

오픈소스 LLM 활용을 위한 Huggingface 알아보기

소개 및 실습

LEAD기술Task 김기수 연구원

2024. 01. 12

목차

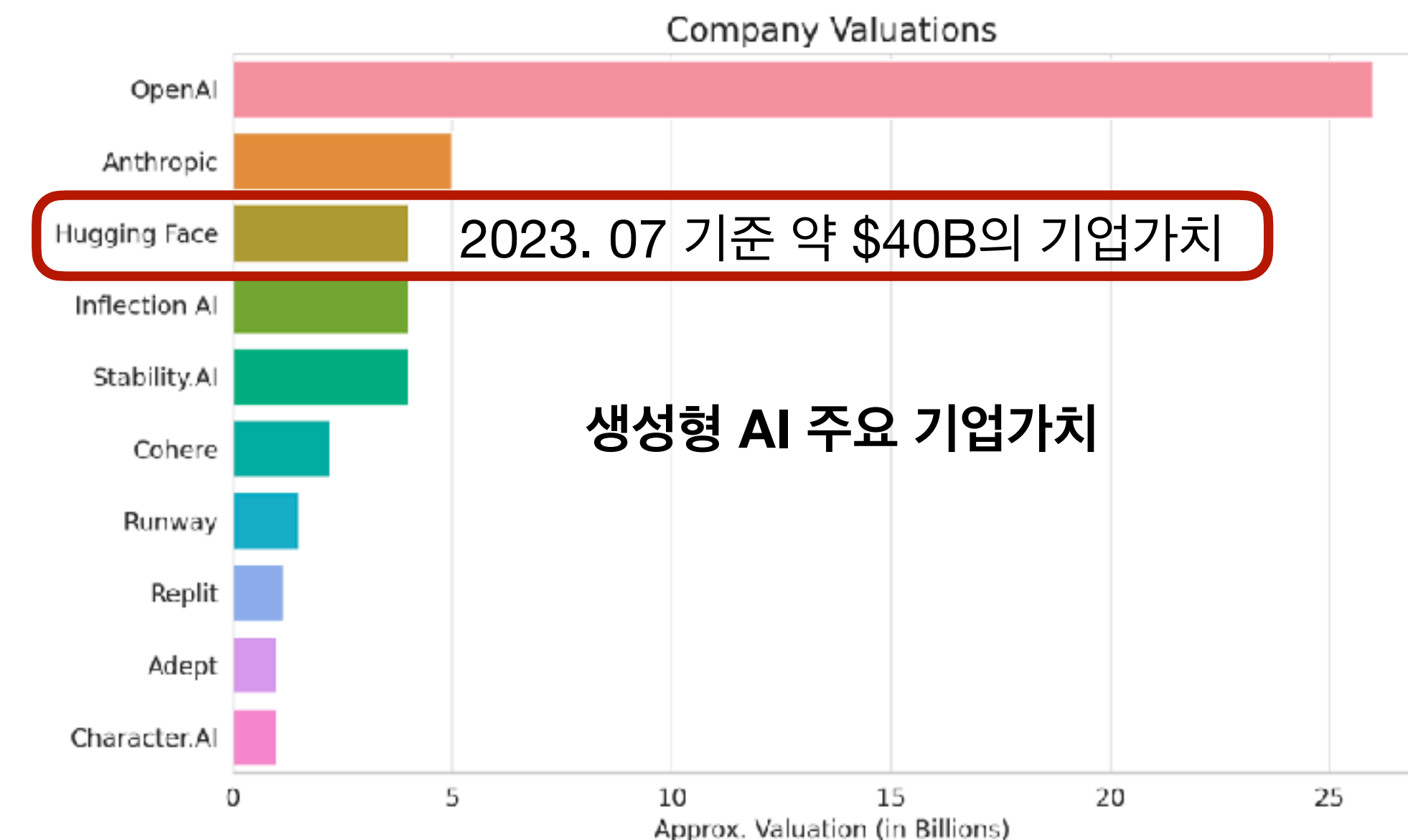
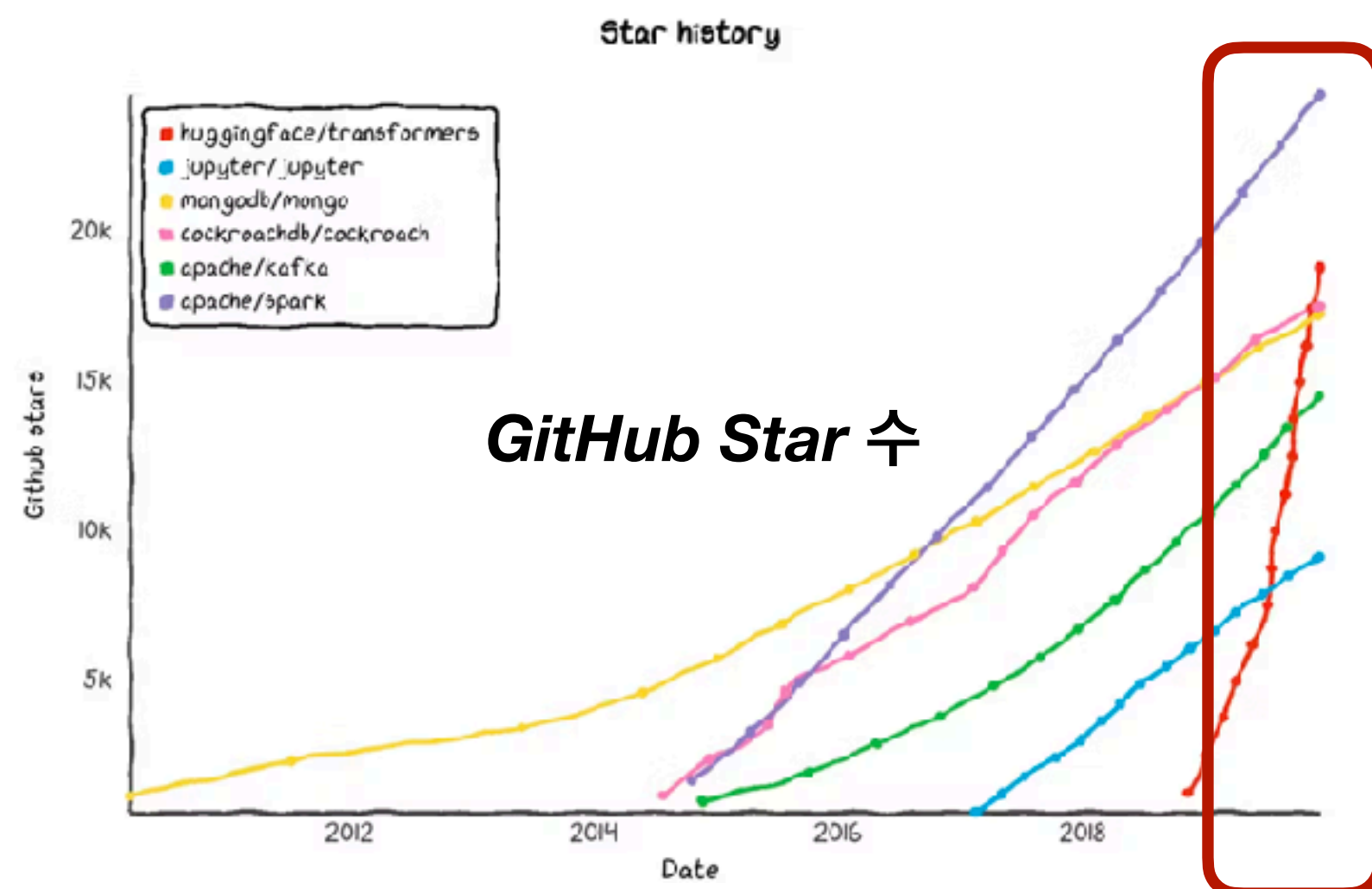
1. Huggingface 소개
2. Huggingface의 API
3. pipeline 모듈을 활용한 실습
 1. 감정 분석 (Sentiment Analysis)
 2. 질의 응답 (Question Answering)
4. 사전학습 모델 미세조정 (Fine-tuning)
5. 의견



Huggingface 소개

자연어 처리 및 기계학습 (NLP and ML) 을 위한 오픈소스 플랫폼

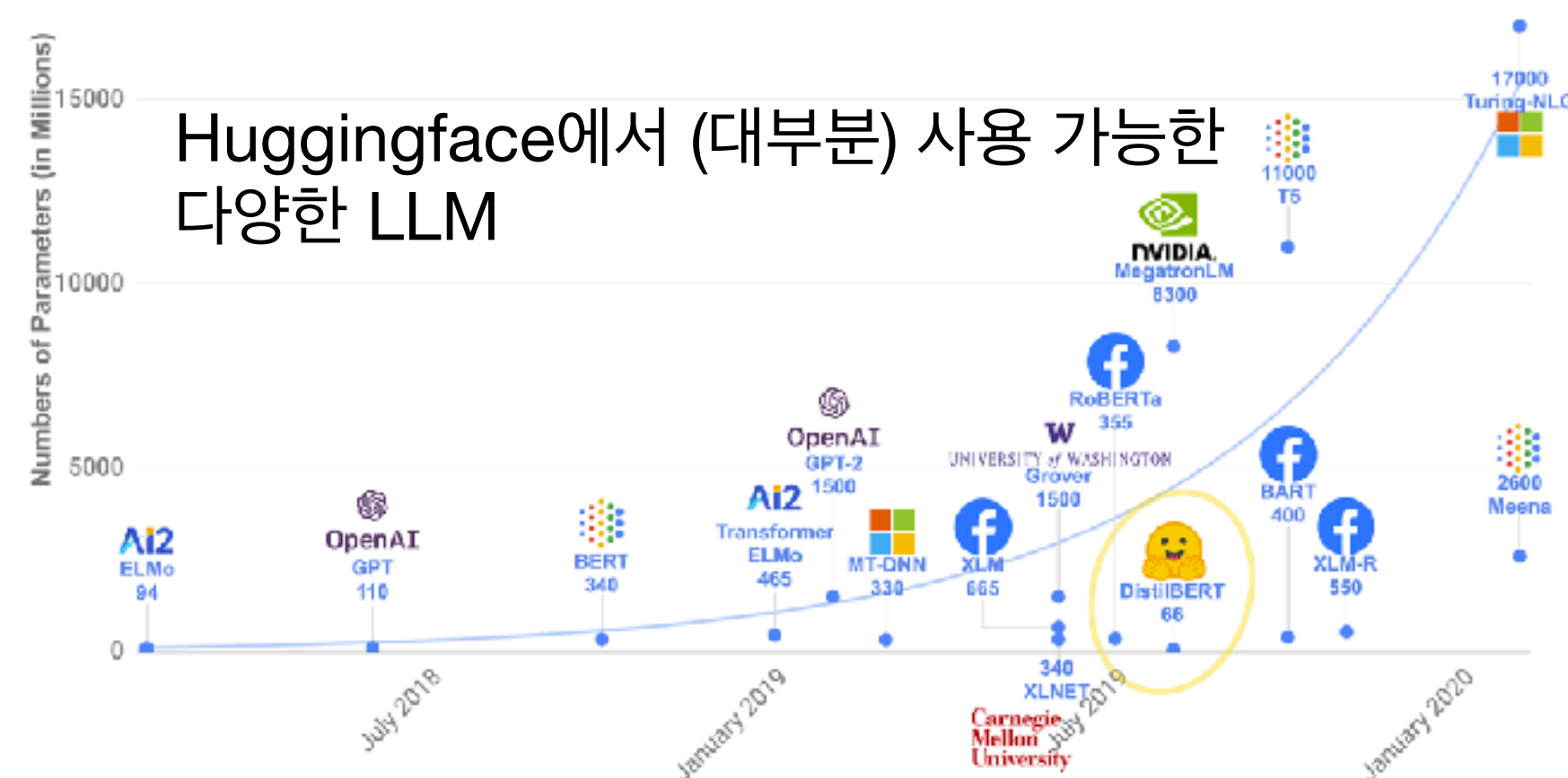
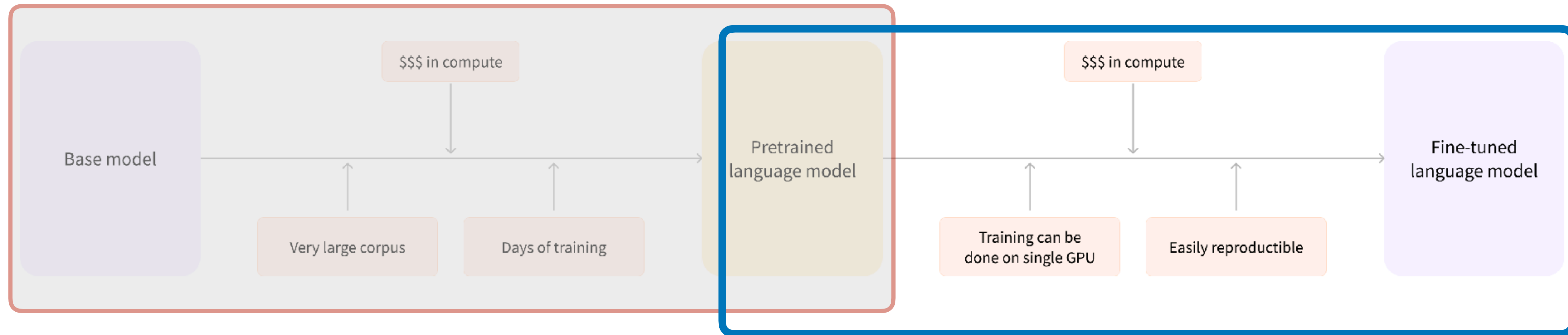
- 2016년 뉴욕을 거점으로 프랑스 기업가들이 설립
- 청소년을 타겟으로 하는 챗봇 회사로 시작했으나, ML 플랫폼 개발로 **기존 사업 아이템을 포기하고 방향 전환** **피봇팅 (Pivoting)**
- 대표적인 제품 (Product) 인 **transformers** 라이브러리
- BERT, GPT-3 등 초거대 언어모델의 근간이 되는 Transformers 모델 제공
➡ 사전 학습 모델을 API로 제공하므로 연산 비용 절감 및 빠른 적용 가능



🤗 Huggingface 소개

사전학습 모델 활용의 장점

대규모 데이터 및 컴퓨팅 자원으로 사전학습 (Pre-training) 시키는 과정

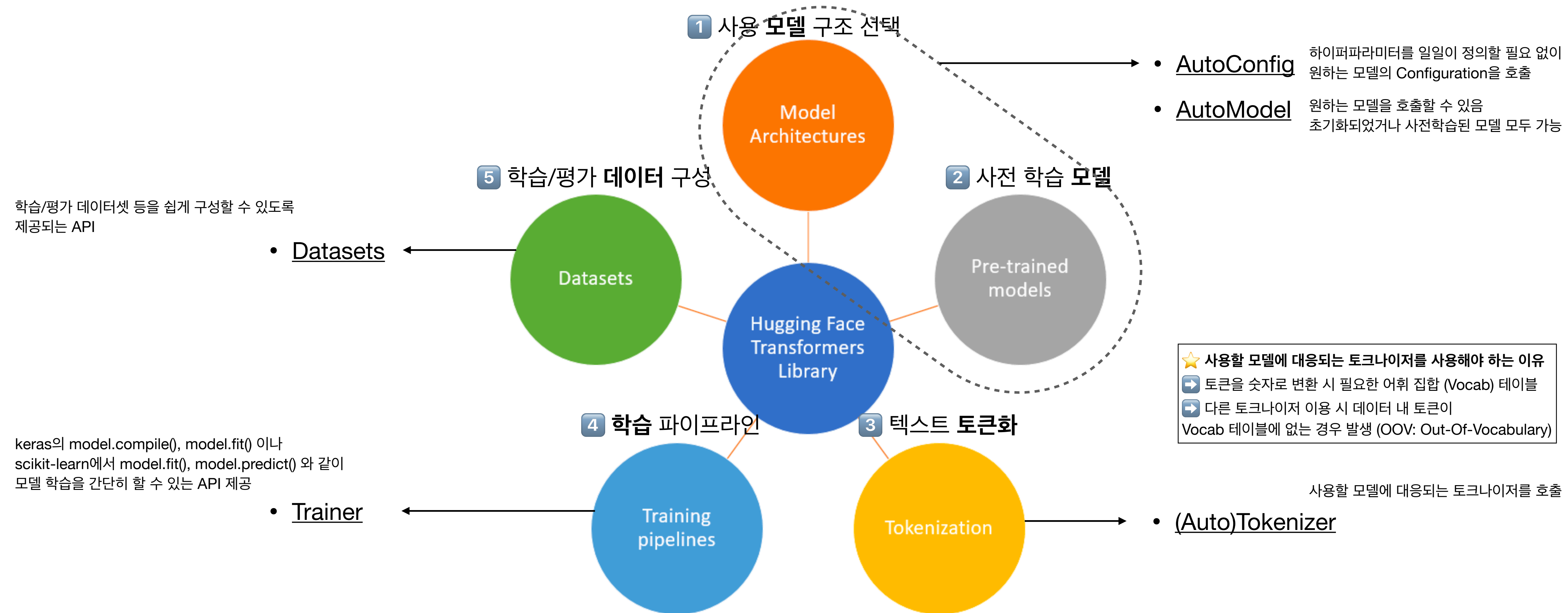


우리 목적에 맞게 미세조정 (*Fine-tuning*) 하는 과정
→ LLM을 문제 해결에 빠르게 적용해볼 수 있음!



Huggingface의 API

transformers 라이브러리의 주요 기능



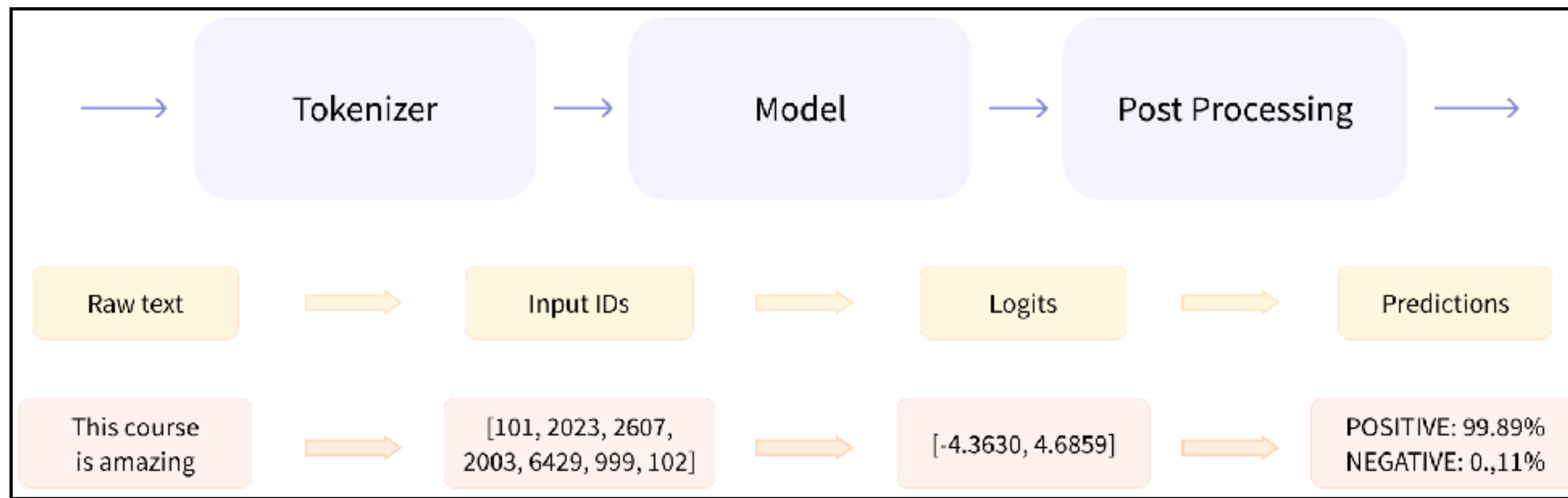
Pipeline 모듈을 활용한 실습 1

감정 분석 (Sentiment Analysis)

주어진 텍스트가 긍정 (Positive) 인지 부정 (Negative) 인지 분류하는 문제

감정 분석 수행 과정

1. 토큰화 (Tokenization)
입력 텍스트 데이터를 모델이 이해할 수 있는 숫자로 변환하는 과정
기존) 토큰라이저를 따로 학습하거나 가져와야 함
2. 모델 (Model)
입력값으로부터 레이블에 대한 (Normalize 되지 않은) 점수 계산
기존) 감정 분석 데이터셋을 확보하여 모델을 학습시켜야 함
logits
3. 후처리 (Post-processing)
Softmax 함수를 적용하여 확률 계산



감정 분석 수행 과정

★ Pipeline 모듈의 장점

자동으로 학습된 토큰라이저와 모델을 호출하여 전/후 처리가 포함된 작업을 간단히 수행할 수 있다

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier([
    "I've been waiting for a HuggingFace course my whole life.",
    "I hate this so much!",
])

## Output

# [{'label': 'POSITIVE', 'score': 0.9598047137260437},
# {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

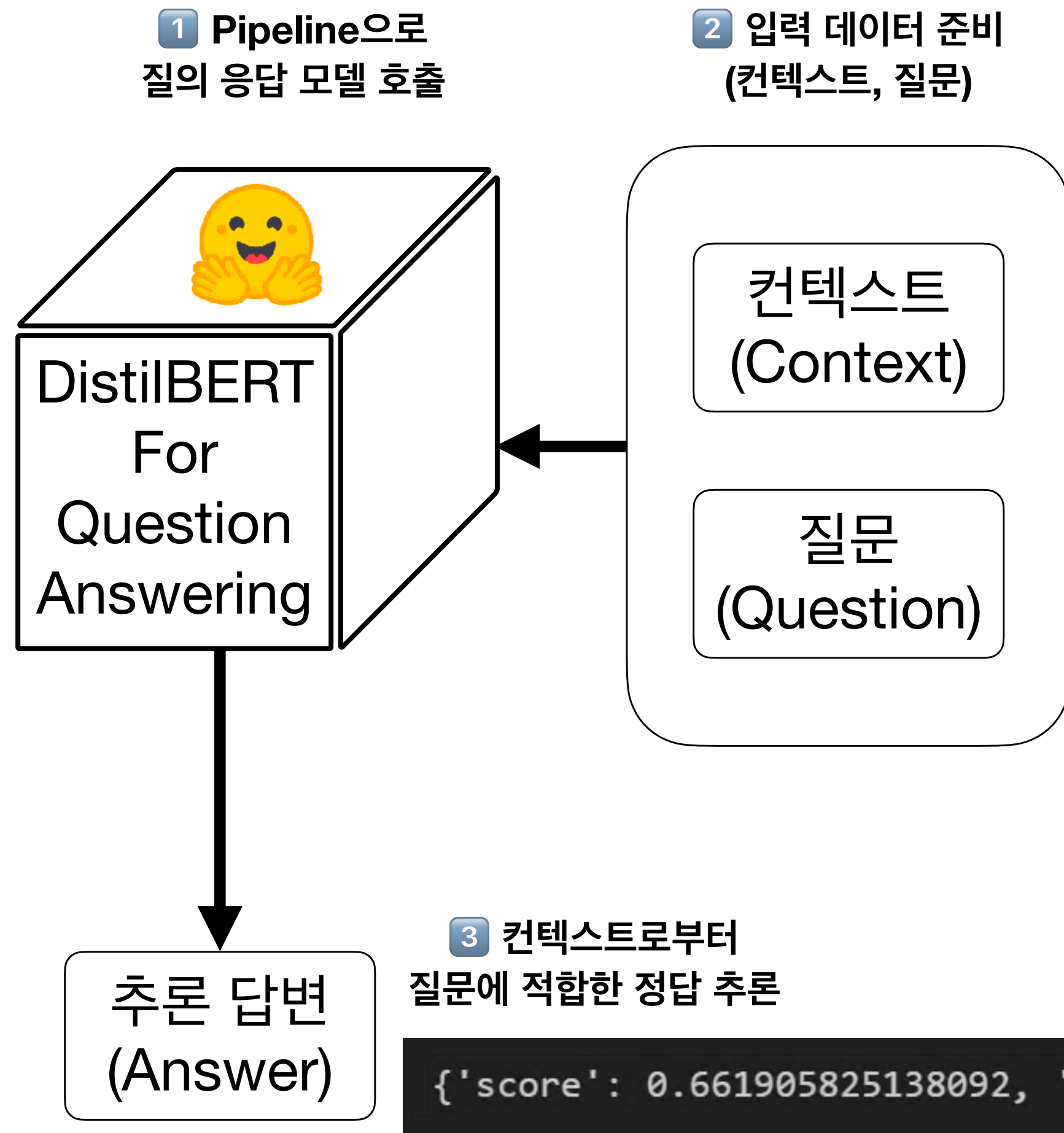
huggingface의 pipeline 함수 예시

Pipeline 모듈을 활용한 실습 2

질의 응답 (Question Answering)

컨텍스트와 질문 (Context, Question) 이 주어졌을 때, 정답 (Answer) 을 찾는 작업

위키피디아의 2020 하계 올림픽 내용 (컨텍스트) 을 주고 해당 내용 기반 질문 하기



```
from transformers import pipeline

qa = pipeline("question-answering")

olympic_wiki_text = """
The 2020 Summer Olympics (Japanese: 2020年夏季オリンピック, Hepburn: Nisen Nijū-nen Kaki Orinpikku),
officially the Games of the XXXII Olympiad (第三十二回オリンピック競技大会, Dai Sanjūni-kai Orinpikku Kyōgi
Taikai) and branded as Tokyo 2020 (東京2020), is an ongoing international multi-sport event being held
from 23 July to 8 August 2021 in Tokyo, Japan, with some preliminary events that began on 21 July.

(... 중략 ...)

Bermuda, the Philippines, and Qatar won their first-ever Olympic gold medals.[8][9][10] San Marino,
Turkmenistan, and Burkina Faso won their first-ever Olympic medals.[11][12][13]

"""

print(qa(
    question="What caused Tokyo Olympic postponed?",
    context=olympic_wiki_text
))
```

답변: COVID-19 팬데믹

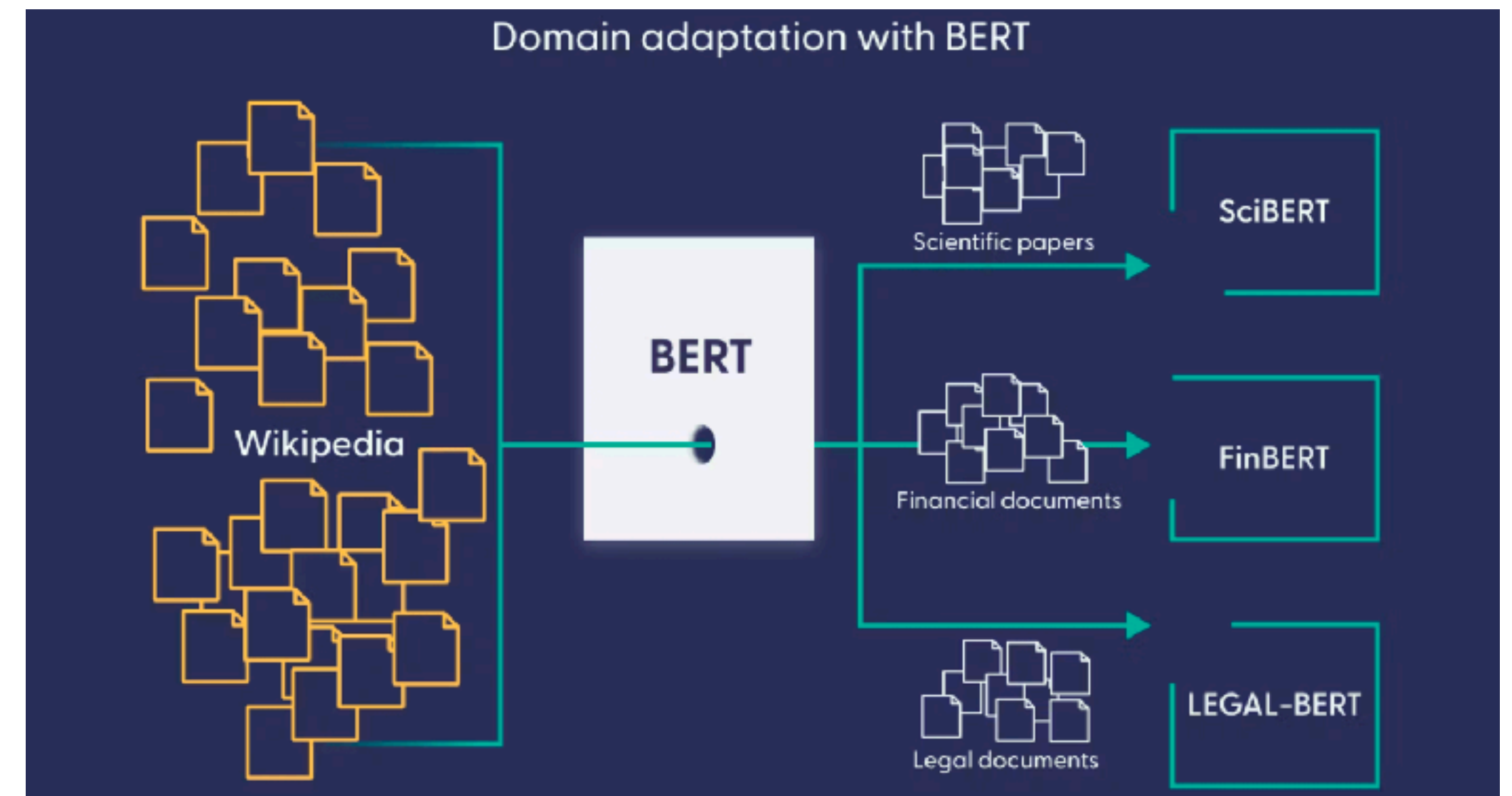
```
{'score': 0.661905825138092, 'start': 635, 'end': 652, 'answer': 'COVID-19 pandemic'}
```

사전학습 모델 미세 조정 (Fine-tuning)

0 왜 Fine-tuning이 필요한가

예) 질의 응답 (Question Answering) 작업을 수행해야 한다.

- 목적: “우리가 가진 법적 계약서 및 학술 논문 (**Context**) 으로부터 질문에 대한 정답을 잘 찾는 모델을 만들고 싶다”
- 만약 사전학습된 BERT 모델을 곧바로 사용한다면?
 - ➔ Context에 포함된 특수 법적 용어, 학술 용어는 희귀한 **Token**으로 간주되고 정답 추론 성능 하락
 - ➔ 특정 분야의 단어 (Domain-specific word) 에 대한 정보가 일반 BERT 모델에는 없다.
- 도메인 데이터를 이용하여 사전학습 언어모델을 미세조정함으로써, 우리가 풀 문제 (Downstream Task) 에 대한 성능을 향상
 - ➔ 이러한 작업을 **Domain Adaptation**이라 한다.

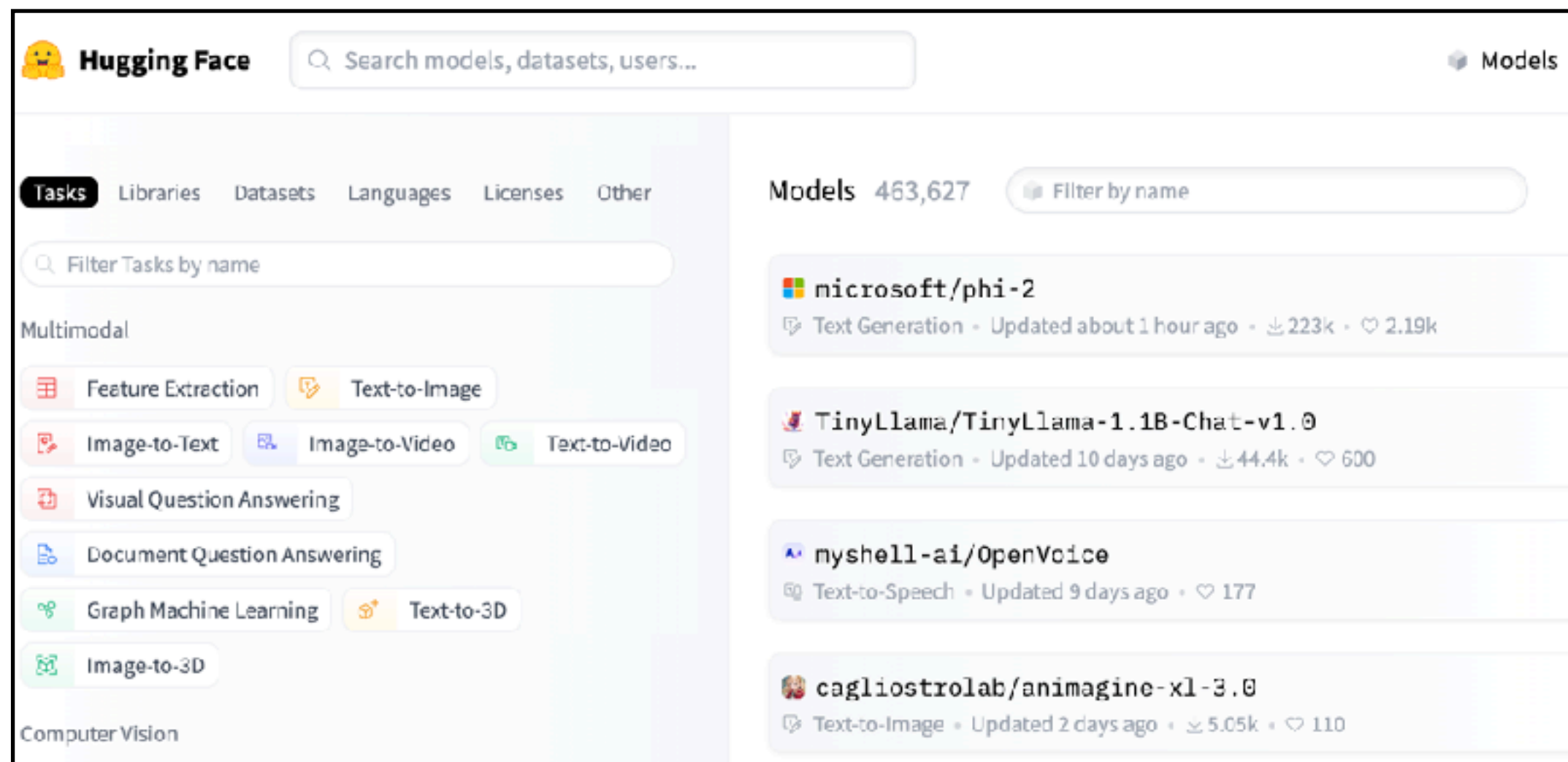


사전학습 모델 미세 조정 (Fine-tuning)

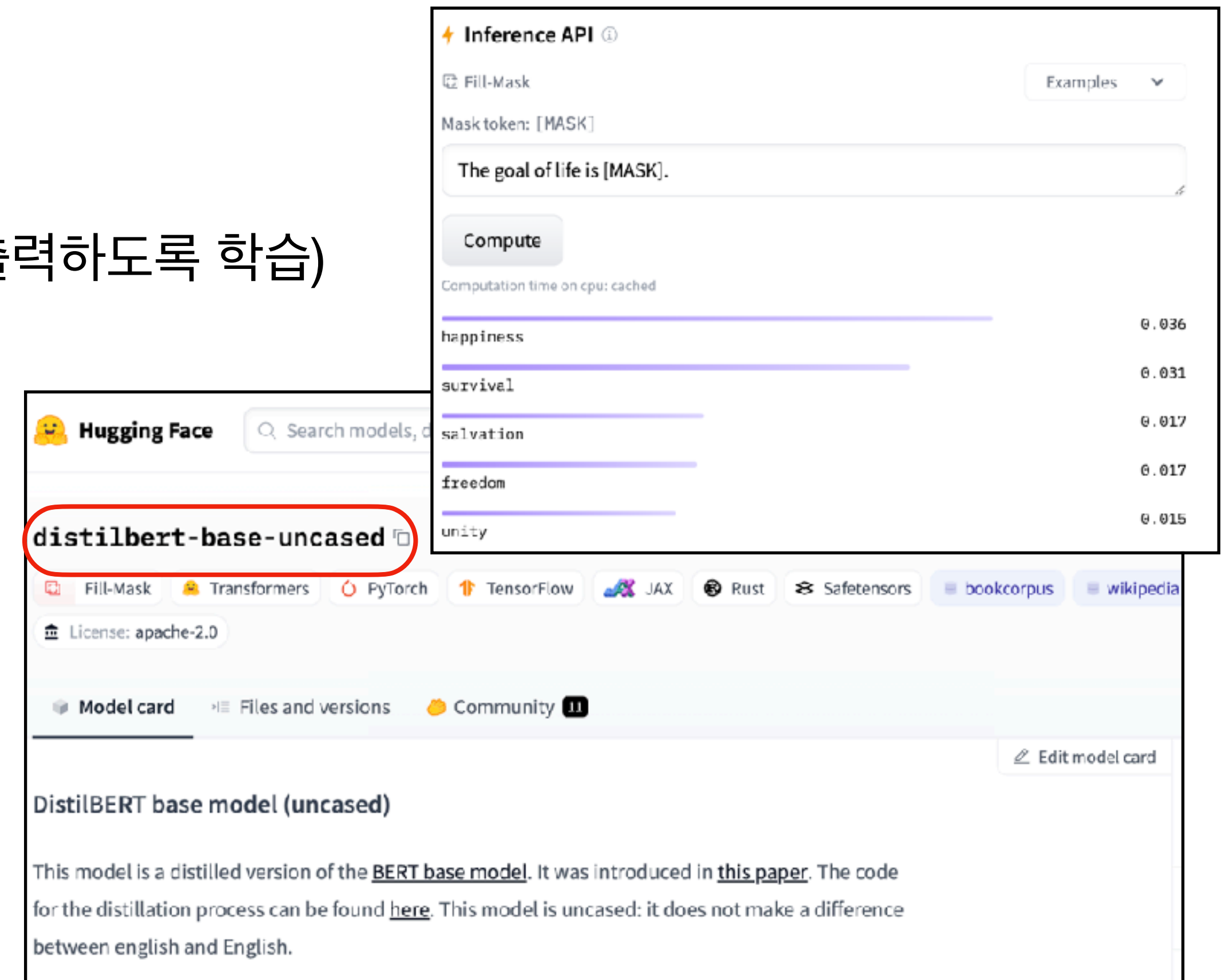
1 Huggingface Hub에서 미세 조정할 사전학습 모델 선정

어떤 목적을 위한 모델을 만들고자 하는가?

- 시퀀스 (문장) 의 의미를 함축하는 모델
 - ➔ BERT 계열, Fill-Mask Task를 수행하는 모델
 - ➔ **DistilBERT** (BERT 대비 작은 사이즈이며, BERT 같이 출력하도록 학습)



Huggingface Hub



DistilBERT 및 Fill-Mask 작업

사전학습 모델 미세 조정 (Fine-tuning)

2 선정된 사전학습 모델 및 토큰라이저 로드

DistilBERT (BERT 대비 작은 사이즈이며, BERT 같이 출력하도록 학습)

- *model*: 사전학습된 DistilBERT 모델 (**distilbert-base-uncased**)
- *tokenizer*: 사전학습 모델 (**model**) 학습에 활용된 텍스트 데이터 토큰화에 사용된 토큰라이저

```
from transformers import AutoModelForMaskedLM, AutoTokenizer

model_checkpoint = "distilbert-base-uncased"

model =
AutoModelForMaskedLM.from_pretrained(model_checkpoint)
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

사전학습 모델 및 토큰라이저 불러오기

사전학습 모델 미세 조정 (Fine-tuning)

3 데이터 준비

IMDb (Large Movie Review Dataset, 영화 리뷰 데이터셋)

- 감정 분석 모델의 벤치마크 데이터셋으로 활용
- “위키의 팩트 데이터로 학습된 BERT 모델에 IMDb 데이터로 미세조정하여, 리뷰의 주관적 요소를 이해할 것으로 기대”

```
from datasets import load_dataset

imdb_dataset = load_dataset("imdb")


sample = imdb_dataset["train"].shuffle(seed=42).select(range(3))

for row in sample:
    print(f"\n>>> Review: {row['text']}")
    print(f">>> Label: {row['label']}")

'>>> Review: This is your typical Priyadarshan movie... and Sharman Joshi is the best.'
'>>> Label: 0'

'>>> Review: Okay, the story makes no sense, ... Skip this mess.'
'>>> Label: 0'

'>>> Review: I saw this movie at the theaters when I was about 6 or 7 years old. ... Hope this helps.'
'>>> Label: 1'
```



IMDb 데이터셋 로드 및 샘플 데이터

사전학습 모델 미세 조정 (Fine-tuning)

4 데이터 전처리

1. 전체 텍스트에 대해 텍스트 토큰화 수행 (Tokenization)
2. 사전 학습 모델의 입력값 최대 길이 제한 보다 작은 길이를 설정 (**chunk_size**, BERT의 경우 512이므로 해당 값 미만)
3. 토큰화된 데이터를 **chunk_size** 단위로 분할하여 학습/평가 데이터 생성

```
def tokenize_function(examples):  
    result = tokenizer(examples["text"])  
    if tokenizer.is_fast:  
        result["word_ids"] = [result.word_ids(i) for i in  
range(len(result["input_ids"]))]  
    return result  
  
# Use batched=True to activate fast multithreading!  
tokenized_datasets = imdb_dataset.map(  
    tokenize_function, batched=True, remove_columns=["text", "label"]  
)  
  
tokenizer.model_max_length # 512
```

데이터 전처리 일부 과정 (1. 전체 텍스트 토큰화)

사전학습 모델 미세 조정 (Fine-tuning)

5 Hyperparameter 설정

TrainingArguments

- *batch_size* (모델이 한번 가중치를 업데이트할 때 사용되는 학습 샘플 수)
- *learning_rate*, *weight_decay* (모델 학습 시 가중치를 얼마나 많이, 어떻게 변화시킬 것인지)
- *logging_steps* (얼마나 자주 학습 오차를 확인하고 기록할 것인지)
- *fp16=True* (학습 가중치를 32bit이 아닌 16bit로 표현하여 약간의 오차를 감수하되, 연산 속도를 향상시킴)
➔ **Mixed-precision training**이라 함

```
from transformers import TrainingArguments

batch_size = 64
# Show the training loss with every epoch
logging_steps = len(downsampled_dataset["train"]) // batch_size

model_name = model_checkpoint.split("/")[-1]
# distilbert-base-uncased

training_args = TrainingArguments(
    output_dir=f"{model_name}-finetuned-imdb",
    overwrite_output_dir=True,
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    weight_decay=0.01,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    push_to_hub=True,
    fp16=True,
    logging_steps=logging_steps,
)
```

TrainingArguments를 활용한 Hyperparameter 설정

사전학습 모델 미세 조정 (Fine-tuning)

6 Trainer API를 활용한 미세 조정 및 결과 비교

```
from transformers import Trainer

trainer = Trainer(
    model=model, args=training_args,
    train_dataset=downsampled_dataset["train"],
    eval_dataset=downsampled_dataset["test"],
    data_collator=data_collator,
    tokenizer=tokenizer,
)

trainer.train()
```

모델 미세조정을 위한 trainer 선언 및 학습 수행

```
from transformers import pipeline

text = "This is a great [MASK]."
```

mask_filler = pipeline(
 model="distilbert-base-uncased-finetuned-
 }mdb" 미세조정된 모델의 경로
)

```
preds = mask_filler(text)

for pred in preds:
    print(f">>> {pred['sequence']}")
```

Before Fine-tuning 미세조정 전 마스킹 예측 (Top-5)

```
'>>> This is a great deal.'
'>>> This is a great success.'
'>>> This is a great adventure.'
'>>> This is a great idea.'
'>>> This is a great feat.'
```

After Fine-tuning 미세조정 후 마스킹 예측 (Top-5)

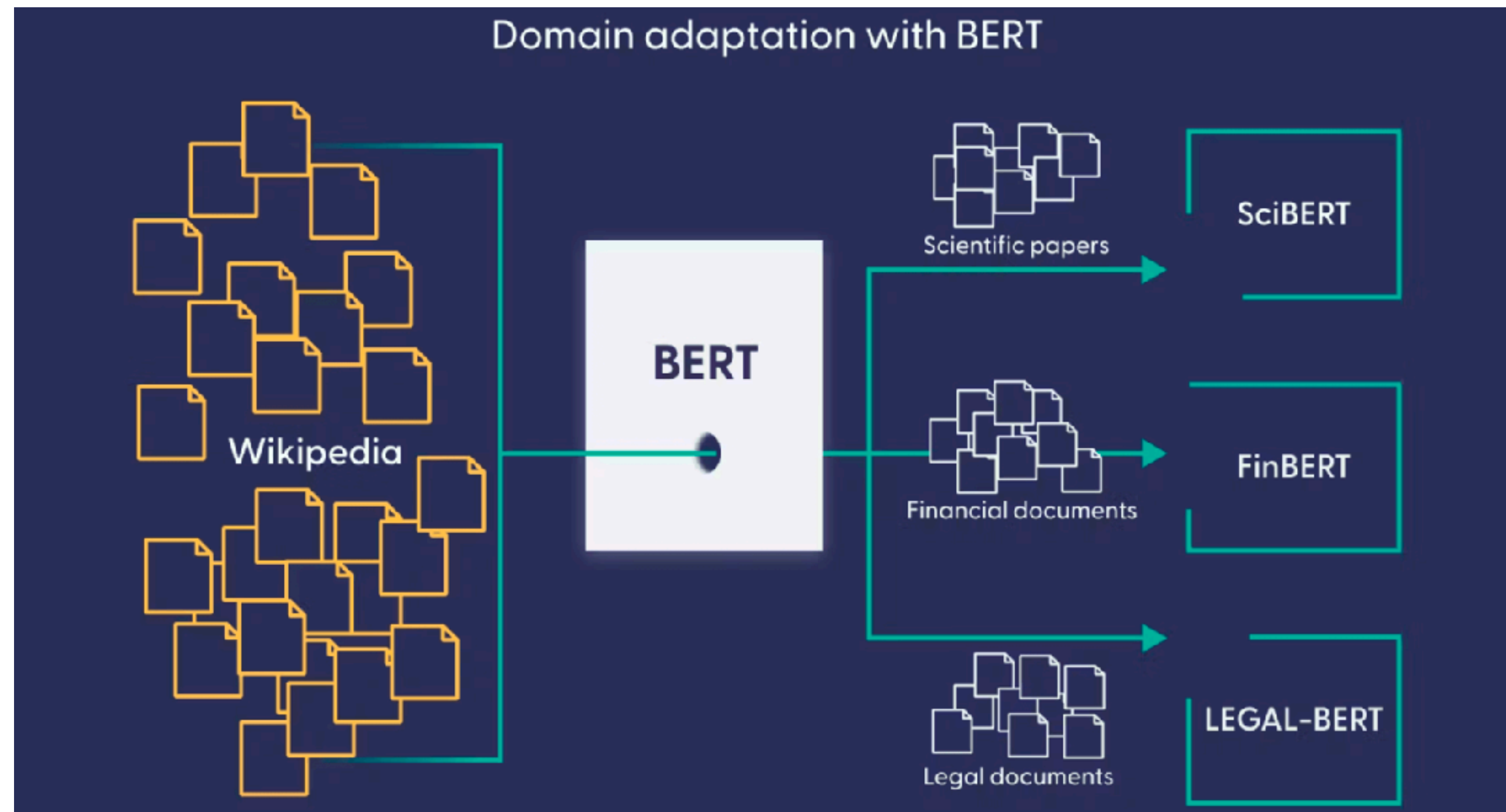
```
'>>> this is a great movie.'
'>>> this is a great film.'
'>>> this is a great story.'
'>>> this is a great movies.'
'>>> this is a great character.'
```

미세조정된 모델을 활용한, 문장 마스킹 대체 작업 (Fill-Mask)

사전학습 모델 미세 조정 (Fine-tuning)

7 정리

- 위키피디아 및 책 데이터로 학습된 일반 BERT 모델을 특정 도메인 (영화) 에 적합한 모델로 활용하기 위해 미세 조정을 수행 (일반 도메인에서 영화 도메인으로의 **Domain Adaptation**)



의견

활용적 측면

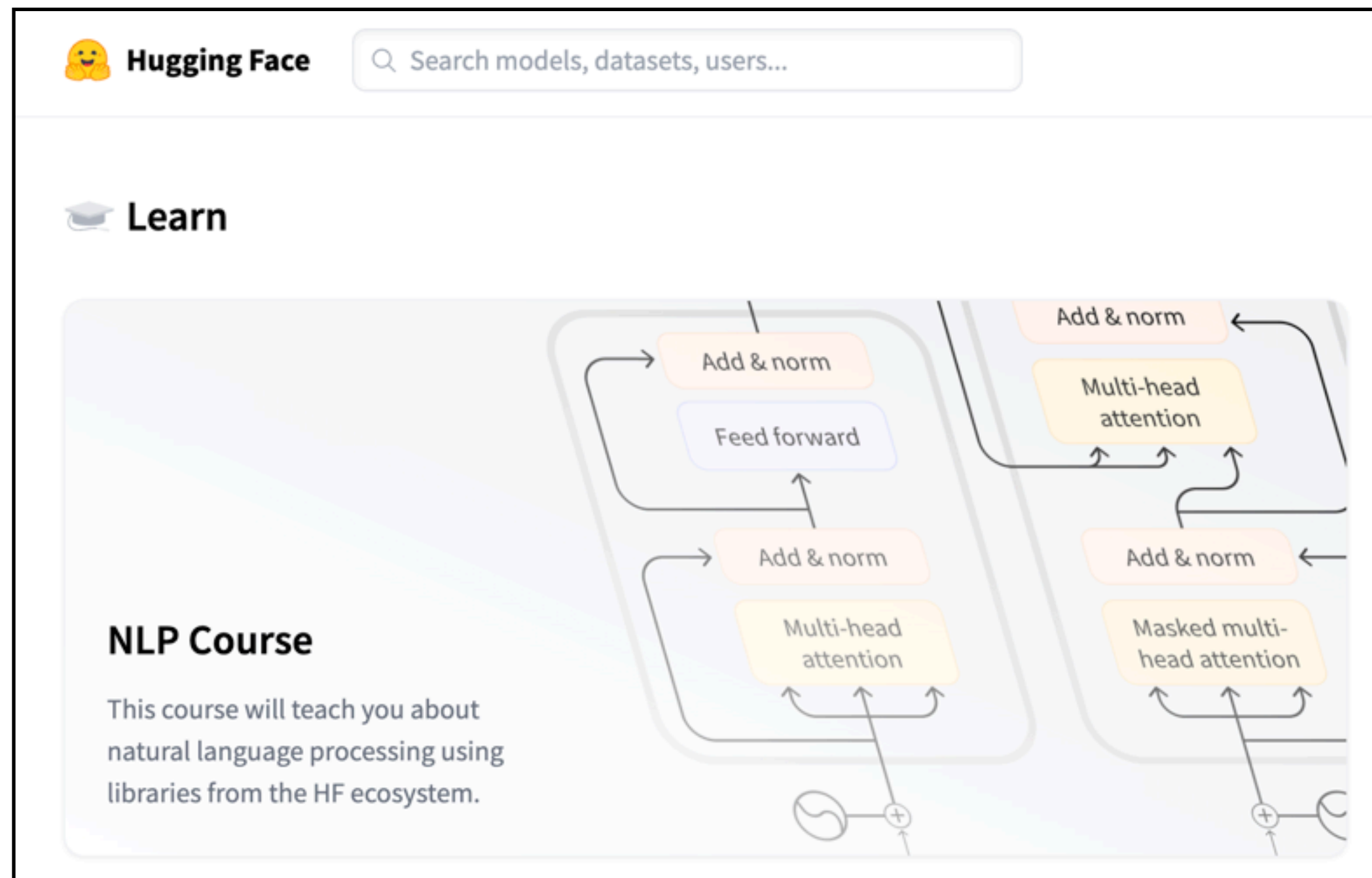
- BERT, GPT 등 대표적인 언어모델 뿐만 아니라 Llama2 등 초거대언어모델 (LLM) 또한 활용 가능
- LLM을 기업 및 특정 도메인에 맞게 활용하는 방안 또한 Huggingface에서 제공
 1. PEFT (Parameter-efficient Fine-Tuning, 학습 파라미터 수를 줄인 효율적 미세 조정 방식)
 2. RAG (Retrieval-augmented Generation, 외부 정보/지식에 접근하여 텍스트를 생성하는 방식)

➡ 언어모델 관련 작업을 할 때 필수적으로 활용되는 라이브러리인만큼, 기본적인 이해와 활용 방법을 익히는 것을 추천👍

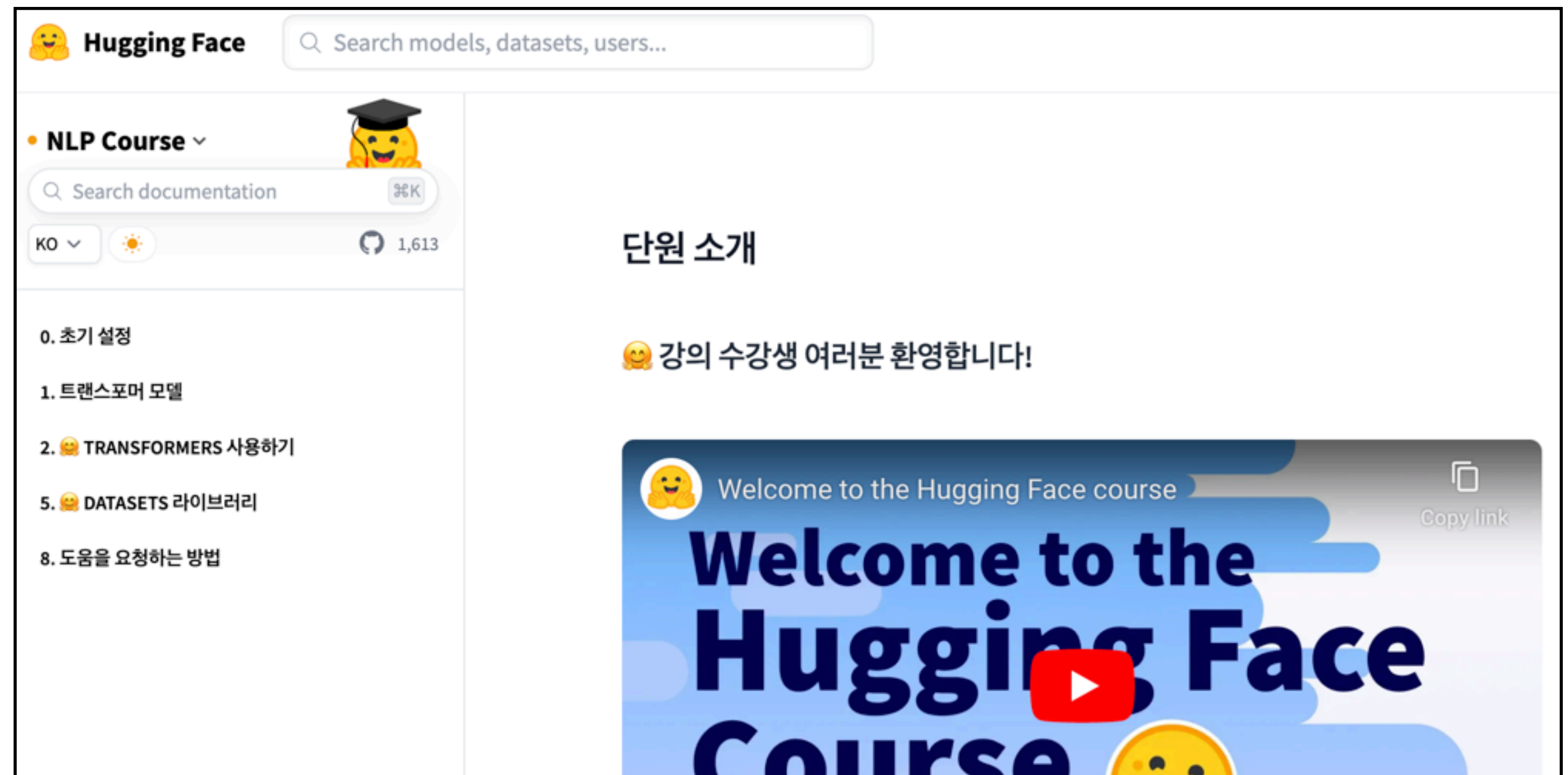
의견

학습 추천

Huggingface에서 제공하는 NLP Course를 통해서 라이브러리 사용법을 익히는 것을 추천 (한글화 작업중)



Huggingface Learn



NLP Course 한글 페이지