

• 데이터 로드

```
31 def load_dataset(self, dataset_id, tokenizer):
32     def tokenization(examples):
33         sources = []
34         targets = []
35         prompt = self.PROMPT_TEMPLATE
36         for instruction, input, output in zip(examples['instruction'], examples['input'], examples['output']):
37             if input is not None and input != "":
38                 instruction = instruction + '\n' + input
39                 source = prompt.format_map({'instruction': instruction})
40                 target = f"{output}{tokenizer.eos_token}"
41
42                 sources.append(source)
43                 targets.append(target)
44
45         tokenized_sources = tokenizer(sources, return_attention_mask=False)
46         tokenized_targets = tokenizer(targets, return_attention_mask=False, add_special_tokens=False)
47
48         all_input_ids = []
49         all_labels = []
50         for s, t in zip(tokenized_sources['input_ids'], tokenized_targets['input_ids']):
51             input_ids = torch.LongTensor(s + t)[:self.MAX_SEQ_LEN]
52             labels = torch.LongTensor([-100] * len(s) + t)[:self.MAX_SEQ_LEN]
53             assert len(input_ids) == len(labels)
54             all_input_ids.append(input_ids)
55             all_labels.append(labels)
56
57         results = {'input_ids': all_input_ids, 'labels': all_labels}
58         return results
59
60     all_datasets = []
61     raw_dataset = load_dataset(dataset_id)
62     tokenization_func = tokenization
```

Hugging Face Hub Model load

• 토큰화

```
31 def load_dataset(self, dataset_id, tokenizer):
32     def tokenization(examples):
33         sources = []
34         targets = []
35         prompt = self.PROMPT_TEMPLATE
36         for instruction, input, output in zip(examples['instruction'], examples['input'], examples['output']):
37             if input is not None and input != "":
38                 instruction = instruction + '\n' + input
39                 source = prompt.format_map({'instruction': instruction})
40                 target = f"{output}{tokenizer.eos_token}"
41
42                 sources.append(source)
43                 targets.append(target)
44
45         tokenized_sources = tokenizer(sources, return_attention_mask=False)
46         tokenized_targets = tokenizer(targets, return_attention_mask=False, add_special_tokens=False)
47
48         all_input_ids = []
49         all_labels = []
50         for s, t in zip(tokenized_sources['input_ids'], tokenized_targets['input_ids']):
51             input_ids = torch.LongTensor(s + t)[:self.MAX_SEQ_LEN]
52             labels = torch.LongTensor([-100] * len(s) + t)[:self.MAX_SEQ_LEN]
53             assert len(input_ids) == len(labels)
54             all_input_ids.append(input_ids)
55             all_labels.append(labels)
56
57         results = {'input_ids': all_input_ids, 'labels': all_labels}
58         return results
59
60     all_datasets = []
61     raw_dataset = load_dataset(dataset_id)
62     tokenization_func = tokenization
```

Tokenizer Text to Number ID

모델 학습을 위한 데이터 전처리

② 데이터 전처리

• 배치 및 Collator

```
1  from dataclasses import dataclass
2  from typing import Dict, Sequence
3
4  import torch
5  import transformers
6
7
8  @dataclass
9  class DataCollatorForSupervisedDataset:
10     tokenizer: transformers.PreTrainedTokenizer
11
12     def __call__(self, instances: Sequence[Dict]) -> Dict[str, torch.Tensor]:
13         input_ids, labels = tuple(
14             [instance[key] for instance in instances] for key in ("input_ids", "labels")
15         )
16         input_ids = torch.nn.utils.rnn.pad_sequence(
17             input_ids, batch_first=True, padding_value=self.tokenizer.pad_token_id
18         )
19         labels = torch.nn.utils.rnn.pad_sequence(
20             labels, batch_first=True, padding_value=-100
21         )
22         return dict(
23             input_ids=input_ids,   패딩을 통해 배치의 데이터 길이 최적화
24             labels=labels,
25             attention_mask=input_ids.ne(self.tokenizer.pad_token_id),
26         )
```

• 자동화 생성

```
62     raw_dataset = load_dataset(dataset_id)
63     tokenization_func = tokenization
64     tokenized_dataset = raw_dataset.map(
65         tokenization_func,
66         batched=True,
67         remove_columns=["instruction", "input", "output"],
68         keep_in_memory=False,
69         desc="preprocessing on dataset",
70     )
71     processed_dataset = tokenized_dataset
72     processed_dataset.set_format('torch')
73     all_datasets.append(processed_dataset['train'])
74     all_datasets = concatenate_datasets(all_datasets)
75     return all_datasets
76
77     def get_data_collator(self, tokenizer):
78         collator = DataCollatorForSupervisedDataset(tokenizer)
79         return collator
```

Tokenizing DataSet

모델 학습을 위한 데이터 전처리

- 상세 설명

- 데이터 로드

- `load_dataset` 메서드를 통해서 `dataset_id`에 해당하는 데이터셋을 로드하고 Hub에 있는 데이터셋도 가져올 수 있습니다.

- 토큰화

- 텍스트 데이터를 숫자 ID로 바꾸는 토큰화 작업을 수행합니다.
 - Instruction, input, output 모델 학습에 필요한 `sources`와 `targets` 형식으로 나뉘 뒤 `tokenized`로 숫자 ID 시퀀스로 변환 합니다.

- 배치 및 Coallator

- 데이터의 길이가 서로 다르기 때문에 `Input_ids`와 `labels`를 패딩해 길이를 맞춰 줍니다.
 - `DataCollatorForSupervisedDataset` 클래스에서 서로 다른 데이터가 패딩되어 같은 길이로 맞춰 집니다.

- 최종 배치 생성

- 토큰화된 데이터셋을 최종적으로 `DataCollatorForSupervisedDataset` 클래스에서 배치로 생성합니다.