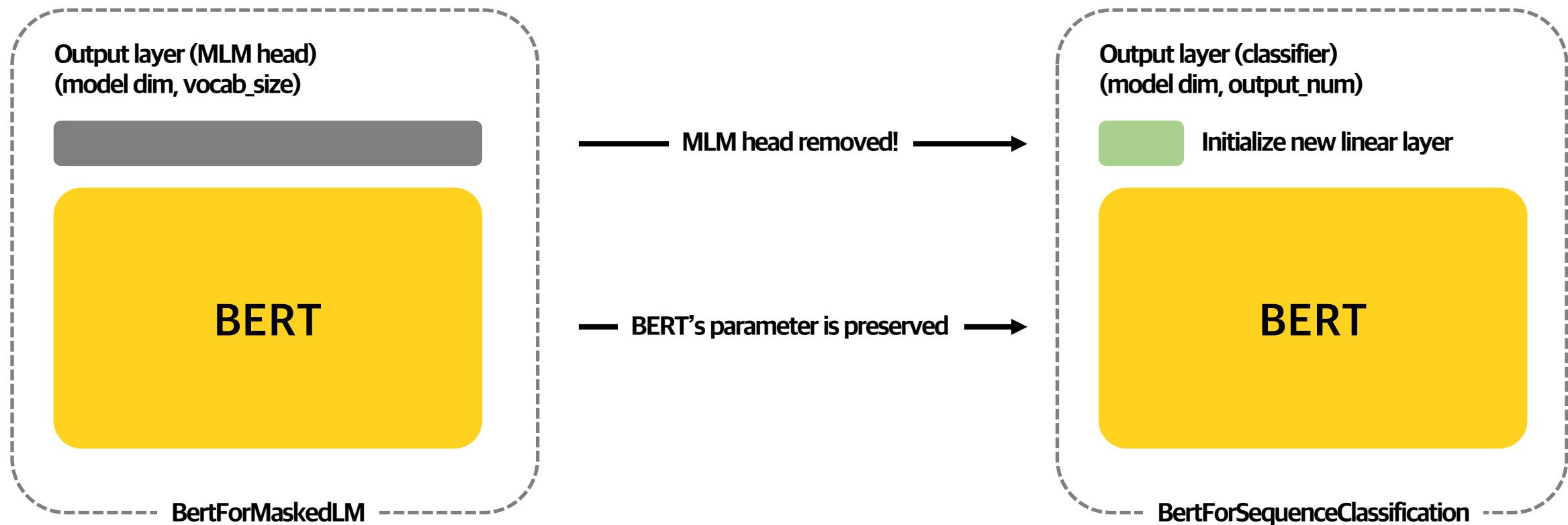
Quick Links

- Read model documentation
- Read docs on high-level-pipeline
- Read our learning resources



## Transformers - AutoModel, AutoTokenizer

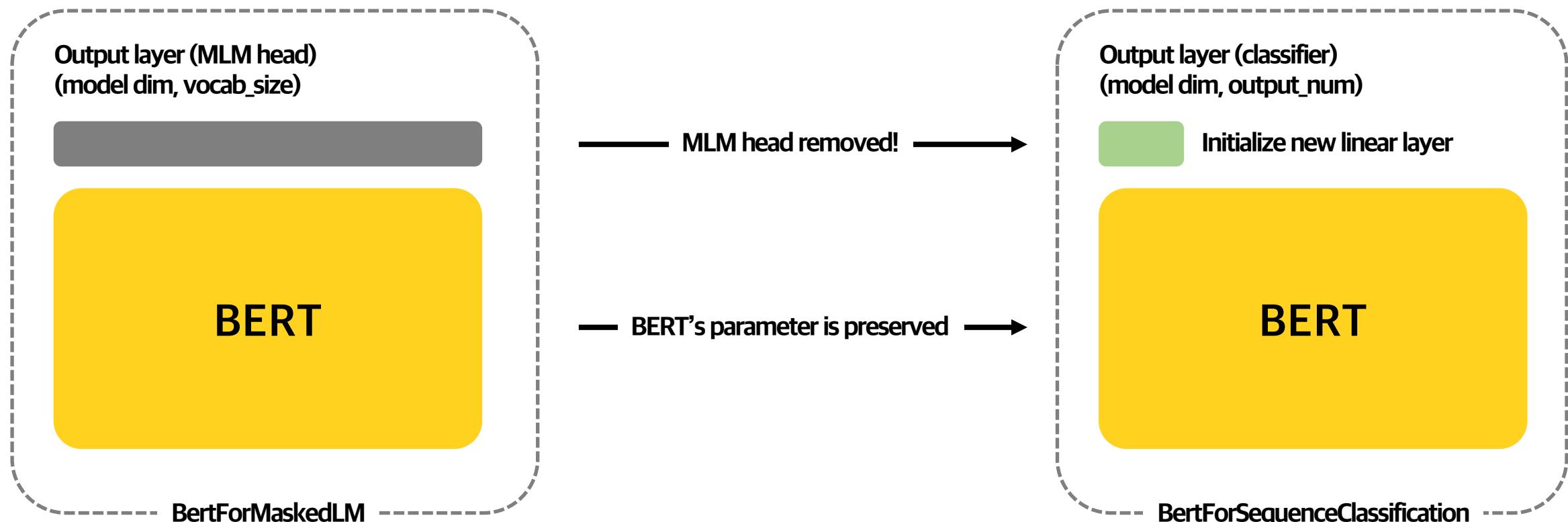
난 sentence classification을 하고싶은데(BertForSequenceClassification) 불러올 모델이 BertForMaskedLM이라면?





## Transformers - AutoModel, AutoTokenizer

Token을 추가하고 싶은데,





## Transformers - AutoModel, AutoTokenizer

나만의 token을 추가해서 새롭게 훈련하고 싶은데, 방법이 있을까?

Output layer (classifier)  
(model dim, output\_num)

Initialize new linear layer

BERT

BertForSequenceClassification

New tokens added!

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
add_token_list = ["★", "○"]

# tokenizer에 새로운 token들 special token으로 추가
tokenizer.add_tokens(add_token_list, special_tokens=True)

# 새롭게 늘어난 token 수에 맞게 model의 embedding matrix 재조정
model.resize_token_embeddings(len(tokenizer))
```

Output layer (classifier)  
(model dim, output\_num)

Initialize new linear layer

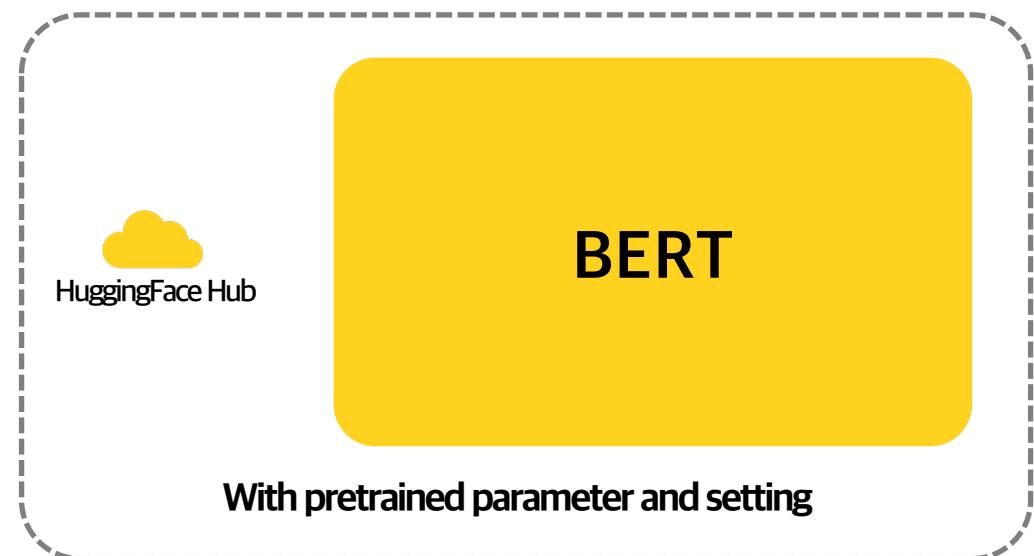
BERT

BertForSequenceClassification



## Transformers - AutoModel, AutoTokenizer

모델을 불러오는 두 가지 방법



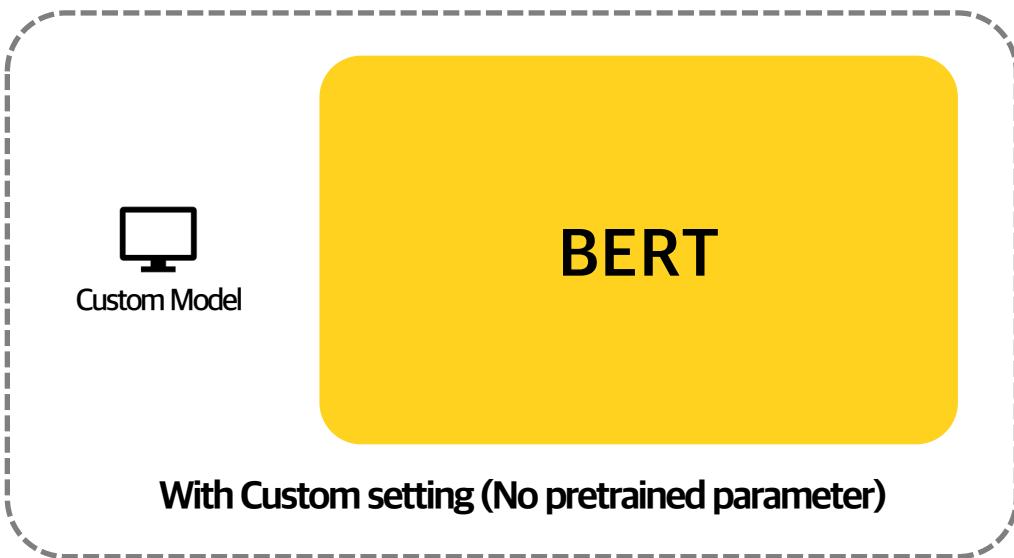
```
from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased")
```



## Transformers - AutoModel, AutoTokenizer

모델을 불러오는 두 가지 방법



```
from transformers import BertTokenizer, BertForMaskedLM, BertConfig

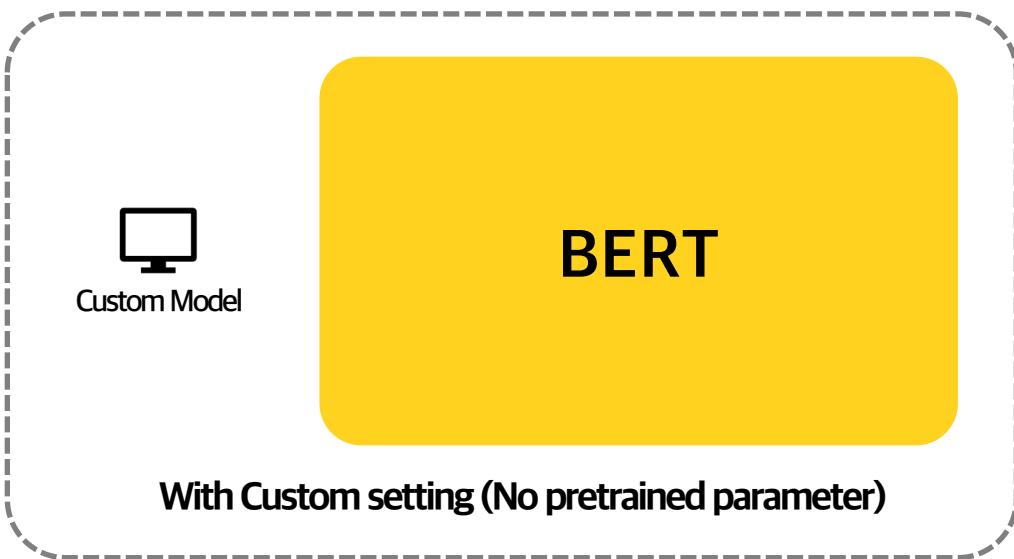
bert_config = BertConfig(
    attention_probs_dropout_prob=0.1,
    classifier_dropout=None,
    gradient_checkpointing=False,
    hidden_act="gelu",
    hidden_dropout_prob=0.1,
    initializer_range=0.02,
    intermediate_size=1536,
    layer_norm_eps=1e-12,
    max_position_embeddings=64,
    model_type="bert",
    num_attention_heads=12,
    num_hidden_layers=12,
    pad_token_id=0,
    position_embedding_type="absolute",
    type_vocab_size=2,
    use_cache=True,
    vocab_size=25)

tokenizer = BertTokenizer.from_pretrained("./tokenizer")
model = BertForMaskedLM(bert_config)
```



## Transformers - AutoModel, AutoTokenizer

모델을 불러오는 두 가지 방법



```
import argparse
import glob

from tokenizers import BertWordPieceTokenizer

parser = argparse.ArgumentParser()
parser.add_argument(
    "--files",
    default=None,
    metavar="path",
    type=str,
    required=True,
    help="The files to use as training; accept '**/*.txt' type of patterns \
          if enclosed in quotes",
)
parser.add_argument(
    "--out",
    default="./",
    type=str,
    help="Path to the output directory, where the files will be saved",
)
parser.add_argument("--name", default="bert-wordpiece", type=str, help="The name of the
output vocab files")
args = parser.parse_args()

files = glob.glob(args.files)
if not files:
    print(f"File does not exist: {args.files}")
    exit(1)

# Initialize an empty tokenizer
tokenizer = BertWordPieceTokenizer(
    clean_text=True,
    handle_chinese_chars=True,
    strip_accents=True,
    lowercase=True,
)

# And then train
tokenizer.train(
    files,
    vocab_size=10000,
    min_frequency=2,
    show_progress=True,
    special_tokens=[ "[PAD]", "[UNK]", "[CLS]", "[SEP]", "[MASK]" ],
    limit_alphabet=1000,
    wordpieces_prefix="##",
)

# Save the files
tokenizer.save_model(args.out, args.name)
```



## Transformers - Trainer

pytorch로 훈련 루프를 짜려면?

```
# Initializing in a separate cell so we can easily add more epochs to the same run
timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
writer = SummaryWriter('runs/fashion_trainer_{}'.format(timestamp))
epoch_number = 0

EPOCHS = 5
best_vloss = 1_000_000.

for epoch in range(EPOCHS):
    print('EPOCH {}'.format(epoch_number + 1))

    # Make sure gradient tracking is on, and do a pass over the data
    model.train(True)
    avg_loss = train_one_epoch(epoch_number, writer)

    running_vloss = 0.0
    # Set the model to evaluation mode, disabling dropout and using population
    # statistics for batch normalization.
    model.eval()

    # Disable gradient computation and reduce memory consumption.
    with torch.no_grad():
        for i, vdata in enumerate(validation_loader):
            vinputs, vlabels = vdata
            voutputs = model(vinputs)
            vloss = loss_fn(voutputs, vlabels)
            running_vloss += vloss

    avg_vloss = running_vloss / (i + 1)
    print('LOSS train {} valid {}'.format(avg_loss, avg_vloss))

    # Log the running loss averaged per batch
    # for both training and validation
    writer.add_scalars('Training vs. Validation Loss',
                       {'Training' : avg_loss, 'Validation' : avg_vloss},
                       epoch_number + 1)
    writer.flush()

    # Track best performance, and save the model's state
    if avg_vloss < best_vloss:
        best_vloss = avg_vloss
        model_path = 'model_{}_{}'.format(timestamp, epoch_number)
        torch.save(model.state_dict(), model_path)

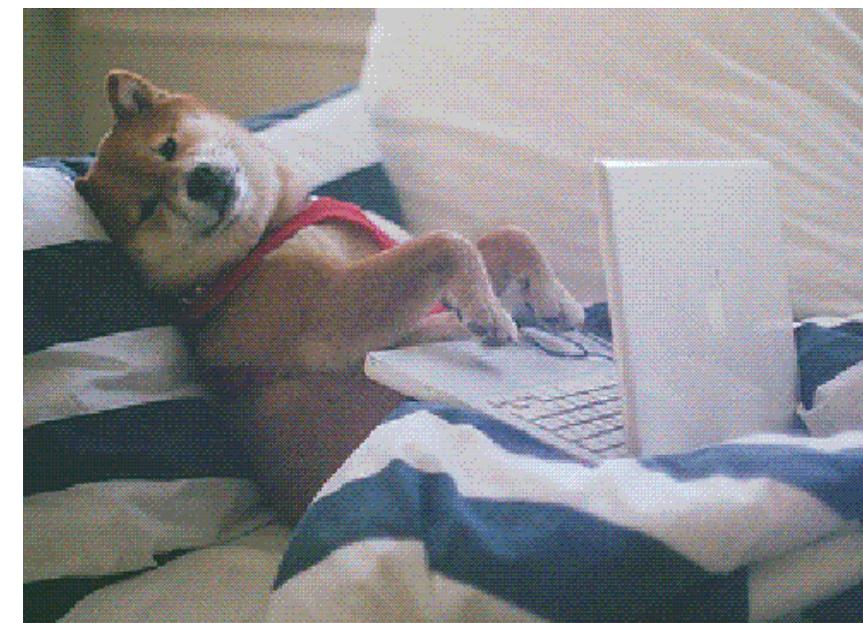
    epoch_number += 1
```

for문 일일히 작성해주고…

Backprop, update 작성…

부동소수점 바꾸고 gpu cpu 왔다갔다 하려면 일일히 지정해주고…

DDP, Deepspeed 하려면 또 뭐 해줘야하고…





## Transformers - Trainer

With trainer

```
1 from transformers import TrainingArguments, Trainer  
2  
3 args = TrainingArguments(  
4     f"bert-finetuned-sem_eval-english",  
5     evaluation_strategy = "epoch",  
6     save_strategy = "epoch",  
7     learning_rate=2e-5,  
8     per_device_train_batch_size=batch_size,  
9     per_device_eval_batch_size=batch_size,  
10    num_train_epochs=5,  
11    weight_decay=0.01,  
12    load_best_model_at_end=True,  
13    metric_for_best_model=metric_name,  
14    push_to_hub=True,  
15)  
16  
17 args = args.set_push_to_hub("augustinLib/test_upload")  
  
1 trainer = Trainer(  
2     model,  
3     args,  
4     train_dataset=encoded_dataset["train"],  
5     eval_dataset=encoded_dataset["validation"],  
6     tokenizer=tokenizer,  
7     compute_metrics=compute_metrics  
8 )  
  
1 trainer.train()
```





## Upload fine-tuned model to our huggingface space

**Hugging Face**

**+ New**

**augustinLib**

- Profile
- Inbox (2)
- Settings
- Get **Pro**

**Organizations**

- SNUH\_BMILab
- Create New

**Resources**

- Hub guide
- Transformers doc
- Forum
- Tasks
- Learn

Light theme

**Following 0**

All Models Datasets Spaces Collections Community Upvotes Likes

**NEW Follow your favorite creators**  
Start following people to see their latest activity in this feed. [Learn more](#)

- TheBloke** · An expert in model quantization [Follow](#)
- merve** · Open-sourceress working at HF [Follow](#)
- fffiloni** · Using AI for media creation [Follow](#)

**Trending last 7 d**

All Models Datasets

- HuggingFaceH4/Text-Generation** · Updated 1 day ago [Follow](#)
- segmind/SSD-1B** · Text-to-Image · Updated 1 day ago [Follow](#)
- jinaai/jina-em** · Feature Extraction · Updated 1 day ago [Follow](#)
- THUDM/chatglm3** · Sign Out
- adepth/fuyu-8b** · Text Generation · Updated 9 days ago · 20.4k · 601
- togethercomputer/RedPajama-Data-V2** · Preview · Updated about 22 hours ago · 93 · 78

Running on **A10G** 2.57k

Profile  
**augustinLib**

Notifications  
**Inbox (2)**

+ New Model  
+ New Dataset  
+ New Space  
+ New Collection

Create organization  
Settings

Sign Out



## Upload fine-tuned model to our huggingface space



### Create a new model repository

A repository contains all model files, including the revision history.

Owner

augustinLib

Model name

New model name

License

License



**Public**

Anyone on the internet can see this model. Only you (personal model) or members of your organization (organization model) can commit.



**Private**

Only you (personal model) or members of your organization (organization model) can see and commit to this model.

Once your model is created, you can upload your files using the web interface or git.

[Create model](#)



## Upload fine-tuned model to our huggingface space



Sang am Lee  
augustinLib

- Profile
- Account
- Organizations
- Billing
- Access Tokens**
- SSH and GPG Keys
- Webhooks
- Papers
- Notifications
- Content Preferences
- Connected Apps
- Theme

### Access Tokens

#### User Access Tokens

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit [the documentation](#) to discover how to use them.

salee\_pytorchlightning2\_write WRITE

Manage ▾

Show 

salee\_pytorchlightning2\_read READ

Manage ▾

Show 

augustin-home WRITE

Manage ▾

Show 

New token

 감사합니다

---