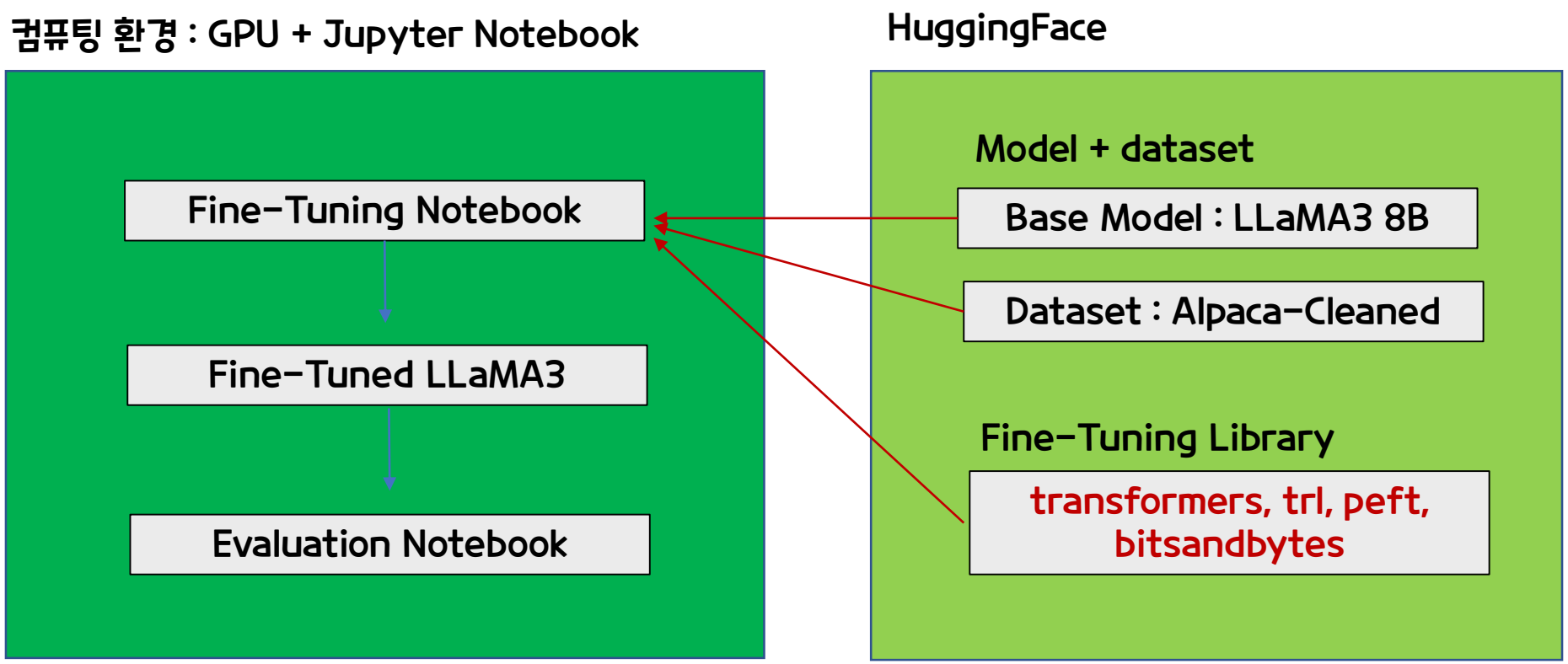
1. LLaMA3-8B Tuning & Model Evaluation

LLaMA3-8B 환경 설정 (1)

- 실습 환경 : Colab + HuggingFace



LLaMA3-8B 환경 설정 (2)

- **HuggingFace**

  : 이 회사는 NLP 및 머신러닝 커뮤니티를 위한 다양한 도구와 플랫폼을 제공함. 주요 플랫폼 및 서비스는 다음과 같다.

  - Hugging Face Model Hub : 다양한 사전 학습된 모델들을 호스팅하고 공유하는 플랫폼
  - Tranformers 라이브러리리 : 트랜스포머 모델을 쉽게 사용할 수 있게 해주는 오픈소스 라이브러리
  - Dataset 라이브러리 : 다양한 데이터셋을 쉽게 다운로드하고 처리할 수 있게 해주는 라이브러리
  - Spaces : AI 모델을 웹 애플리케이션으로 쉽게 배포할 수 있는 플랫폼임. 즉, 머신러닝 모델, 데이터 시각화, 데모 웹 애플리케이션 등을 간편하게 배포하고 공유할 수 있는 공간임. (Streamlit, Gradio, Flask 사용)

- **AI 모델 활용을 위한 다양한 라이브러리 제공**

  - LLM 모델 : Transformers
  - Quantization : BitsAndBytes
  - LoRA(Low-Rank Adaption) : PEFT(Parameter-Efficient Fine Tuning)
    : 모델의 전체 가중치를 업데이트하지 않고 일부 저차원(low-rank) 가중치 행렬만 학습하는 방식
  - TRL (Transformer Reinforcement Learning) : 강화 학습기법을 사용하여 LLM을 Fine-Tuning하는 Framework
    1) RLHF(Reinforcement Learning with Human Feedback
    2) SFT(Supervised Fine-Tuning) 및 DPO(Direct Preference Optimization) 지원

LLaMA3-8B 환경 설정 (3)

- HuggingFace : 환경 설정
  1. Account 만들기 : https://hugginface.co/join
  2. Access Token 만들기 :
     - 우상단 클릭 -> Settings -> 좌측 Access Tokens
     - Create new token -> write 클릭 후 -> token 생성 후 메모해두기
  3. LLaMA3-8B Access 신청하기
     - https://huggingface.co/meta-llama/Meta-Llama-3-8B
     - 해당 페이지 하단에 모델 사용 승인을 위해 제출해야 할 사항들이 있으니 확인해서 (Deploy -> Inference API)
     - submit 하고 나면 금방 메일로 승인 메일을 받을 수 있음(10분 내외)

LLaMA3-8B 환경 설정 (4)

- Google Colab : 환경 설정
  - https://colab.google.com/
  - Google 계정 로그인
  - Colab Pro : Monthly subscription with $9.99를 추천
    - RAM 최대 25GB, 긴 연결 시간
    - 매달 100 computing unit (GPU, TPU, RAM, 연결 시간 등의 사용량 측정) 제공

LLaMA3-8B(Base Model & Evaluation) 실행 (1) : 코드 실습

```
[ ]  1 # 필요 Library install
     2 !pip install transformers==4.39.2 peft==0.10.0 trl==0.8.6 bitsandbytes==0.43.0 accelerate==0.29.0
```

숨겨진 출력 표시

```
[ ]  1 # HF token 설정
     2 from huggingface_hub import login
     3 login(token="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")
```

```
[ ]  1 # 모델 경량화: Quantization 설정
     2 from transformers import BitsAndBytesConfig
     3 import torch
     4
     5 quantization_config=BitsAndBytesConfig(
     6     load_in_4bit=True, # 4비트 양자화 활성화
     7     bnb_4bit_compute_dtype=torch.bfloat16, # 연산 데이터 타입을 bfloat 16으로 설정
     8     bnb_4bit_use_double_quant=True, # double 양자화 사용
     9     bnb_4bit_quant_type='nf4' # 4bit 데이터 타입을 NF4로 설정
    10 )
```

bfloat16(brain)은 float 16보다 소수점 정밀도를 낮지만 표현 범위가 넓어 안정적인 연산가능

```
     1 # 기본 Llama 3 모델 로드
     2 from transformers import AutoModelForCausalLM
     3 model = AutoModelForCausalLM.from_pretrained(
     4     "meta-llama/Meta-Llama-3-8B",
     5     quantization_config = quantization_config,
     6     device_map = {"": 0}  # 모든 레이터를 0번 GPU에 배치
     7 )
```

AutoModelForCausalLM은 자동회귀(auto-regressive)언어 모델을 로드하는 클래스임.

LLaMA3-8B(Base Model & Evaluation) 실행 (2)

```python
# Tokenizer 설정
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("meta-llama/Meta-Llama-3-8B") # 1. LLaMA3 모델의 사전 학습된 토크 나이저 로드
tokenizer.add_special_tokens({"pad_token": "<|reserved_special_token_250|>"}) # 2. 패딩 토큰 추가(LLaMA는 기본적으로 PAD 토큰이 없음)
model.config.pad_token_id = tokenizer.pad_token_id # 3. 모델 설정에 패딩 토큰 ID 반영

```

숨겨진 출력 표시

```python
questions = [
    "Parameter-Efficient Fine Tuning에 대해 알려줘",
    "LLM에서 가장 유명한 LLM에는 뭐가 있어?",
    "What is a famous tall tower in Seoul?",
    "LLM에서 파인튜닝이 뭐야?",
    "운영체제가 뭐하는 거야?.",
    "메모리가 뭐야?"
]
```

Alpaca 형식이란 ?

- Standford Alpaca 모델에서 처음 사용된 텍스트 정형화 방식
- OpenAI의 text-davinci-003로 생성한 데이터셋을 활용하여 "Instruction Following' 형식의 학습을 지원
- 주어진 명령어(instruction)에 대한 적절한 응답(response)을 생성하도록 모델을 학습시키는 방식

```python
# Prompt/Response Format 관련 설정
EOS_TOKEN = tokenizer.eos_token  # tokenizer.eos_token을 가져와서 할당함. LLaMA모델에서는 기본적으로 EOS_TOKEN="</s>"로 설정됨

def convert_to_alpaca_format(instruction, response): # 입력된 instruction(질문)과 response(답변)을 Alpaca 형식의 텍스트로 변환하는 함수임.
    alpaca_format_str = f"""Below is an instruction that describes a task. Write a response that appropriately completes the request.₩
    ₩n₩n### Instruction:₩n{instruction}₩n₩n### Response:₩n{response}₩
    """

    return alpaca_format_str
```

LLaMA3-8B(Base Model & Evaluation) 실행 (3)

```python
# 주어진 명령어(Instructon)를 LLaMA 모델에 입력하고, AI가 생성한 응답을 반환하는 역할을 함.
# 즉, LLM이 Instruction을 기반으로 텍스트를 생성하도록 테스트하는 함수임.
def test_model(instruction_str, model): # 사용자가 입력하는 질문(Instruction)
    inputs = tokenizer(
    [
        convert_to_alpaca_format(instruction_str,"",) # ""이것은 모델이 직접 응답을 생성해야 하기 때문
    ], return_tensors = "pt").to("cuda")
    outputs = model.generate(**inputs,
                             max_new_tokens = 128, # 한번에 생성할 최대 토큰수(128개)
                             use_cache = True, # 캐시 활용
                             temperature = 0.05, # 낮은 값 (즉, 더 결정적인 응답 생성, 창의성은 낮은)
                             top_p = 0.95) # 상위 95% 확률을 가진 토큰만 선택함
    return(tokenizer.batch_decode(outputs)[0]) # 모델이 생성한 토큰 출력을 텍스트로 변환하여 반환
```

tokenizer([…], return_tensors="pt"): […]은 변환된 텍스트를 모델이 처리할 수 있도록 토큰화 및 Pytorch 텐서(pt) 변환

.to("cuda") : 입력 텐서를 GPU에서 실행하도록 이동

```python
# 여러 개의 질문(questions)목록을 모델에 입력하여 응답을 생성하고, 이를 저장하는 과정을 수행함.
answers_dict = {
  "base_model_answers": [],
 }
for idx, question in enumerate(questions):
    print(f"Processing EXAMPLE {idx}==============")
    base_model_output = test_model(question, model)
    answers_dict['base_model_answers'].append(base_model_output)
```

LLaMA3-8B(Base Model & Evaluation) 실행 (4)

```python
# 모델이 생성한 응답을 정리하여 출력하는 함수임

def simple_format(text, width=120):
    return '₩n'.join(line[i:i+width] for line in text.split('₩n') for i in range(0, len(line), width))


for idx, question in enumerate(questions):
    print(f"EXAMPLE {idx}==============")
    print(f"Question: {question}")

    print("<<Base Model 답변>>")
    base_model_answer = answers_dict["base_model_answers"][idx].split("### Response:")[1]
    print(simple_format(base_model_answer))
    print("---")

```

실행 순서	설명
① `text.split('\n')` 실행	`\n` 기준으로 문자열을 **리스트로 변환**
② `for line in text.split('\n')` 실행	리스트에서 한 줄씩 `line`에 저장
③ `for i in range(0, len(line), width)` 실행	`width` 크기로 줄을 나누기 위한 반복
④ `line[i:i+width]` 실행	`width` 크기만큼 슬라이싱 수행
⑤ `'\n'.join(...)` 실행	슬라이싱된 결과를 `\n`으로 합쳐 반환

LLaMA3-8B(Base Model & Evaluation) 실행 (5)

```
EXAMPLE 0==============
Question: Parameter-Efficient Fine Tuning에 대해 알려줘
<<Base Model 답변>>
     Parameter-Efficient Fine Tuning은 모델의 파라미터를 효율적으로 사용하는 방법이다. 모델의 파라미터를 효율적으로 사용하
 파라미터를 효율적으로 사용하는 방법은 모델의 파라미터를 효율적으로 사용하는 방법이다. 모델의 파라미터를 효율적으로 사용하
는 방법은 모델의 파라
---
EXAMPLE 1==============
Question: LLM에서 가장 유명한 LLM에는 뭐가 있어?
<<Base Model 답변>>
    <|end_of_text|>
---
EXAMPLE 2==============
Question: What is a famous tall tower in Seoul?
<<Base Model 답변>>
     The N Seoul Tower is a communication and observation tower located in Yongsan-dong, Yongsan-gu, Seoul, South Korea.
 It is situated on Namsan Mountain, and is 236 meters (774 ft) in height. The tower is used for communication and broadc
asting purposes, and is also a tourist attraction. The tower is located in the Yongsan District of Seoul, and is accessi
ble by cable car, or by a series of escalators. The tower is lit up at night, and is visible throughout the city. The to
wer is also used for weddings and other events. The tower is also used for broadcasting
---
EXAMPLE 3==============
Question: LLM에서 파인튜닝이 뭐야?
<<Base Model 답변>>
    <|end_of_text|>
---
EXAMPLE 4==============
Question: 운영체제가 뭐하는 거야?.
<<Base Model 답변>>
    <|end_of_text|>
---
EXAMPLE 5==============
Question: 메모리가 뭐야?
<<Base Model 답변>>
    <|end_of_text|>
```

수고했어요.!!!