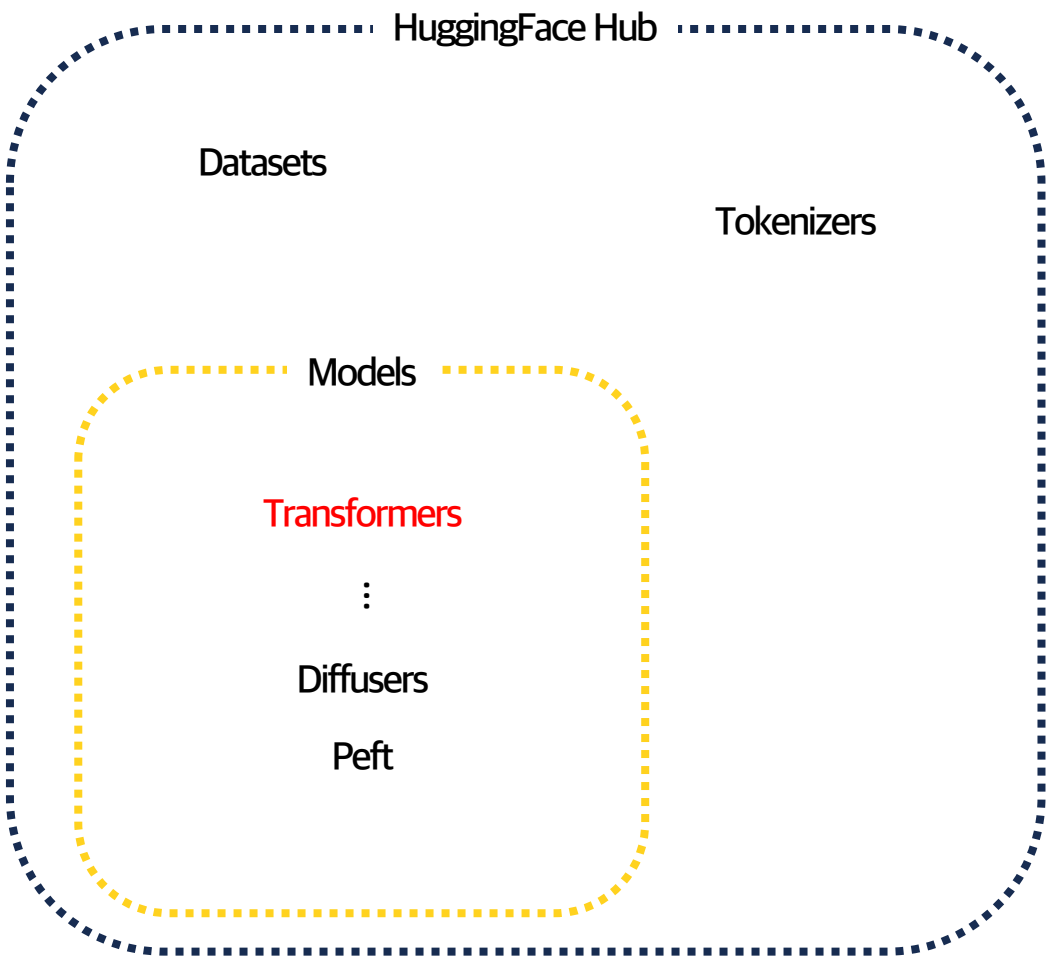
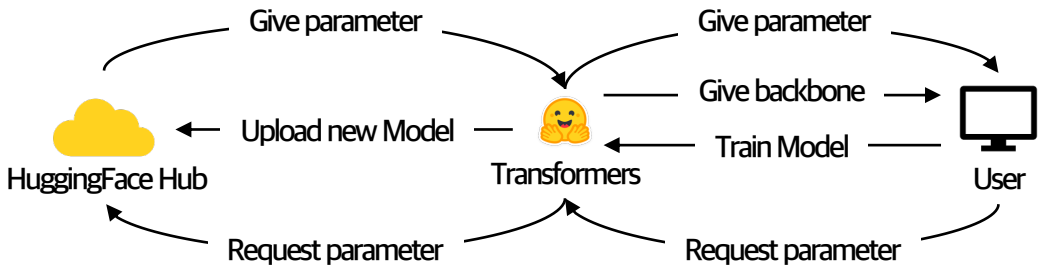
Hugging Face

HuggingFace의 구성 요소



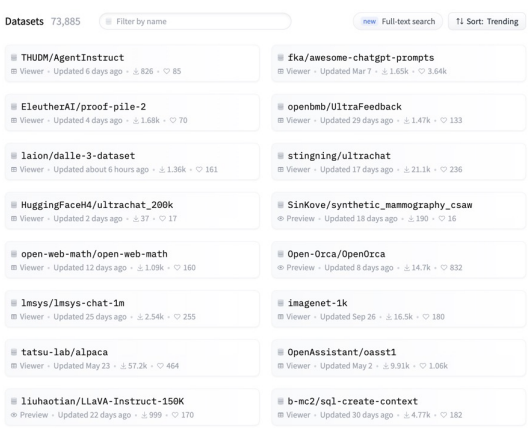
HuggingFace는 기본적으로 Github와 같이,
여러 model의 parameter, dataset, tokenizer와 같은 것들을 공유하는 hub이다

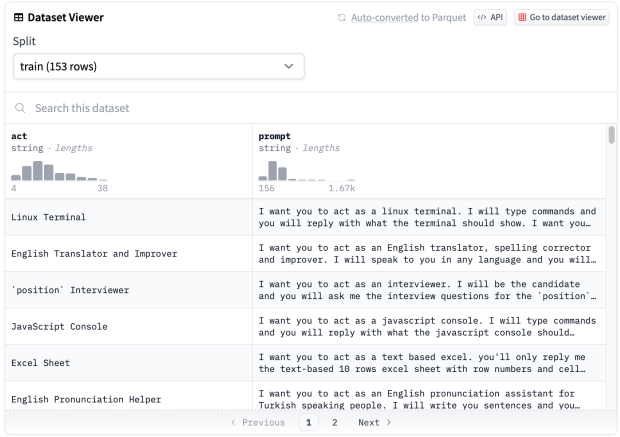
Transformers, Diffusers, Peft와 같은 라이브러리들은
공유의 대상이 되는 model들에 대한 기본 뼈대를 제공하고,
Hub로부터 해당 model에 대한 parameter를 불러올 수 있는 기능을 제공함

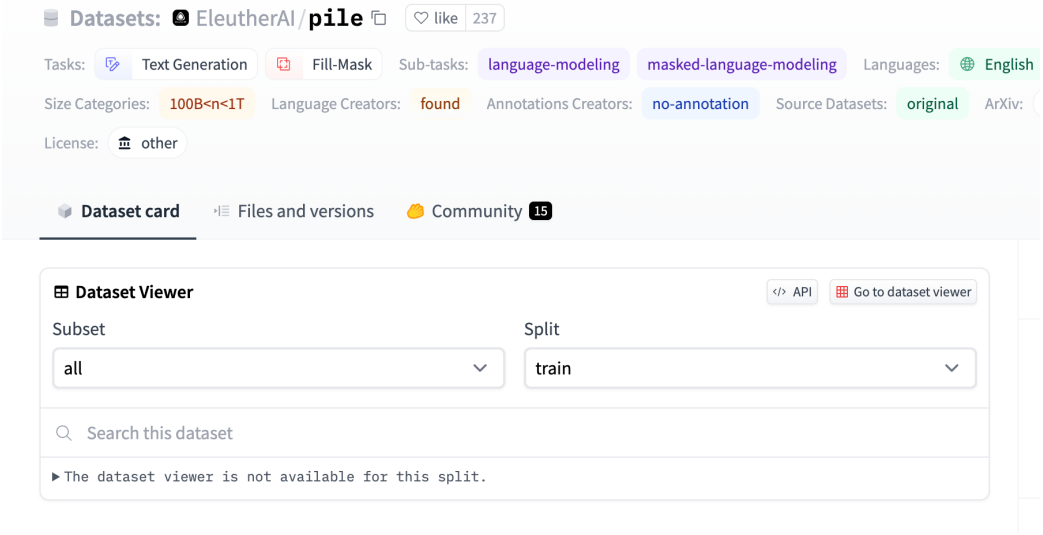


Hugging Face

Datasets







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

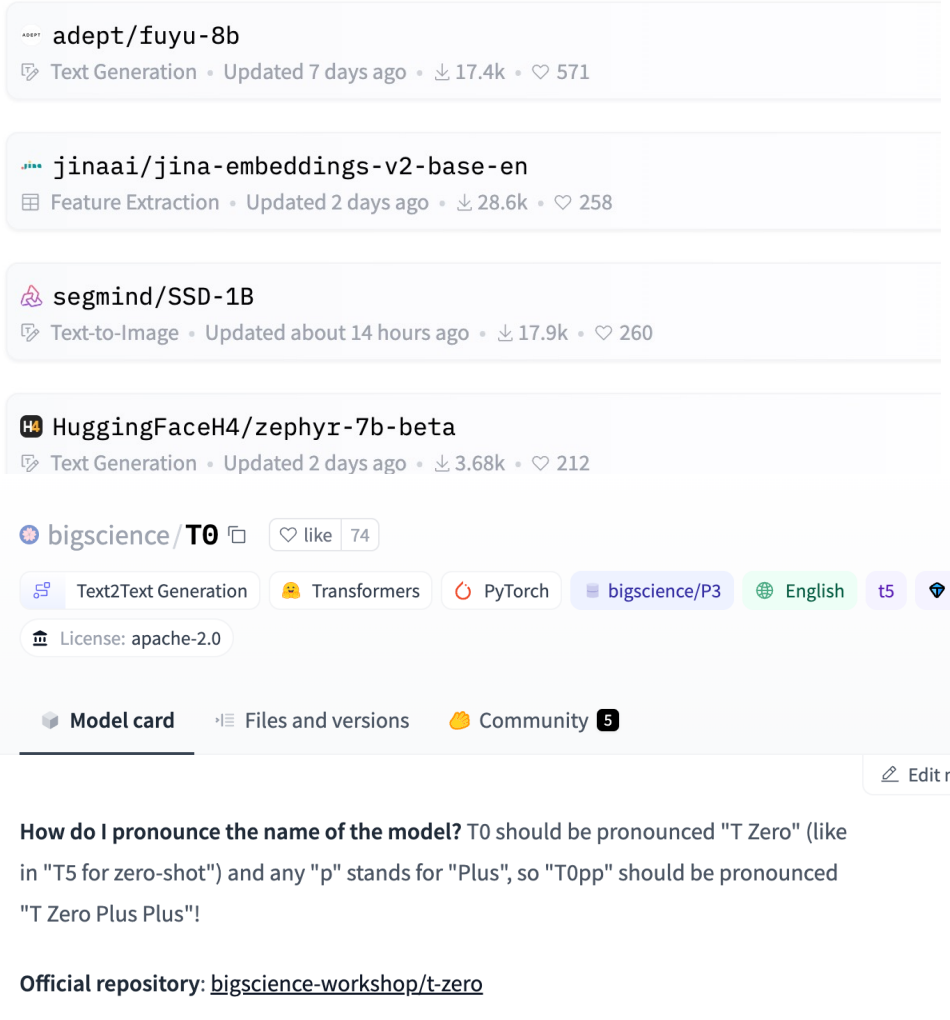
```
>>> from datasets import load_dataset

>>> 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")
```

Hugging Face

Transformers

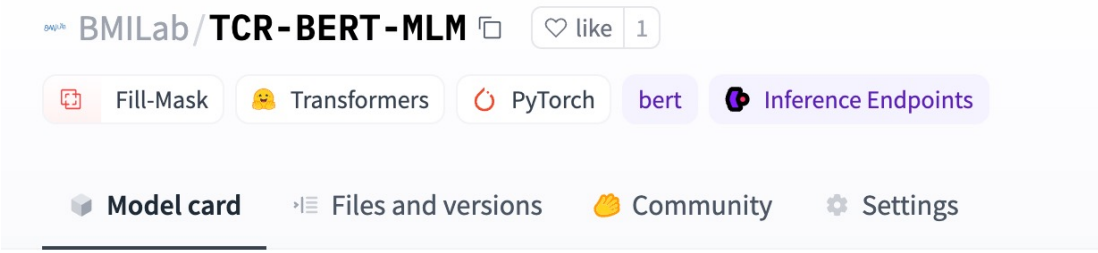


Model들의 backbone과 학습된 parameter를 가져올 수 있게함

NLP 뿐만 아니라 audio, image 등 다양한 modality의 model 보유

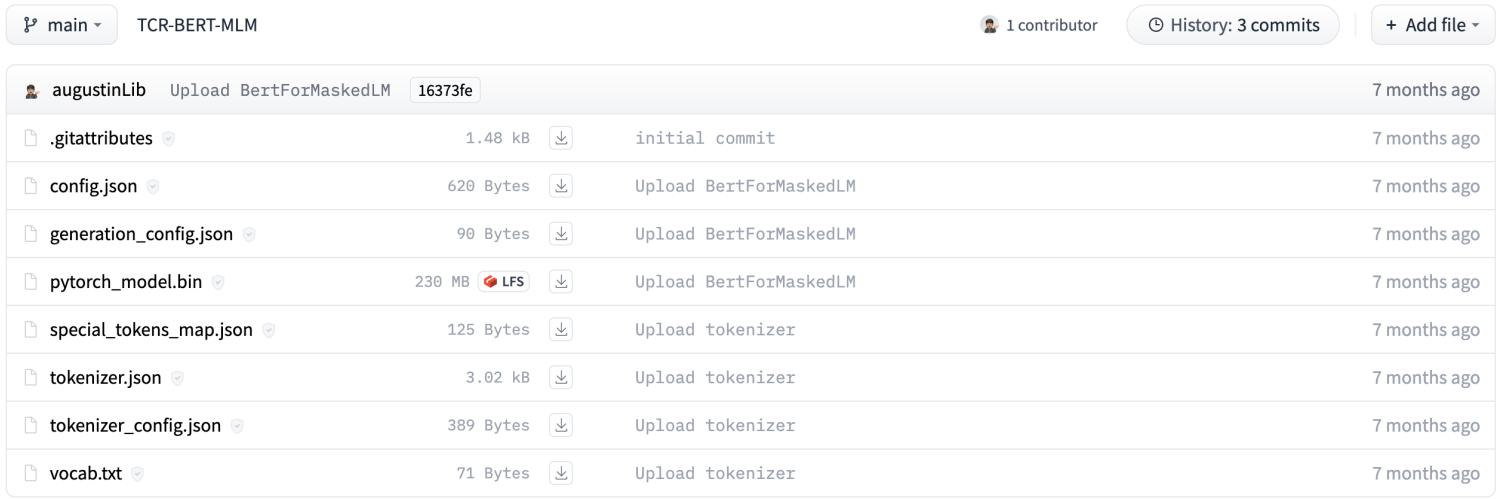
```python
# Load model directly
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

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



Hugging Face

Transformers



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

Hugging Face

Transformers - AutoModel, AutoTokenizer

MODELS

TEXT MODELS

- ALBERT
- BART
- BARThez
- 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

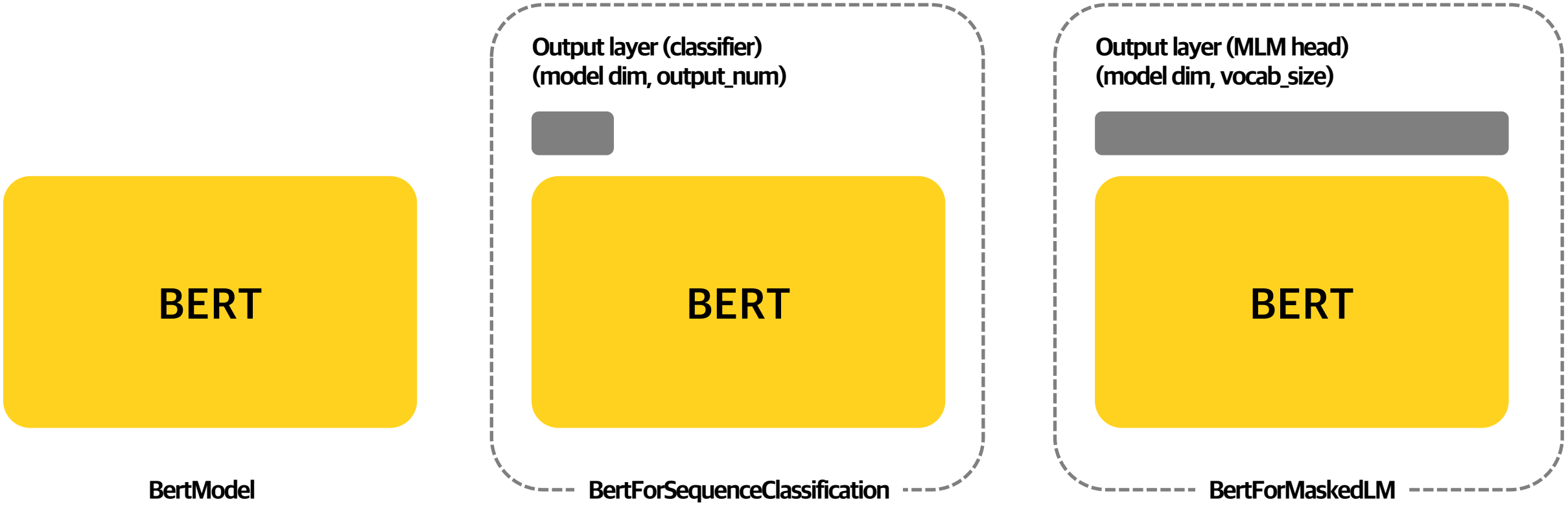
그리고 각각의 model들에도 너무 많은 class가 존재

Transformers - AutoModel, AutoTokenizer

개별 model들에 존재하는 class들은 어떤 걸까?

모델이 여러 task를 수행할 때 필요한 요소 (끝 단의 아키텍쳐, forward() 지정) 들을 미리 구성함

(끝 단의 아키텍쳐, forward 뿐만 아니라 loss도 미리 넣어놔줘서 편함)



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
```

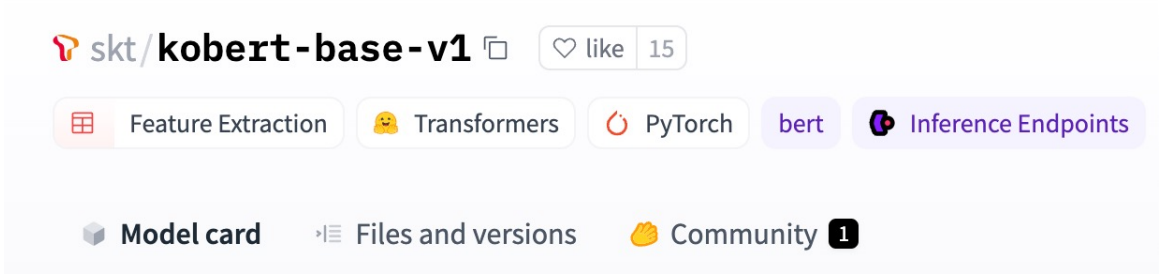
✓ 0.8s

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

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(
```

AutoModelForSequenceClassification -> BertForSequenceClassification

Transformers - AutoModel, AutoTokenizer

사실 정답은 이미 주어져 있다



How to use from the Transformers library

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

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

Copy

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

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

Copy

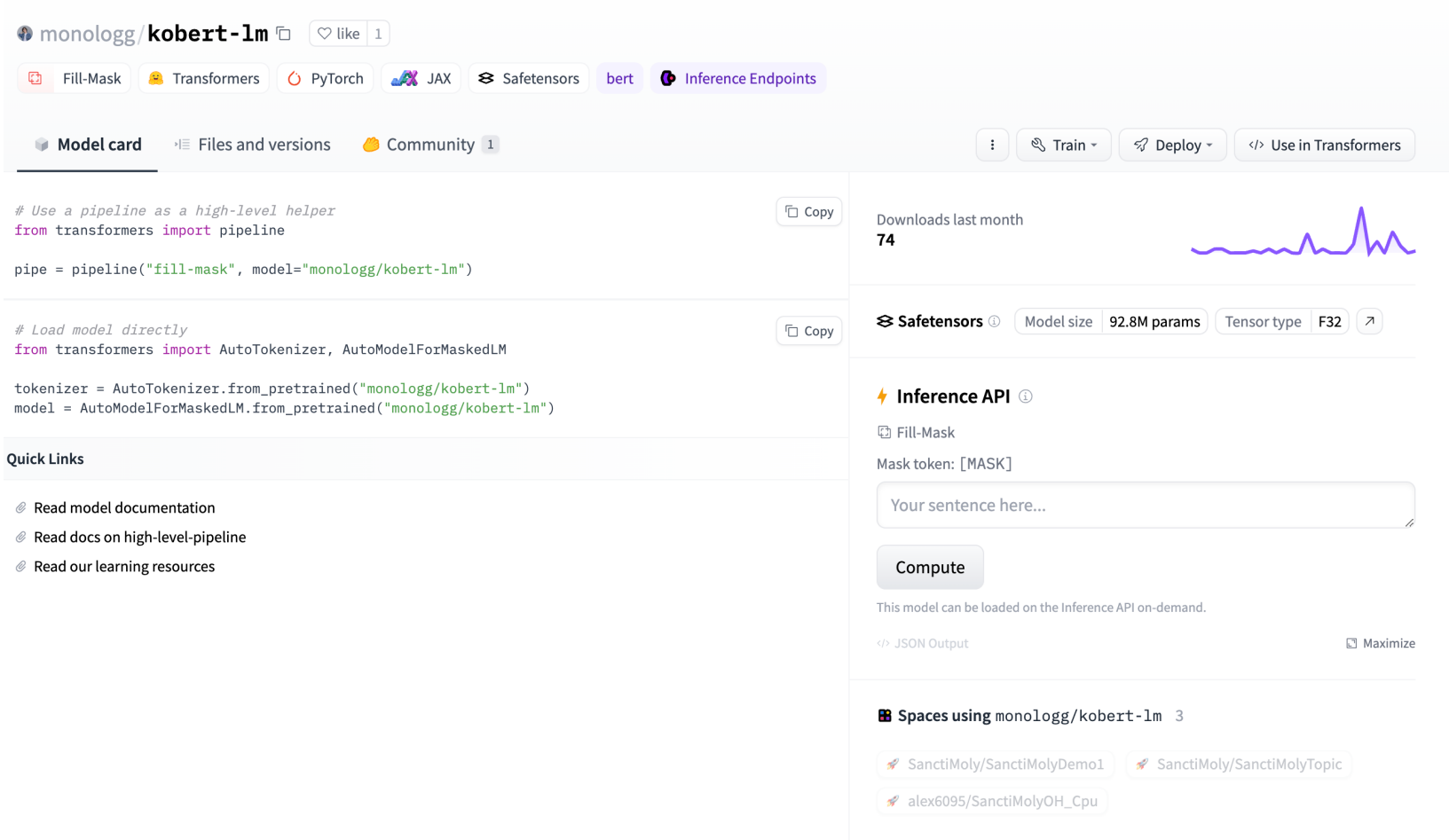
Quick Links

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

Hugging Face

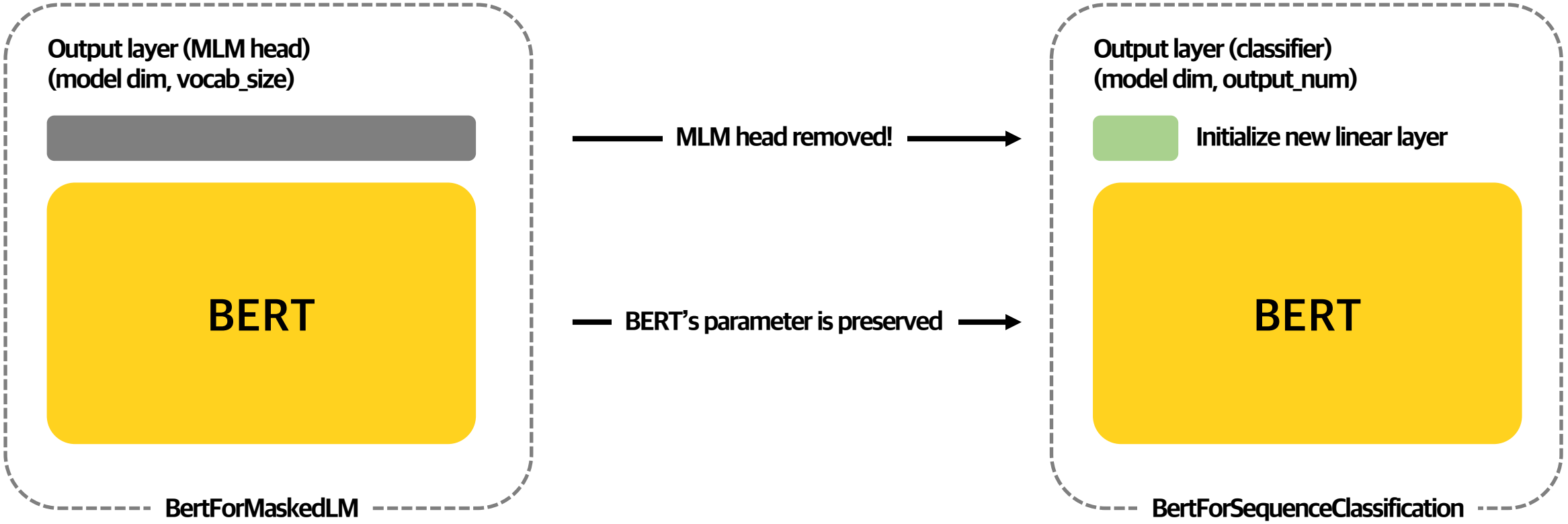
Transformers - AutoModel, AutoTokenizer

사실 정답은 이미 주어져 있다



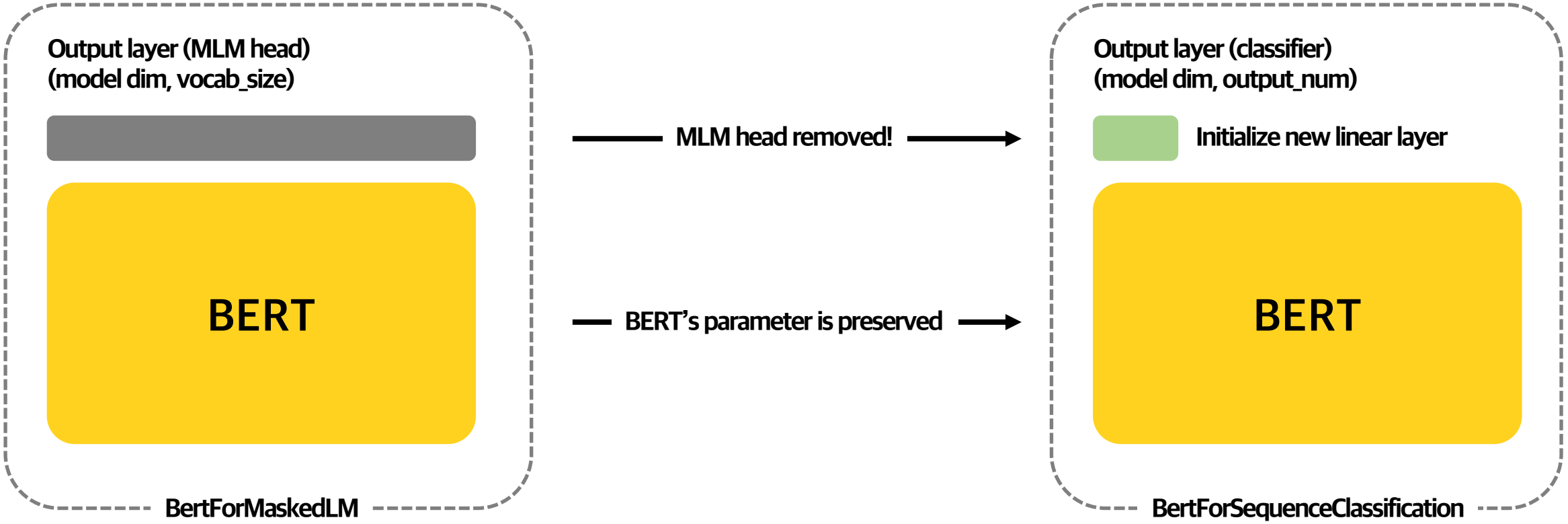
Transformers - AutoModel, AutoTokenizer

난 sentence classfication을 하고싶은데(BertForSequenceClassification) 불러올 모델이 BertForMaskedLM이라면?



Transformers - AutoModel, AutoTokenizer

Token을 추가하고 싶은데,

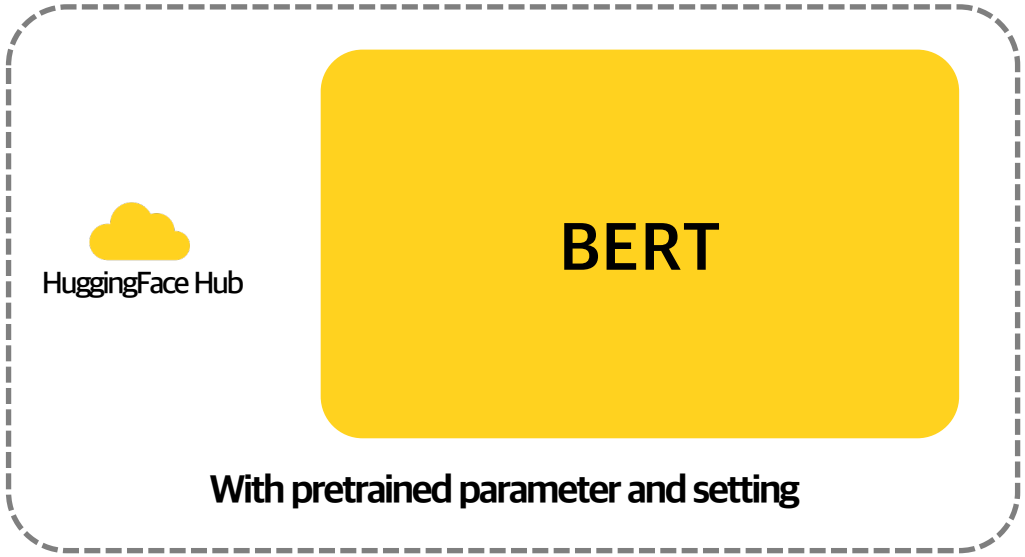


Transformers - AutoModel, AutoTokenizer

나만의 token을 추가해서 새롭게 훈련하고 싶은데, 방법이 있을까?

Transformers - AutoModel, AutoTokenizer

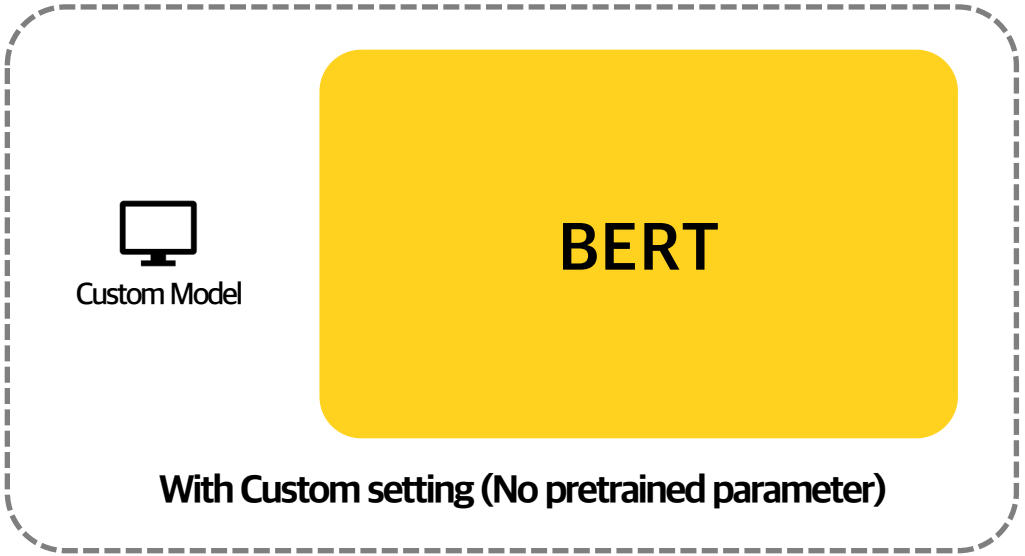
모델을 불러오는 두 가지 방법




```python
from transformers import AutoTokenizer, AutoModelForSequenceClassification


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

Transformers - AutoModel, AutoTokenizer

모델을 불러오는 두 가지 방법

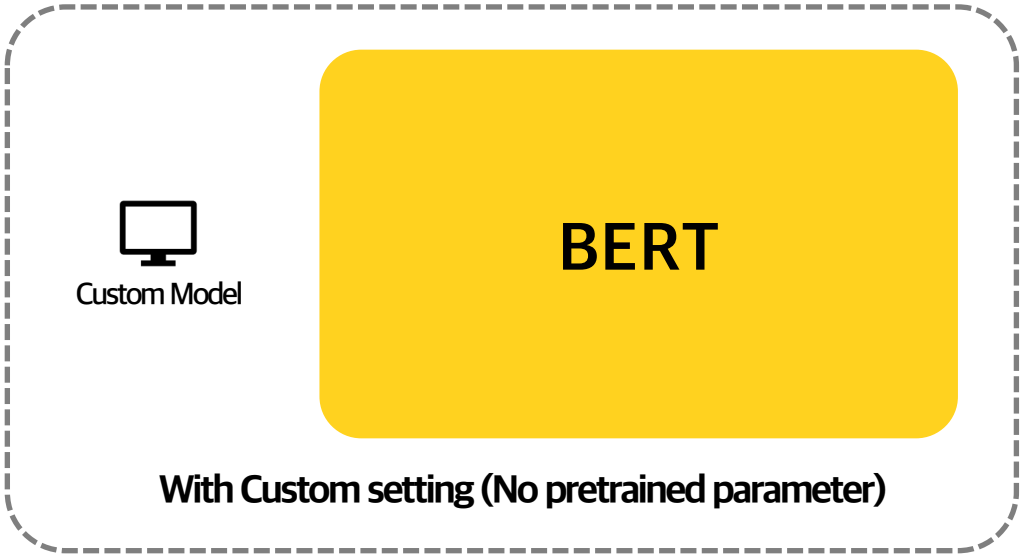



```python
from transformers import BertTokenizer, BertForMaskedLM, BertConfig


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)
```

Hugging Face

Transformers - AutoModel, AutoTokenizer

모델을 불러오는 두 가지 방법




```python
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)
```

Hugging Face

Transformers - Trainer

pytorch로 훈련 루프를 짜려면?

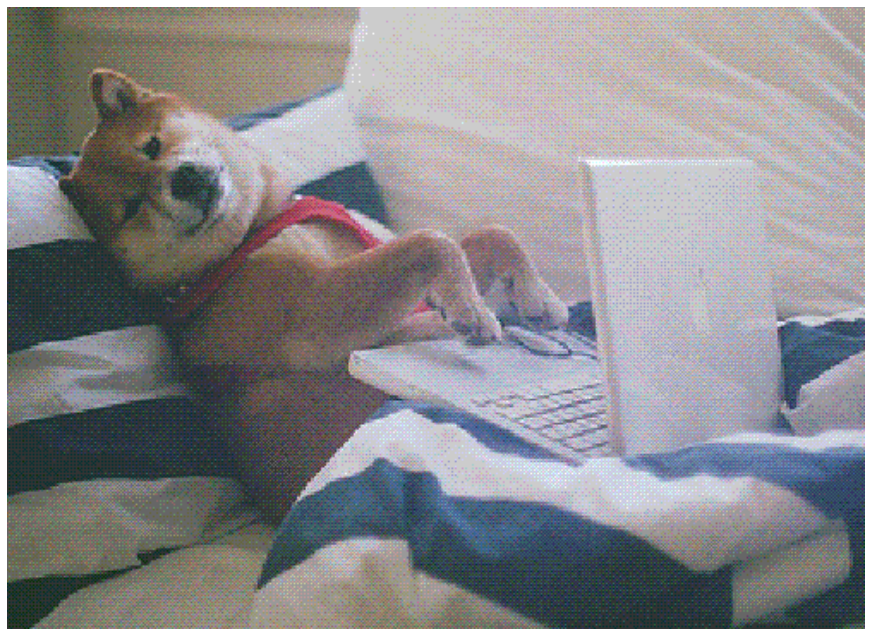
```
# Initializing in a separate cell so we can easily add more epochs to the same run
timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
writer = SummaryWriter('runs/fashion_trainer_{}'.format(timestamp))
epoch_number = 0

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

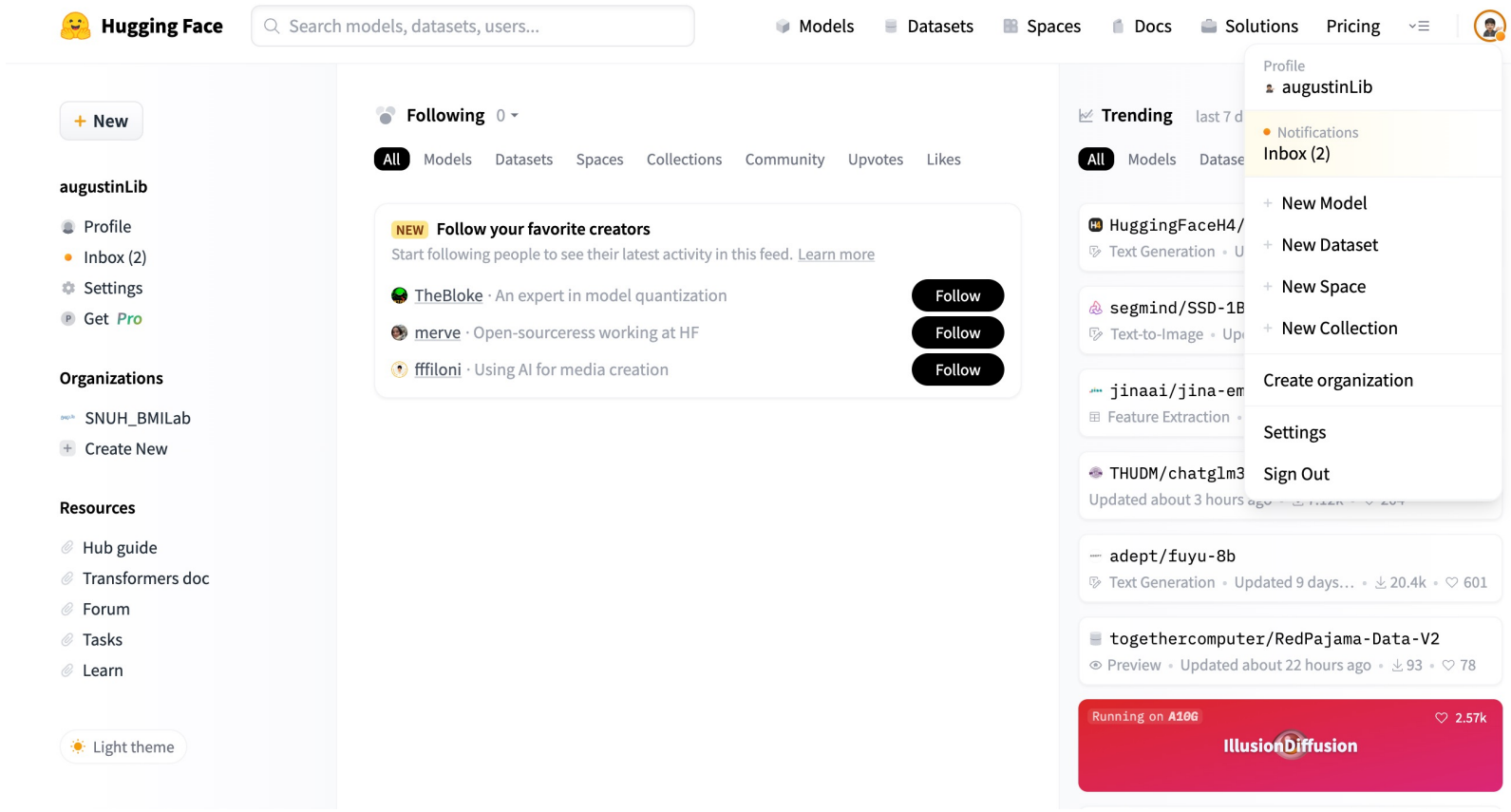
```python
from transformers import TrainingArguments, Trainer

args = TrainingArguments(
    f"bert-finetuned-sem_eval-english",
    evaluation_strategy = "epoch",
    save_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=5,
    weight_decay=0.01,
    load_best_model_at_end=True,
    metric_for_best_model=metric_name,
    push_to_hub=True,
)

args = args.set_push_to_hub("augustinLib/test_upload")
```

```python
trainer = Trainer(
    model,
    args,
    train_dataset=encoded_dataset["train"],
    eval_dataset=encoded_dataset["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

```python
trainer.train()
```

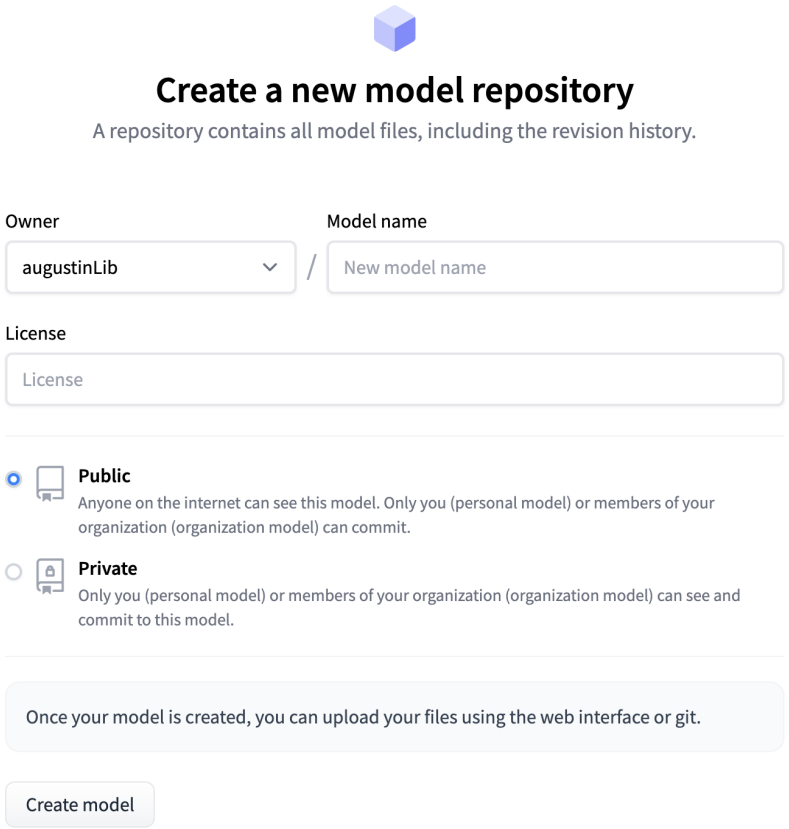
Hugging Face

Upload fine-tuned model to our huggingface space



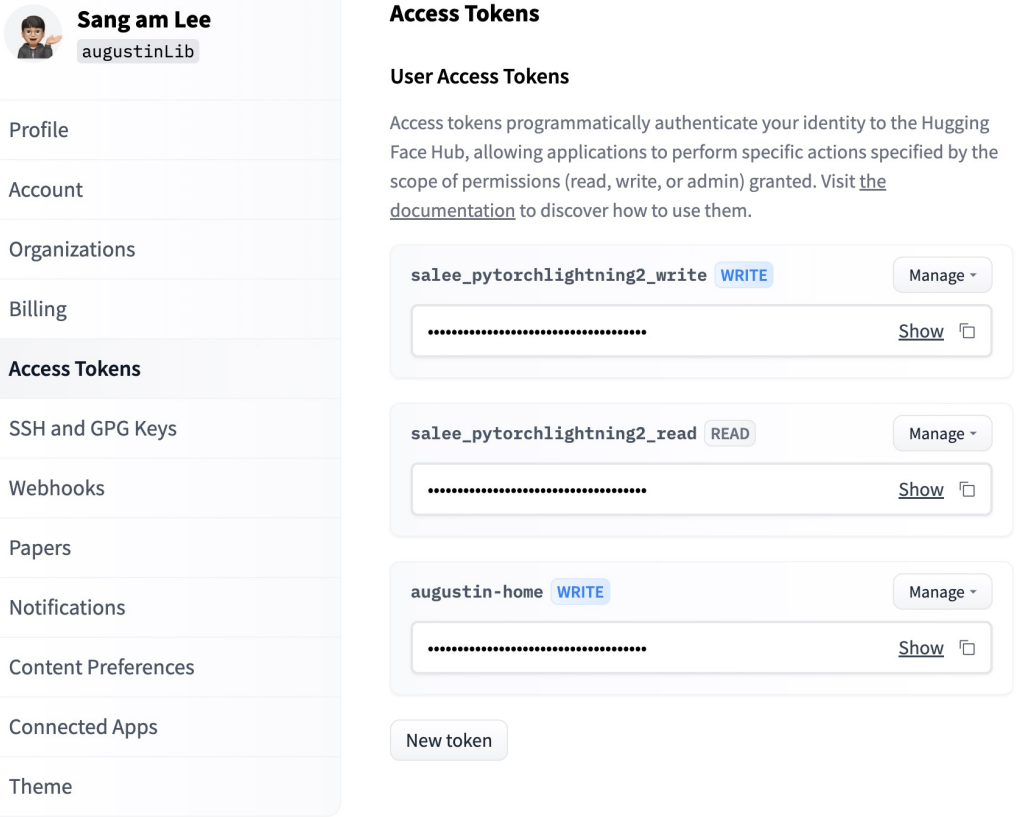
Hugging Face

Upload fine-tuned model to our huggingface space



Hugging Face

Upload fine-tuned model to our huggingface space



🙇감사합니다