

- HG

```
import os
os.environ['HF_ENDPOINT'] = 'https://hf-mirror.com'
```

- `export USE_MODELSCOPE_HUB=1` `export HF_ENDPOINT=https://hf-mirror.com`

- LoraConfig

```
self.loraconfig = LoraConfig(  
    r=config["lora"]["r"],  
    target_modules=config["lora"]["target_modules"],  
    bias=config["lora"]["bias"],  
    lora_alpha=config["lora"]["lora_alpha"],  
    lora_dropout=config["lora"]["lora_dropout"],  
)
```

- r ローラーのアルファ
 - lora_alpha ローラーのアルファ
 - lora_dropout ローラーのドロップアウト
 - target_modules ローラーのターゲットモジュール

- get peft model

- `lora` `.print_trainable_parameters()`
 - `.print_trainable_parameters()`

```
self.peft_model = get_peft_model(self.model, self.loraconfig)
    self.peft_model.print_trainable_parameters()
```

transformers

- #### • AutoModelForCausalLM


```
self.model = AutoModelForCausalLM.from_pretrained(  
    config["model"]["model_path"],  
    device_map=config["model"]["device_map"],  
    trust_remote_code=config["model"]["trust_remote_code"],
```

```
        quantization_config=quan_configs,  
    )
```

- AutoTokenizer

- AutoTokenizer

```
self.tokenizer = AutoTokenizer.from_pretrained(  
    config["model"]["model_path"],  
    trust_remote_code=config["tokenizer"]["trust_remote_code"],  
)
```

- Tokenizer

- max_length 亂数生成器
 - return_tensors 亂数 pt 亂数生成器
 - ``truncation`` 亂数生成器

```
tokenized = self.tokenizer(  
    combined_texts,  
    truncation=True,  
    max_length=config["tokenizer"]["max_length"],  
    padding=False,  
    return_tensors=None # 亂数Python乱数生成器  
)
```

- .tokenizer.encode 亂数生成器 Token ID 亂数

- text 亂数生成器
 - add_special_tokens 亂数生成器 token
 - max_length 亂数
 - truncation 亂数生成器
 - return_tensors 亂数生成器 pt 亂数

```
turn_tokens = self.tokenizer.encode(  
    self.tokenizer.apply_chat_template([turn]),  
    tokenize=False),  
    add_special_tokens=False  
)
```

- .apply_chat_template 亂数生成器

- tokenize 亂数生成器 token IDs 亂数 true 亂数生成器
 - return_tensors 亂数生成器 "pt" (PyTorch), "np" (NumPy)
 - max_length 亂数 token 亂数

- truncation 亂数生成
- padding 亂数生成 "longest" □ True □ False
- return_dict 亂数生成

```
tokenized = self.tokenizer.apply_chat_template(
    conversation,
    tokenize=True,
    max_length=config["tokenizer"]["max_length"],
    truncation=config["tokenizer"]["truncation"],
    return_tensors=None,
    return_dict=True
)
```

- BitsAndBytesConfig

- 亂数生成 load_model
 - load_in_8bit 亂数生成8bit乱数生成
 - load_in_4bit 亂数生成4bit乱数生成
 - torch_dtype 亂数生成

```
quan_configs = BitsAndBytesConfig(
    load_in_8bit=config["model"]["load_in_8bit"],
    torch_dtype=config["model"]["torch_dtype"]
)
```

- Trainer

- 亂数生成
 - model 亂数生成模型 peft_model
 - tokenizer 亂数生成
 - args □ TrainingArguments □□
 - train_dataset 亂数生成
 - eval_dataset 亂数生成
 - data_collator 亂数生成
 - callbacks 亂数生成

```
trainer = Trainer(
    model=model,
    tokenizer=self.tokenizer,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    data_collator=data_collator,
```

```
        callbacks=[EarlyStoppingCallback(config["training"]  
["patience"])]  
    )
```

- TrainingArguments

- `output_dir` 旣定值
 - `learning_rate` 旣定值
 - `num_train_epochs` 旣定值
 - `weight_decay` 旣定值L2正則化
 - `per_device_train_batch_size` 旣定值
 - `per_device_eval_batch_size` 旣定值
 - `eval_strategy` 旣定值
 - `save_strategy` 旣定值
 - `eval_steps` 旣定值
 - `save_steps` 旣定值
 - `logging_steps` 旣定值
 - `logging_dir` `TensorBoard` 旣定值
 - `report_to` 旣定值
 - `fp16` 旣定值
 - `gradient_accumulation_steps` 旣定值
 - `gradient_checkpointing` 旣定值
 - `max_grad_norm` 旣定值
 - `load_best_model_at_end` 旣定值
 - 旣定值
 - `lr_scheduler_type` 旣定值
 - `cosine` 旣定值
 - `linear` 旣定值
 - `metric_for_best_model` 旣定值

```
training_args = TrainingArguments(  
    output_dir=config["model"]["output_path"],  
    learning_rate=config["training"]["learning_rate"],  
    num_train_epochs=config["training"]["num_epochs"],
```

```

        weight_decay=config["training"]["weight_decay"],
        per_device_train_batch_size=config["training"]
        ["per_device_train_batch_size"],
        per_device_eval_batch_size=config["training"]
        ["per_device_eval_batch_size"],
        eval_strategy=config["training"]["eval_strategy"],
        save_strategy=config["training"]["save_strategy"],
        eval_steps=config["training"]["eval_steps"],
        save_steps=config["training"]["save_steps"],
        logging_steps=config["training"]["logging_steps"],
        logging_dir=config["training"]["logging_dir"],
        report_to=config["training"]["report_to"],
        load_best_model_at_end=config["training"]
        ["load_best_model_at_end"],
        fp16=config["training"]["fp16"],
        gradient_accumulation_steps=config["training"]
        ["gradient_accumulation_steps"],
        dataloader_num_workers=4, # データローダーの数
    )
)

```

- `datasets.Dataset` □ Hugging Face datasets データセットの操作
 - `.from_dict()` ディクショナリをデータセットに変換
 - `.map()` データをマッピングする
 - `function` マップ関数
 - `batched` バッチ化
 - `batch_size` バッチサイズ
 - `num_proc` プロセス数
 - `.save_to_disk` データセットをディスクに保存
- `load_dataset` □ datasets データセットの読み込み
 - `split` データを訓練用、検証用、評価用に分割
 - `streaming` データをストリーミングで読み込む（OOM問題を防ぐ）

```

self.dataset = load_dataset("BelleGroup/multiturn_chat_0.8M",
split="train[:70%]", streaming=True)

```

- `DataCollatorForSeq2Seq` □ モデルの入力データを整形式に変換
 - `tokenizer` テキスト tokenizer
 - `model` モデルの種類
 - `padding` パディング
 - `max_length` 最大長
 - `pad_to_multiple_of` パディング幅
 - `label_pad_token_id` ラベルパッド Token ID
 - `return_tensors` テンソル形式

```
data_collator = DataCollatorForSeq2Seq(  
    tokenizer=self.tokenizer,  
    pad_to_multiple_of=config["dataloader"]  
    ["pad_to_multiple_of"],  
    return_tensors=config["dataloader"]["return_tensors"],  
    padding=config["dataloader"]["padding"],  
    label_pad_token_id=config["dataloader"]  
    ["label_pad_token_id"]  
)
```