딥러닝/클라우드

Chapter 18 **Hugging face**

> 오세종 MIT DANKOOK UNIVERSITY

Contents

- 1. Hugging Face
- 2. Transformer 개요
- 3. Pipeline
- 4. Hub 모델의 활용(1)
- 5. Hub 모델의 활용(2)

[실습] Simple Chatting App

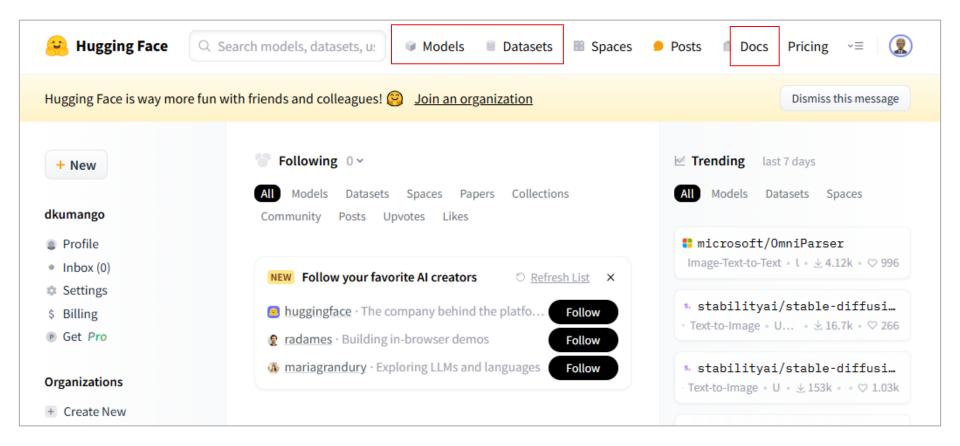
Version info

transformers: 4.46.2

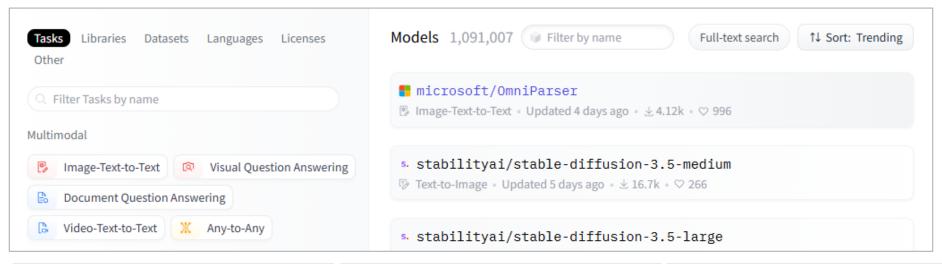
tokenizers: 0.20.3

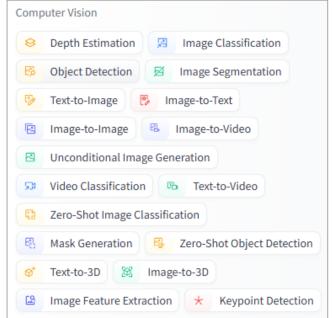
Ref. https://huggingface.co/docs/transformers

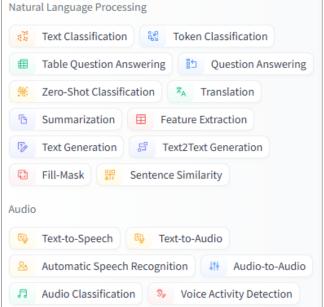
- Hugging face
 - https://huggingface.co/ (link)
 - 기계 학습 모델을 구축, 배포 및 교육하기 위한 도구와 리소스를 개발하는 프 랑스계 미국 회사이자 오픈 소스 커뮤니티
 - 트랜스포머나 데이터셋 같은 AI 프레임워크를 제공하는 세계 최대의 인공지
 능 플랫폼 중 하나
 - 개발자는 여기에 공개된 머신러닝 레퍼런스를 통해 최신 모델을 스스로 구축 할 수 있음
 - 트랜스포머 모델을 만들 때 코드를 작성할 필요없이 여기서 원하는 모델을 다운로드하여 사용할 수 있기 때문에 앱 구축 과정에서 상당히 유용
 - 자연어 처리, 이미지 생성모델, 컴퓨터 비전모델 등 다양한 모델들과 라이브 러리를 제공

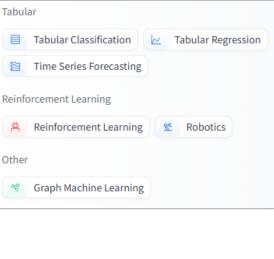


Models









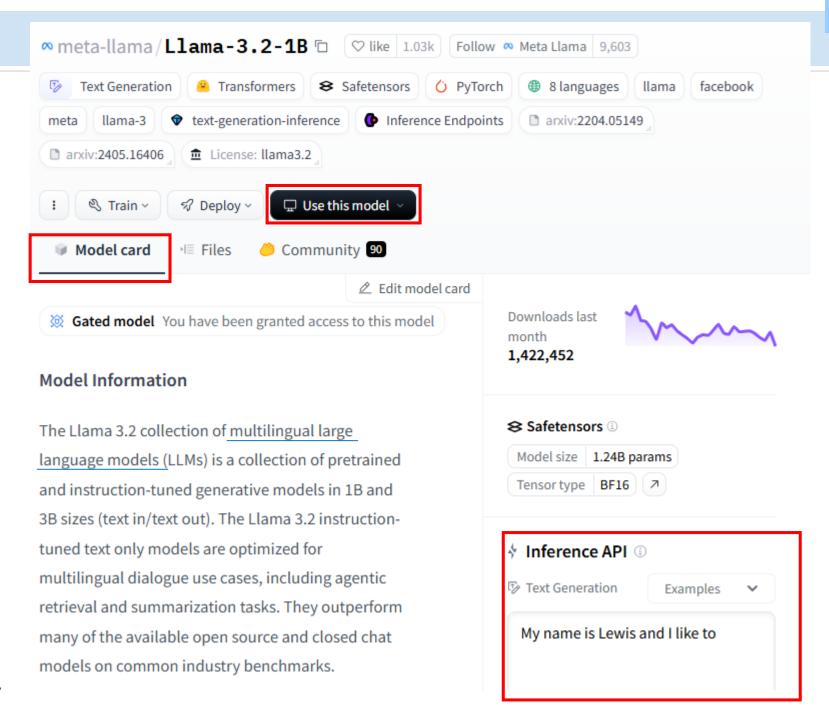
매개변수 수

```
meta-llama/Llama-3.1 8B-Instruct

Text Generation • Updated Sep 26 • ± 6.7M • ♦ • ♥ 3.11k
```

모델 개발업체/개발자 모델명 (checkpoint 라는 용어가 더 많이 사용)

- Model card
 - 각 모델에 대한 설명 제공
 - 성능 평가 결과 제공
 - 모델 다운로드 방법 및 간단한 example 제공
 - o Online playground 제공
 - Huggingface key 가 있거나 개발자 승인이 있어야 다운로드 가능한 모델도 있음
 - Ex) https://huggingface.co/meta-llama/Llama-3.2-1B

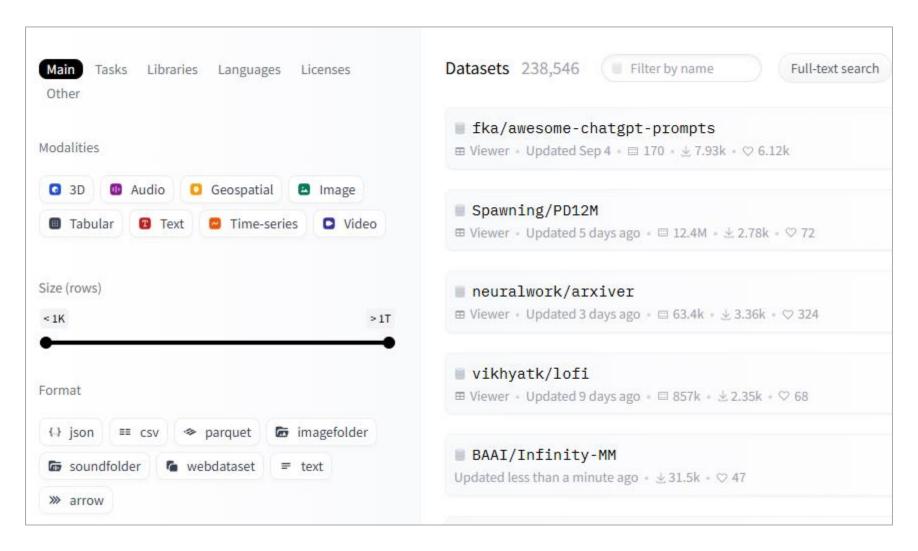


Model Developer: Meta

Model Architecture: Llama 3.2 is an auto-regressive language model that uses an optimized transformer architecture. The tuned versions use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align with human preferences for helpfulness and safety.

	Training Data	Params	Input modalities	Output modalities	Context Length	GQA	Shared Embeddings	
Llama 3.2 (text only)	A new mix of	1B (1.23B)	Multilingual Text	Multilingual Text and	128k	Yes	Yes	
	publicly available online data.			code			the maxim	um number of tokens
		3B (3.21B)	Multilingual Text	Multilingual Text and code			(words or subwords) that the mode can process and consider at once when generating text	

datasets



 https://huggingface.co/datasets/HuggingFaceTB/smolsmoltalk/viewer/default/train?row=0

[{ "content": "I need you to edit something for me. This is the text I wrote, \n\n\"Me and my friends have been waiting for a long time to go back to the movies and catch a movie we been waiting on. Last nite I finely went to go buy tickets and when I got to the movie theater the tickets were sold out, so I was pretty pist. We were all pist. So today we deside if we wanted to go to a game instead, but I dont think we will go to a game now.\"", "role": "user" }, { "content": "Here's a revised version of your text with some suggested edits to improve grammar, clarity, and overall flow:\n\n\"My friends and I have been waiting a long time to return to the movies and see a film we've been eagerly anticipating. Last night, I finally went to buy tickets, but when I arrived at the theater, they were sold out. I was pretty upset, and my friends were too. Today, we discussed the possibility of going to a game instead, but I don't think we'll end up going.\"\n\nI made the following changes:\n\n- Changed \"Me and my friends\" to \"My friends and I\" to correct subject-verb order.\n- Changed \"catch a movie we been waiting on\" to \"see a film we've been eagerly anticipating\" to use more precise and polished language.\n- Changed \"Last nite\" to \"Last night\" to correct spelling.\n- Changed \"finely\" to \"finallv\" to correct spelling and usage.\n- Changed \"pist\" to \"upset\" to use a more common and suitable word for the context.\n- Changed \"deside\" to \"discussed\" to correct spelling and usage.\n- Changed \"dont think we will go\" to \"don't think we'll end up going\" to make the sentence more concise and natural-sounding.\n\nThese changes aim to improve the clarity and coherence of your text while maintaining your original message and tone.", "role": "assistant" }, { "content": "What is the meaning of word 'finely' in English language?", "role": "user" }, { "content": "In the English language, \"finely\" is an adverb that has several meanings. Here are a few possible interpretations:\n\n1. **In a delicate or precise manner**: \"Finely\" can describe an action or process that is done with great care, subtlety, or attention to detail. For example: \"The chef chopped the herbs finely to release their flavors.\"\n\n2. **To a fine or smooth texture**: \"Finely\" can also describe the texture or consistency of something, often implying that it is smooth, powdery, or free from coarseness. For example: \"The sand on the beach was finely ground, making it perfect for building sandcastles.\"\n\n3. **With refinement or elegance**: In some cases, \"finely\" can convey a sense of refinement, sophistication, or elegance. For example: \"The finely crafted furniture

smol-magpie-ultra-short

- 트랜스포머 기반의 다양한 모델을 <u>파이토치, 텐서플로우</u>로 각각 구현해놓은 모듈. 또한 각 모델에 맞는 tokenizer도 구현
- Hugging dace 의 Transformers는 사전 학습된 최첨단 모델들을 쉽게 다운로 드하고 훈련시킬 수 있는 API와 도구를 제공
- 사전학습된 모델을 쓰면 컴퓨팅 비용과 탄소 배출량이 줄고, 모델을 처음부터 훈련시키는 데 필요한 시간과 리소스를 절약
- 다양한 분야의 태스크를 지원
- Pipeline 을 통한 고수준 추상화 제공
 - ☑ 자연어 처리: 텍스트 분류, 개체명 인식, 질의응답, 언어 모델링, 요약, 번역, 객관식 질의응답, 텍스트 생성
 - 🔤 컴퓨터 비전: 이미지 분류, 객체 탐지, 객체 분할
 - 🙅 오디오: 자동음성인식, 오디오 분류
 - ৣ 멀티모달: 표 질의응답, 광학 문자 인식 (OCR), 스캔한 문서에서 정보 추출, 비디오 분류, 시각 질의응답
- 150개가 넘는 지원 모델 (link) + Hub 모델

• 지원 작업

태스크	설명	모달리티	파이프라인 ID
텍스트 분류	텍스트에 알맞은 레이블 붙이기	자연어 처리(NLP)	pipeline(task="senti ment-analysis")
텍스트 생성	주어진 문자열 입력과 이 어지는 텍스트 생성하기	자연어 처리(NLP)	pipeline(task="text- generation")
개체명 인식	문자열의 각 토큰마다 알 맞은 레이블 붙이기 (인물, 조직, 장소 등등)	자연어 처리(NLP)	pipeline(task="ner")
질의응답	주어진 문맥과 질문에 따 라 올바른 대답하기	자연어 처리(NLP)	pipeline(task="ques tion-answering")
빈칸 채우기	문자열의 빈칸에 알맞은 토큰 맞추기	자연어 처리(NLP)	pipeline(task="fill-m ask")
요약	텍스트나 문서를 요약하 기	자연어 처리(NLP)	pipeline(task="sum marization")
번역	텍스트를 한 언어에서 다 른 언어로 번역하기	자연어 처리(NLP)	pipeline(task="trans lation")

태스크	설명	모달리티	파이프라인 ID
이미지 분류	이미지에 알맞은 레이블 붙 이기	컴퓨터 비전(CV)	pipeline(task="image-cla ssification")
이미지 분할	이미지의 픽셀마다 레이블 붙이기(시맨틱, 파놉틱 및 인스턴스 분할 포함)	컴퓨터 비전(CV)	pipeline(task="image-se gmentation")
객체 탐지	이미지 속 객체의 경계 상 자를 그리고 클래스를 예측 하기	컴퓨터 비전(CV)	pipeline(task="object-det ection")
오디오 분류	오디오 파일에 알맞은 레이 블 붙이기	오디오	pipeline(task="audio-clas sification")
자동 음성 인식	오디오 파일 속 음성을 텍 스트로 바꾸기	오디오	pipeline(task="automatic -speech-recognition")
시각 질의응답	주어진 이미지와 질문에 대 해 올바르게 대답하기	멀티모달	pipeline(task="vqa")
문서 질의응답	주어진 문서와 질문에 대해 올바르게 대답하기	멀티모달	pipeline(task="document -question-answering")
이미지 캡션 달기	주어진 이미지의 캡션 생성 하기	멀티모달	pipeline(task="image-to-t ext")

- Transformer 설치
 - o CPU 버전

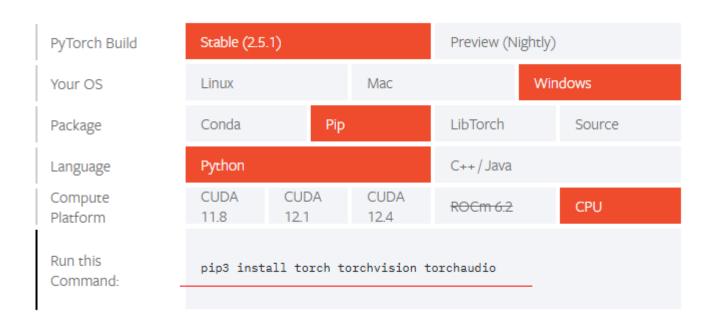
pip install transformers datasets evaluate accelerate

- o GPU 버전
 - PyTorch + cuda 사전 설치 후

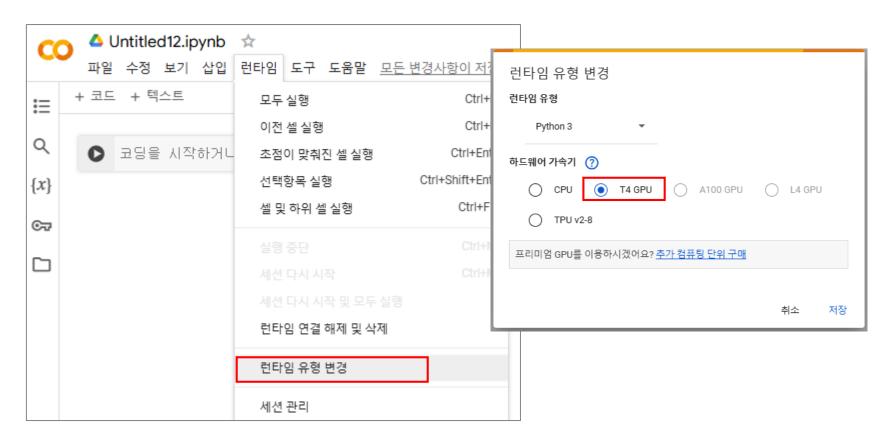
pip install transformers[torch] # PyTorch

② Transformers는 Python 3.6+, PyTorch 1.1.0+, TensorFlow 2.0+ 및 Flax에서 테스트되었습니다.

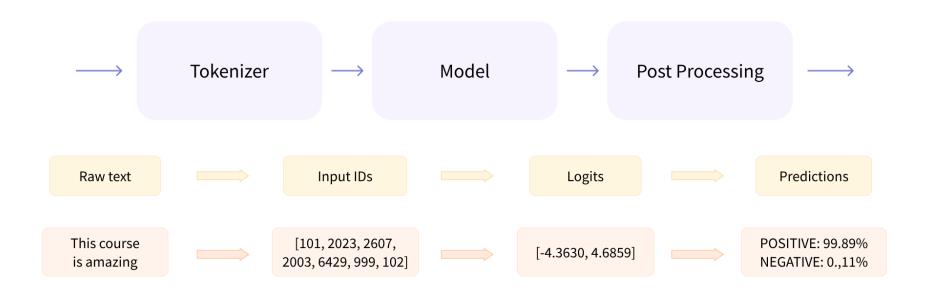
- Pytorch 설치
 - https://pytorch.org/get-started/locally/



- Note
 - LLM 실습은 Giga 단위의 저장공간과 일정 수준 이상의 computing power 를 필요로 하므로 자원 부족시 google colab 의 사용 고려
 - Colab 사용시 GPU 를 활성화 시키는 것이 유리



- Pipeline 이란
 - transformers 라이브러리의 가장 기본 객체
 - 모델 사용을 위한 인터페이스 제공
 - 전처리 + 후처리 과정을 모델과 연결하여 쉽게 NLP 모델을 사용할 수 있게 함.



https://hugging face.co/learn/nlp-course/ko/chapter 2/2



o sentiment-analysis : 감정 분석

• question-answering : 질의 응답

o text-generation : 텍스트 생성

o translation : 번역

o summarization : 요약

○ image-classification: 이미지 분류

o object-detection: 객체 탐지

● Image-to-text: 이미지에 캡션 달기

○ feature-extraction : 특징 추출 (텍스트에 대한 벡터 표현 추출)

o fill-mask : 마스크 채우기

o ner : 개체명 인식 (named entity recognition)

zero-shot-classification : 제로샷 분류

▶ sentiment-analysis (감성분석)

18_transformers.py

```
from transformers import pipeline
                                                      모델을 다운로드
pipe = pipeline('sentiment-analysis',
                   model='nlptown/bert-base-multilingual-uncased-sentiment')
out = pipe('I am very happy')
out
out = pipe('I am very tired')
out
out = pipe('I am very sad')
                                         >>> out = pipe('I am very happy')
out
                                         >>> out
                                         [{'label': '5 stars', 'score': 0.7119459509849548}]
                                         >>> out = pipe('I am very tired')
                                         >>> out
                                         [{'label': '2 stars', 'score': 0.4566650986671448}]
                                         >>> out = pipe('I am very sad')
                                         >>> out
                                         [{'label': '1 star', 'score': 0.4505811035633087}]
```

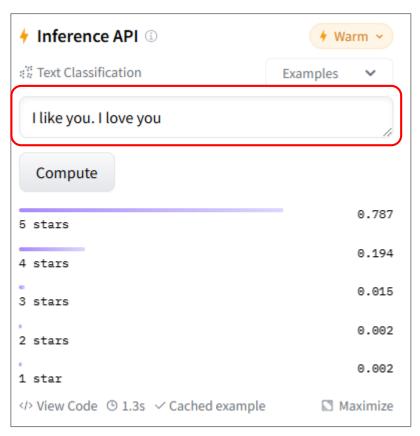
● Model: 감정분석에 사용할 모델을 지정. 미지정시 default model 적용

- nlptown/bert-base-multilingual-uncased-sentiment
 - https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment

Training data

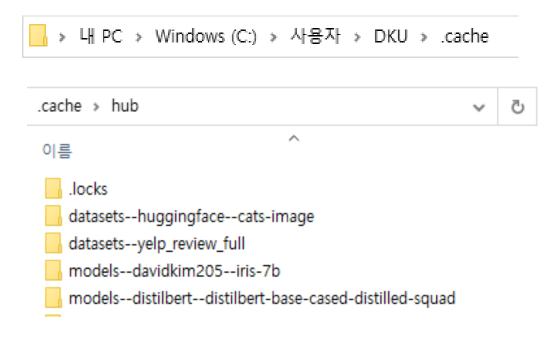
Here is the number of product reviews we used for finetuning the model:

Language	Number of reviews
English	150k
Dutch	80k
German	137k
French	140k
Italian	72k
Spanish	50k

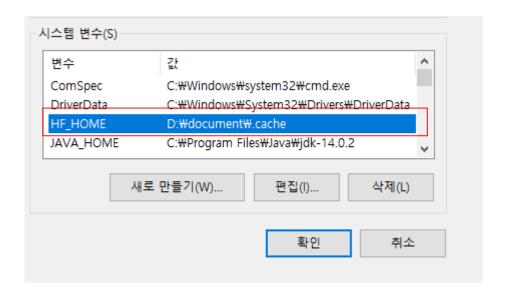


Note

- Pipline 에서 특정 모델을 호출하면 local computer의 .cache 폴더에 다운로드 된다. (최초 호출시 1회만)
- 다음번에 그 모델을 호출하면 .cache 폴더에서 모델을 읽는다.
 (.cache 폴더의 경로 변경 가능)
- Window 에서 .cache 의 위치



- Note
 - o .cache 의 경로를 바꾸고 싶으면 시스템 환경 변수 'HF_HOME' 에 등록



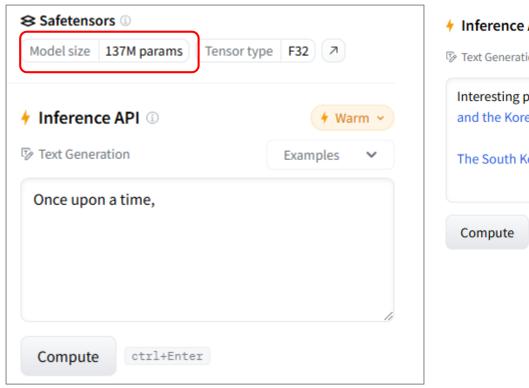
>>>

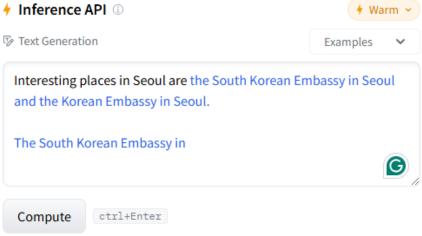
text-generation (문장생성)

```
[{'generated_text': 'Today is a beautiful day and I am feeling this g ood," he said. "And so I have decided to say goodbye to it all and th en let'}, {'generated_text': 'Today is a beautiful day and I am feeling so blessed to be here with you all. I was looking forward to getting this very special gift. When'}, {'generated_text': 'Today is a beautiful day and I am feeling very well today."\n\nThe team in question had two previous losses this season, both at home against'}]

>>> out[2]['generated_text']
'Today is a beautiful day and I am feeling very well today."\n\nThe team in question had two previous losses this season, both at home against'
```

- openai-community--gpt2
 - https://huggingface.co/openai-community/gpt2





text2text-generation

주어진 맥락에서 답을 찾음

```
from transformers import pipeline

pipe = pipeline("text2text-generation")
# default model : google-t5--t5-base

out = pipe("translate from English to French: I'm very happy")
out
out = pipe("question: where is the capital of south korea?")
out
out = pipe("question: who is taller? context: John's height is 170
and Tom's height is 180")
out
```

```
>>> out = pipe("translate from English to French: I'm very happy")
>>> out
[{'generated_text': 'Je suis très heureux'}]
>>> out = pipe("question: where is the capital of south korea?")
>>> out
[{'generated_text': 'Seoul'}]
>>> out = pipe("question: who is taller? context: John's height is 17
0 and Tom's height is 180")
>>> out
[{'generated_text': 'Tom'}]
>>>
```

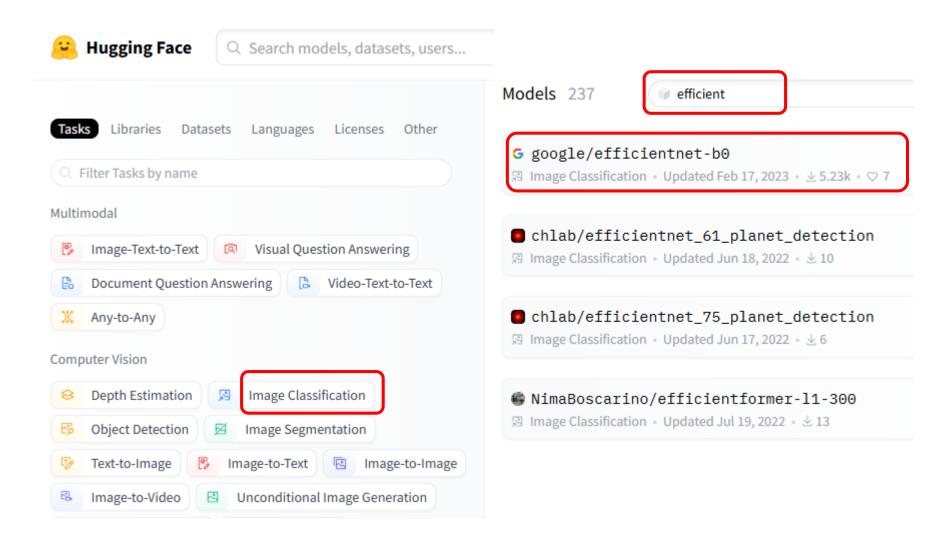
question-answering

```
from transformers import pipeline
pipe = pipeline("question-answering")
# default model :distilbert-base-cased-distilled-squad
ctx = "My name is Ganesh and I am studying Data Science"
que = "What is Ganesh studying?"
out = pipe(context = ctx, question = que)
out
out['answer']
que = "Ganesh lieves in Korea?"
out = pipe(context = ctx, question = que)
out
out['answer']
```

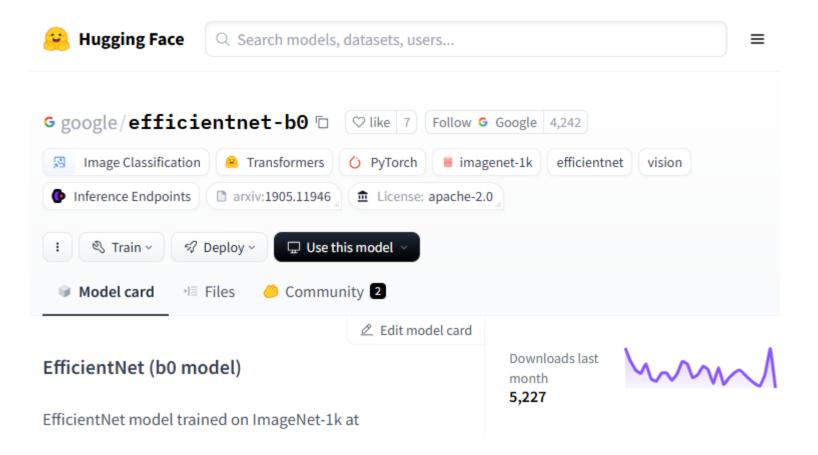
- Type of answer
 - ✓ 추출적(Extractive) 질의 응답: 주어진 문맥에서 답변을 추출
 - ✓ 생성적(Abstractive) 질의 응답: 문맥에서 질문에 올바르게 답하는 답변을 생성

```
>>> que = "What is Ganesh studying?"
>>> out = pipe(context = ctx, question = que)
>>> out
{'score': 0.9988707900047302, 'start': 36, 'end': 48, 'answer': 'Data Science'}
>>> out['answer']
'Data Science'
>>>
>>> que = "Ganesh lieves in Korea?"
>>> out = pipe(context = ctx, question = que)
>>> out
{'score': 0.40489712357521057, 'start': 22, 'end': 48, 'answer': 'I am studying Data Science'}
>>> out['answer']
'I am studying Data Science'
>>> _____
```

• 이미지 분류의 예



https://huggingface.co/google/efficientnet-b0



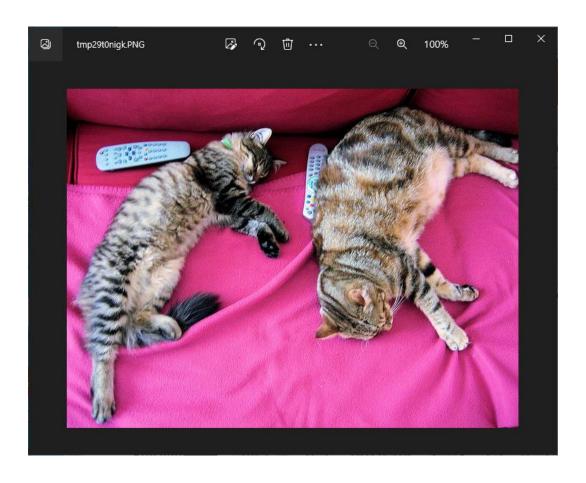
How to use

Here is how to use this model to classify an image of the COCO 2017 dataset into one of the 1,000 ImageNet classes:

```
import torch
from datasets import load_dataset
from transformers import EfficientNetImageProcessor, I
dataset = load_dataset("huggingface/cats-image")
image = dataset["test"]["image"][0]
preprocessor = EfficientNetImageProcessor.from_pretra:
model = EfficientNetForImageClassification.from_pretra
inputs = preprocessor(image, return_tensors="pt")
with torch.no_grad():
   logits = model(**inputs).logits
# model predicts one of the 1000 ImageNet classes
predicted_label = logits.argmax(-1).item()
print(model.config.id2label[predicted_label]),
```

```
import torch
from datasets import load dataset
from transformers import EfficientNetImageProcessor,
EfficientNetForImageClassification
dataset = load_dataset("huggingface/cats-image")
image = dataset["test"]["image"][0]
image.show(image)
preprocessor =
EfficientNetImageProcessor.from_pretrained("google/efficientnet-b0")
model =
EfficientNetForImageClassification.from_pretrained("google/efficientnet-
b0")
```

```
>>> image
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=640x480 at 0x1C4816A3110>
>>>
```



model =
EfficientNetForImageClassification.from_pretrained("google/effic
ientnet-b0")

 모델 최초 사용시 huggingface 에서 모델을 다운로드하여 사용자 컴퓨터의 .cache 폴더에 저장. 이후에는 .cache 폴더에서 모델 로드

```
inputs = preprocessor(image, return_tensors="pt")

# predict
with torch.no_grad():
    logits = model(**inputs).logits

# model predicts one of the 1000 ImageNet classes
predicted_label = logits.argmax(-1).item()
print(model.config.id2label[predicted_label])
```

```
>>> inputs = preprocessor(image, return tensors="pt")
>>> inputs
{'pixel values': tensor([[[[ 0.3138, 0.5193, 0.5707, ..., -0.4739, -0.3712, -0.1486],
         [ 0.5022, 0.3994, 0.5022, ..., -0.2513, -0.3027, -0.5082],
         [ 0.2453, 0.3994, 0.3309, ..., -0.5596, -0.5767, -0.4911],
         [ 1.8550, 1.7180, 1.5297, ..., 1.7352, 1.4440, 1.3927],
         [ 1.8722, 1.8893, 1.7523, ..., 1.4098, 1.2214, 0.9646].
         [ 1.7523, 1.7009, 1.6667, ..., 0.5193, 0.2282, -0.3541]],
        [[-1.5805, -1.5630, -1.4930, ..., -1.7906, -1.7556, -1.7381],
         [-1.4230, -1.6155, -1.4580, ..., -1.7556, -1.7206, -1.6506],
         [-1.7031, -1.6155, -1.6506, ..., -1.8256, -1.8256, -1.9132],
                                                                    1000개 class에 대한 확률값
 >>> logits
 tensor([[-8.3136e-02, -5.1970e-01, -1.2377e+00, 7.1986e-01, -9.0372e-01,
          -1.0230e+00, -1.2709e+00, -1.7049e+00, -1.0651e+00, -5.1778e-01,
          -5.1863e-01, -8.5410e-01, 2.7605e-01, -1.5336e+00, 3.6058e-01,
          -6.8823e-01, -1.0736e+00, -1.1702e+00, -7.7999e-01, -1.6102e+00,
          -1.4702e+00. -5.1136e-01, -1.7086e+00, -3.4374e-01, 2.0212e-01,
          -8.7222e-02. 6.8773e-01. -1.4984e-01. -4.1708e-01. -1.2360e+00.
 >>> predicted label = logits.argmax(-1).item()
 >>> predicted label
 281
  >>> print(model.config.id2label[predicted label]),
 tabby, tabby cat
  (None,)
```

- Chatting test
 - Model: HuggingFaceTB/SmolLM2-360M-Instruct
 - https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct
 - SmolLM2 is a family of compact language models available in three size: 135M, 360M, and 1.7B parameters.
 - They are capable of solving a wide range of tasks while being lightweight enough to run on-device.
 - The instruct model additionally supports tasks such as text rewriting, sumarization and function calling (for the 1.7B)

- ✓ -instruct 있는 모델: 사용자 명령에 반응하는 특화된 훈련을 통해, 명령에 더 정확하게 답하거나, 특정 작업을 수행하는 데 최적화된 모델
- ✓ -instruct 없는 모델: 일반적인 언어 패턴을 학습하여 모든 종류의 텍스트 생성에 적합한 범용 모델



- 모델이 사용자 지시나 명령(예: 질문, 명령어 등)에 더 잘 반응하도록 훈련하는 과정
- 모델이 명령어와 그에 따른 응답 쌍으로 이루어진 데이터셋을 학습
- 이를 통해 모델은 사용자가 던지는 질문이나 지시를 더 잘 이해하고, 그에 맞는 정확하고 적절한 응답을 생성할 수 있게 됨

예시: "한국의 수도가 어디인가요?" 와 같은 질문을 할 때 Instruct 가 아닌 모델은 일반적인 텍스트 예측 모델로서 질문을 단순히 텍스트로 이해하고, 답을 제대로 도출하지 못할 수 있음 하지만, Instruct 모델은 이런 질문에 대해 "미국의 수도는 워싱턴 D.C.입니다"와 같은 정확한 답변을 제시

	일반모델	-Instruct 모델
훈련 방식	일반적인 언어 데이터 학습	명령어와 지시에 따른 학습 (Instruction-tuning)
주요 사용 사례	텍스트 생성, 번역, 일반적인 NLP 작업	질문-답변, 명령어 처리, 명확한 작업 지시
응답의 정확성	응답이 다소 모호하거나 일반적일 수 있음	명확하고 지시 중심적인 응답 제공
상호작용의 성격	개방형 질문에 적합	명령과 지시를 따르는 작업에 적합

Chatting test

18.chat_SmolLM_test.py

```
from transformers import AutoModelForCausalLM, AutoTokenizer
checkpoint = "HuggingFaceTB/SmolLM2-360M-Instruct"

device = "cpu"  # "cuda" for GPU usage, "cpu" for CPU usage
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
model = AutoModelForCausalLM.from_pretrained(checkpoint).to(device)
```

○ 언어 모델을 사용할 때는 일반적으로 tokenizer 와 pretrained model 을 호출

```
def get_answer(messages):
    input text=tokenizer.apply chat template(messages,
                                              tokenize=False)
    inputs = tokenizer.encode(input_text,
                              return tensors="pt").to(device)
    outputs = model.generate(inputs,
                            max new tokens=1024,
                            temperature=0.2,
                            top p=0.9,
                            do sample=True)
    tmp = tokenizer.decode(outputs[0])
    tmp = tmp[-(len(tmp)-len(input text)):]
    tmp = tmp.replace("<|im_start|>assistant\n", "")
    tmp = tmp.replace("<|im end|>", "")
    return(tmp)
```

```
>>> input text=tokenizer.apply chat template(messages, tokenize=False)
>>> print(input text)
<|im start|>system
You are a helpful AI assistant named SmolLM, trained by Hugging Face< im end >
</im start|>user
Show me python cod for histogram chart< im end >
>>> inputs = tokenizer.encode(input text, return tensors="pt").to(device)
>>> inputs
tensor([[ 1, 9690, 198, 2683, 359, 253, 5356, 5646, 11173, 3365,
         3511, 308, 34519, 28, 7018, 411, 407, 19712, 8182, 2,
          198, 1, 4093, 198, 21175, 549, 11129, 12123, 327, 31308,
         6473, 2, 198]])
>>>
>>> tmp = tokenizer.decode(outputs[0])
>>> tmp
```

"<|im_start|>system\nYou are a helpful AI assistant named SmolLM, trained by Hugging Face<|im_end|>\n<|im_start|>user \nShow me python cod for histogram chart<|im_end|>\n<|im_start|>assistant\nHere's a simple Python code snippet to cre ate a histogram chart:\n\n``python\nimport matplotlib.pyplot as plt\nimport numpy as np\n\n# Generate random data\nn p.random.seed(0)\ndata = np.random.randint(0, 100, size=1000)\n\n# Create a histogram\nplt.hist(data, bins=20, densit y=True, alpha=0.7, color='blue', edgecolor='black')\n\n# Add title and labels\nplt.title('Histogram of Data')\nplt.xl abel('Value')\nplt.ylabel('Frequency')\n\n# Show the histogram\nplt.show()\n``\n\nThis code uses the `matplotlib` li brary to create a histogram chart. The `bins` parameter determines the number of bins in the histogram, and the `dens ity` parameter makes the histogram density-based, which can be useful for visualizing the data. The `alpha` parameter controls the transparency of the bars, and `color` and `edgecolor` control the color and edge of the bars, respectively.\n\nYou can customize the appearance of the histogram chart by adjusting the parameters of the `plt.hist()` function. For example, you can change the color, transparency, and edge color to suit your needs.kim_end|."

- tokenizer.apply_chat_template()
 - 사용자의 입력 메시지를 사전에 정의된 템플릿에 맞추어 입력 프롬프트 생성

○ 다운로드 받은 모델 파일들중 tokenizer_config.json에 정의됨 (변경 가능)

```
"bos_token": "<\im_start\>",
    "chat_template": "{% for message in messages %}{% if loop.first and messages[0]['role'] != 'system' %}{{ '<\im_start\>system\nYou are a helpful AI assistant named SmolLM, trained by Hugging Face<\im_end\>\n' }}{% endif %}
{{'<\im_start\>' + message['role'] + '\n' + message['content'] + '<\im_end\>' + '\n'}}{% endfor %}{% if
add_generation_prompt %}{{ '<\im_start\>assistant\n' }}{% endif %}",
    "clean_up_tokenization_spaces": false,
    "eos_token": "<\im_end\>",
    "model max length": 2048,
```

- model.generate() 매개변수
 - inputs : 토큰화된 입력 프롬프트
 - max_new_tokens : 생성될 문장의 최대 토큰 수 (클수록 생성 시간이 길어짐)
 - temperature : LLM의 출력 다양성을 제어.
 - 1.0에 가까울 수록 다양한 문장이 생성됨.
 - 0에 가까울수록 더 일관된 응답 생성, 사실적이고 정확한 정보가 필요할 때 유용
 - top_p: 확률이 낮은 단어들을 필터링하여 답변의 품질을 높이는 역할.
 - 클 수록 다양한 문장 생성
 - do_sample : temparature, top_p 의 적용 여부 설정
 - temparature, top_p: LLM 모델이 사용자 요청에 따른 결과를 생성하는 방식과 결과물의 특성을 결정하는 데 중요한 역할

• 다음 슬라이드에 출력 결과

```
>>> print(answer)
Here's a simple Python code snippet to create a histogram chart:
```python
import matplotlib.pyplot as plt
import numpy as np
Generate some data
data = np.random.randint(0, 100, 1000)
Create a histogram
plt.hist(data, bins=20, edgecolor='black', alpha=0.7, label='Histogram')
Add a title and labels
plt.title('Histogram of Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
Add a legend
Show the plot
plt.show()
```

- 한국어로 대화 할 수 있는 App 개발
  - 한국어-영어 변환은 google trans library 이용
    - MIT 라이선스. 상업적으로 사용하는데도 큰 제약이 없고, 소스코드를 공개하지 않 아도 됨
    - 설치: pip install googletrans
    - 참고: <a href="https://boringariel.tistory.com/13">https://boringariel.tistory.com/13</a>
      <a href="https://py-googletrans.readthedocs.io/en/latest/">https://py-googletrans.readthedocs.io/en/latest/</a>
  - 언어 모델: HuggingFaceTB/SmolLM2-360M-Instruct
  - Python GUI : PyQT



# Googletrans: Free and Unlimited Google translate API for Python



Googletrans is a **free** and **unlimited** python library that implemented Google Translate API. This uses the Google Translate Ajax API to make calls to such methods as detect and translate.

#### Features

- · Fast and reliable it uses the same servers that translate.google.com uses
- · Auto language detection
- · Bulk translations
- Customizable service URL
- · Connection pooling (the advantage of using requests.Session)
- HTTP/2 support

### Note on library usage

- . The maximum character limit on a single text is 15k.
- Due to limitations of the web version of google translate, this API does not guarantee that the library would work properly at all times. (so please use this library if you don't care about stability.)
- . If you want to use a stable API, I highly recommend you to use Google's official translate API.
- If you get HTTP 5xx error or errors like #6, it's probably because Google has banned your client IP address.

google trans test

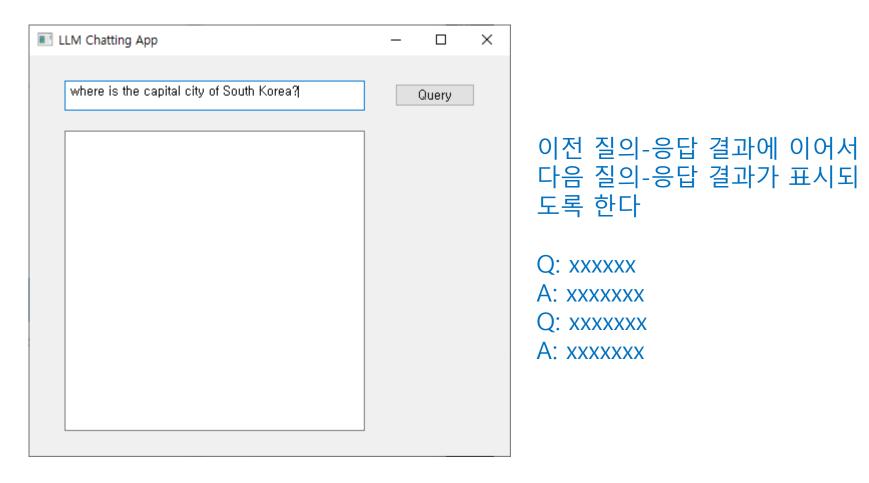
```
from googletrans import Translator translator = Translator()

txt = '''나는 한국어를 영어로 번역하는 것을 좋아합니다.'''
eng = translator.translate(txt, src='ko', dest='en')
eng.text

txt = '''Seoul is the capital of South Korea.'''
kor = translator.translate(txt, src='en', dest='ko')
kor.text
```

```
>>> txt = '''나는 한국어를 영어로 번역하는 것을 좋아합니다.'''
>>> eng = translator.translate(txt, src='ko', dest='en')
>>> eng.text
'I like to translate Korean into English.'
>>>
>>> txt = '''Seoul is the capital of South Korea.'''
>>> kor = translator.translate(txt, src='en', dest='ko')
>>> kor.text
'서울은 한국의 수도입니다.'
```

Chatting GUI



Chatting\_gui\_test.py

```
import sys
from PyQt5.QtWidgets import *
class MyApp(QWidget):
 def __init__(self):
 super().__init__()
 self.initUI()
 # UI 디자인 함수
 def initUI(self):
 self.text_query = QTextEdit()
 self.text_query.setFixedWidth(300) # 텍스트 박스의 폭
 self.text query.setFixedHeight(30) # 텍스트 박스의 높이
 self.btn 1 = QPushButton('Query')
 self.btn 1.setFixedWidth(80)
 self.btn 1.clicked.connect(self.btn 1 clicked)
```

```
self.text answer = QTextEdit()
 self.text answer.setFixedWidth(300)
 self.text answer.setFixedHeight(300)
 gbox = QGridLayout()
 gbox.addWidget(self.text query, 0, 0)
 gbox.addWidget(self.btn 1, 0, 1)
 gbox.addWidget(self.text answer, 1, 0)
 self.setLayout(gbox)
 self.setWindowTitle('LLM Chatting App')
 self.setGeometry(300,300, 480,400)
 self.show()
버튼 클릭시 처리 함수
 def btn 1 clicked(self):
 input = self.text query.toPlainText()
 self.text answer.append("Query:")
 self.text answer.append(input)
```

```
프로그램 실행
if (__name__ == '__main__'):
 app = QApplication(sys.argv)
 ex = MyApp()
 sys.exit(app.exec_())
```

## Note. Transformer.js

- 서버 없이 브라우저에서 직접 Transformers를 실행할 수 있는 웹용 최첨단 머신 러닝 라이브러리임
- Python transformers 라이브러리와 기능적으로 동등하도록 설계되어 동일한 사전 훈련된 모델을 유사한 API로 실행할 수 있음
- 자연어 처리, 컴퓨터 비전, 오디오, 멀티모달 등 다양한 양식의 일반적인 작업을 지원함
- ONNX Runtime을 사용하여 브라우저에서 모델을 실행하며, Optimum을 사용하여 PyTorch, TensorFlow 또는 JAX 사전 훈련 모델을 ONNX로 쉽게 변환할 수 있음
- 참고: <u>https://news.hada.io/topic?id=14299</u>
- https://news.hada.io/topic?id=17427
- https://github.com/huggingface/transformers.js