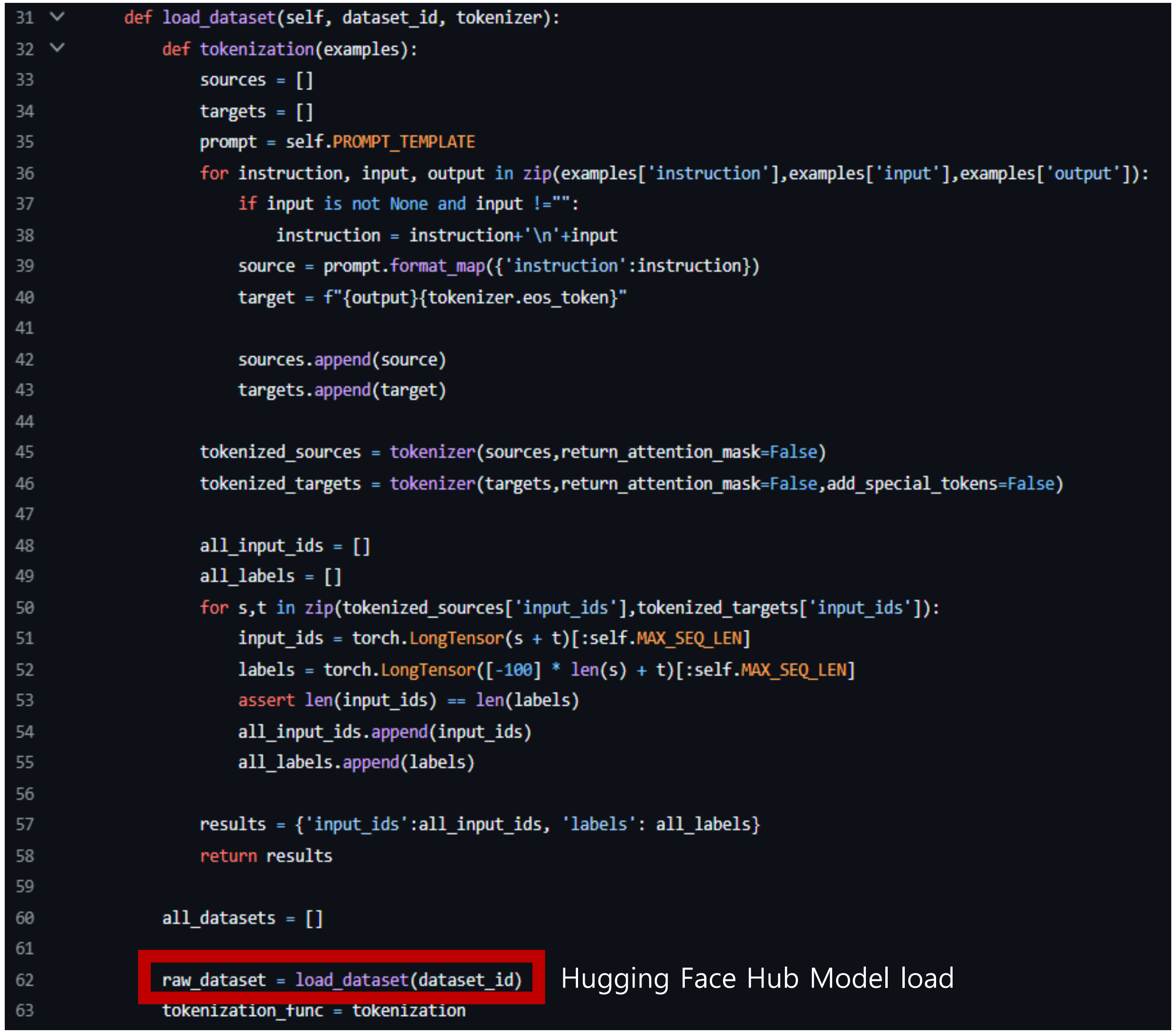
모델 학습을 위한 데이터 전처리

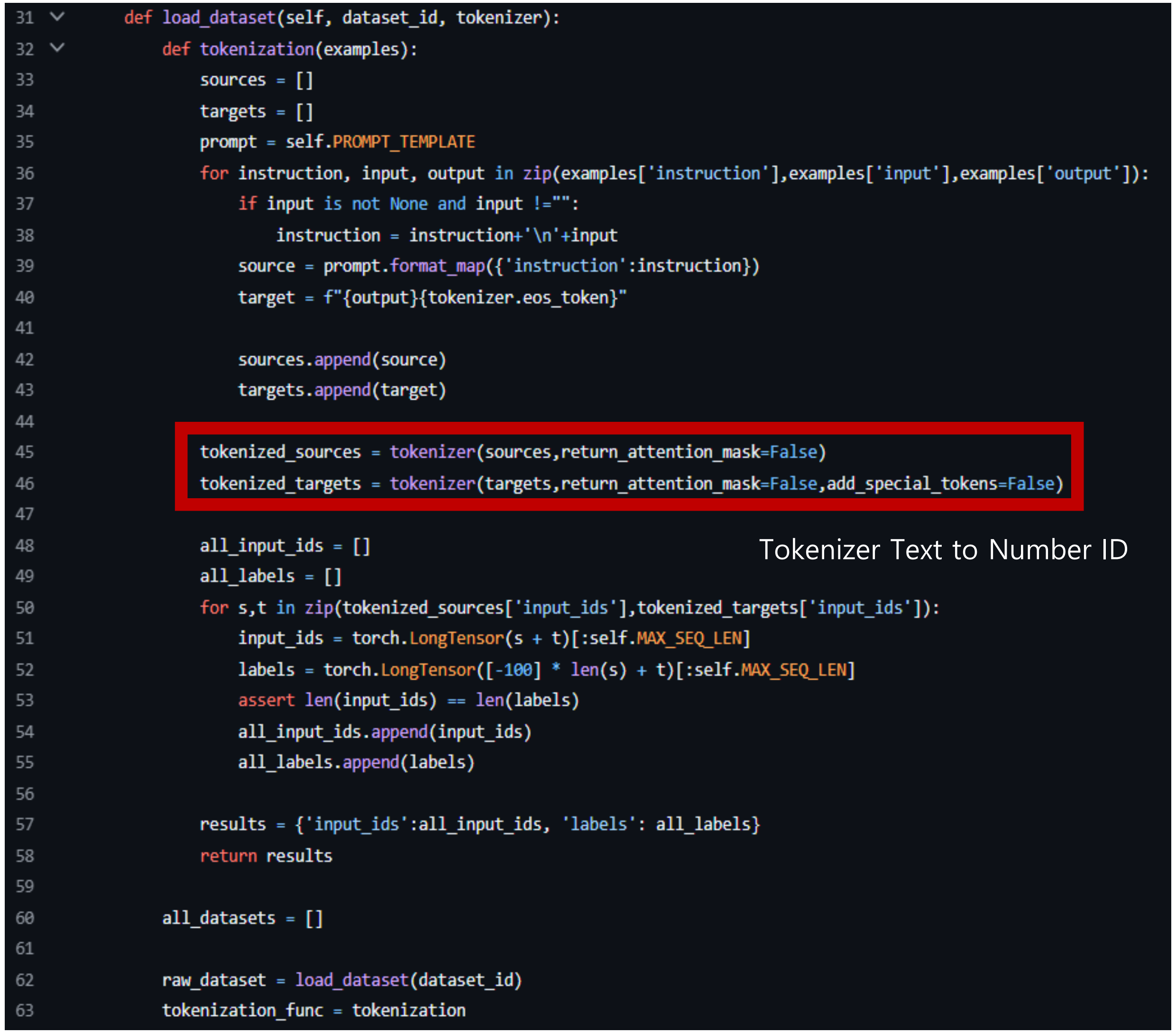
- 데이터 로드



```
    def load_dataset(self, dataset_id, tokenizer):
        def tokenization(examples):
            sources = []
            targets = []
            prompt = self.PROMPT_TEMPLATE
            for instruction, input, output in zip(examples['instruction'],examples['input'],examples['output']):
                if input is not None and input !="":
                    instruction = instruction+'\n'+input
                source = prompt.format_map({'instruction':instruction})
                target = f"{output}{tokenizer.eos_token}"

                sources.append(source)
                targets.append(target)

            tokenized_sources = tokenizer(sources,return_attention_mask=False)
            tokenized_targets = tokenizer(targets,return_attention_mask=False,add_special_tokens=False)

            all_input_ids = []
            all_labels = []
            for s,t in zip(tokenized_sources['input_ids'],tokenized_targets['input_ids']):
                input_ids = torch.LongTensor(s + t)[:self.MAX_SEQ_LEN]
                labels = torch.LongTensor([-100] * len(s) + t)[:self.MAX_SEQ_LEN]
                assert len(input_ids) == len(labels)
                all_input_ids.append(input_ids)
                all_labels.append(labels)

            results = {'input_ids':all_input_ids, 'labels': all_labels}
            return results

        all_datasets = []

        raw_dataset = load_dataset(dataset_id)
        tokenization_func = tokenization
```

Hugging Face Hub Model load

- 토큰화



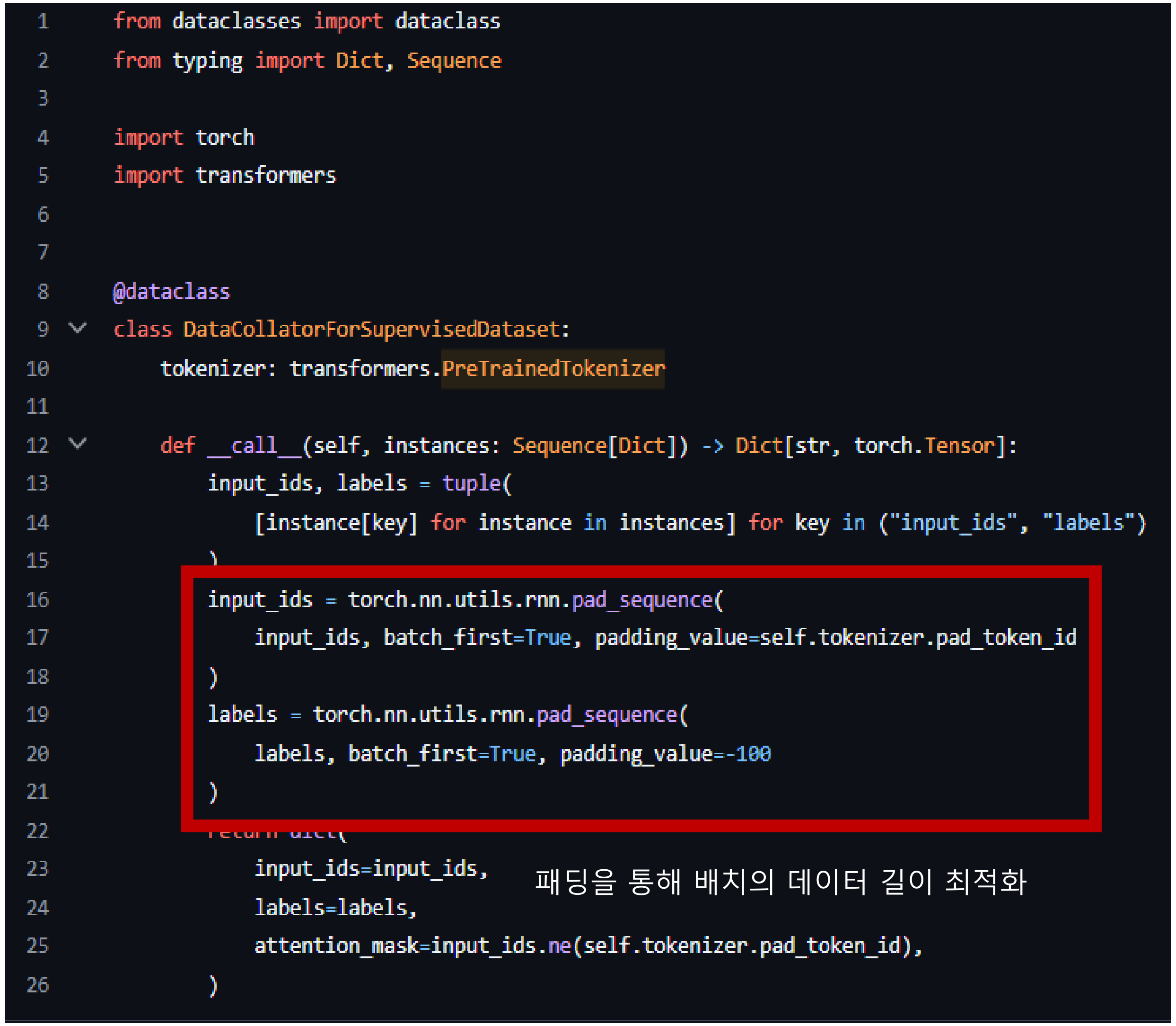
```
    def load_dataset(self, dataset_id, tokenizer):
        def tokenization(examples):
            sources = []
            targets = []
            prompt = self.PROMPT_TEMPLATE
            for instruction, input, output in zip(examples['instruction'],examples['input'],examples['output']):
                if input is not None and input !="":
                    instruction = instruction+'\n'+input
                source = prompt.format_map({'instruction':instruction})
                target = f"{output}{tokenizer.eos_token}"

                sources.append(source)
                targets.append(target)

            tokenized_sources = tokenizer(sources,return_attention_mask=False)
            tokenized_targets = tokenizer(targets,return_attention_mask=False,add_special_tokens=False)

            all_input_ids = []
            all_labels = []
            for s,t in zip(tokenized_sources['input_ids'],tokenized_targets['input_ids']):
                input_ids = torch.LongTensor(s + t)[:self.MAX_SEQ_LEN]
                labels = torch.LongTensor([-100] * len(s) + t)[:self.MAX_SEQ_LEN]
                assert len(input_ids) == len(labels)
                all_input_ids.append(input_ids)
                all_labels.append(labels)

            results = {'input_ids':all_input_ids, 'labels': all_labels}
            return results

        all_datasets = []

        raw_dataset = load_dataset(dataset_id)
        tokenization_func = tokenization
```

Tokenizer Text to Number ID

GitHub 주소: https://github.com/EDDI-RobotAcademy/SI-Follow-LLAMA/tree/main/dataset

모델 학습을 위한 데이터 전처리

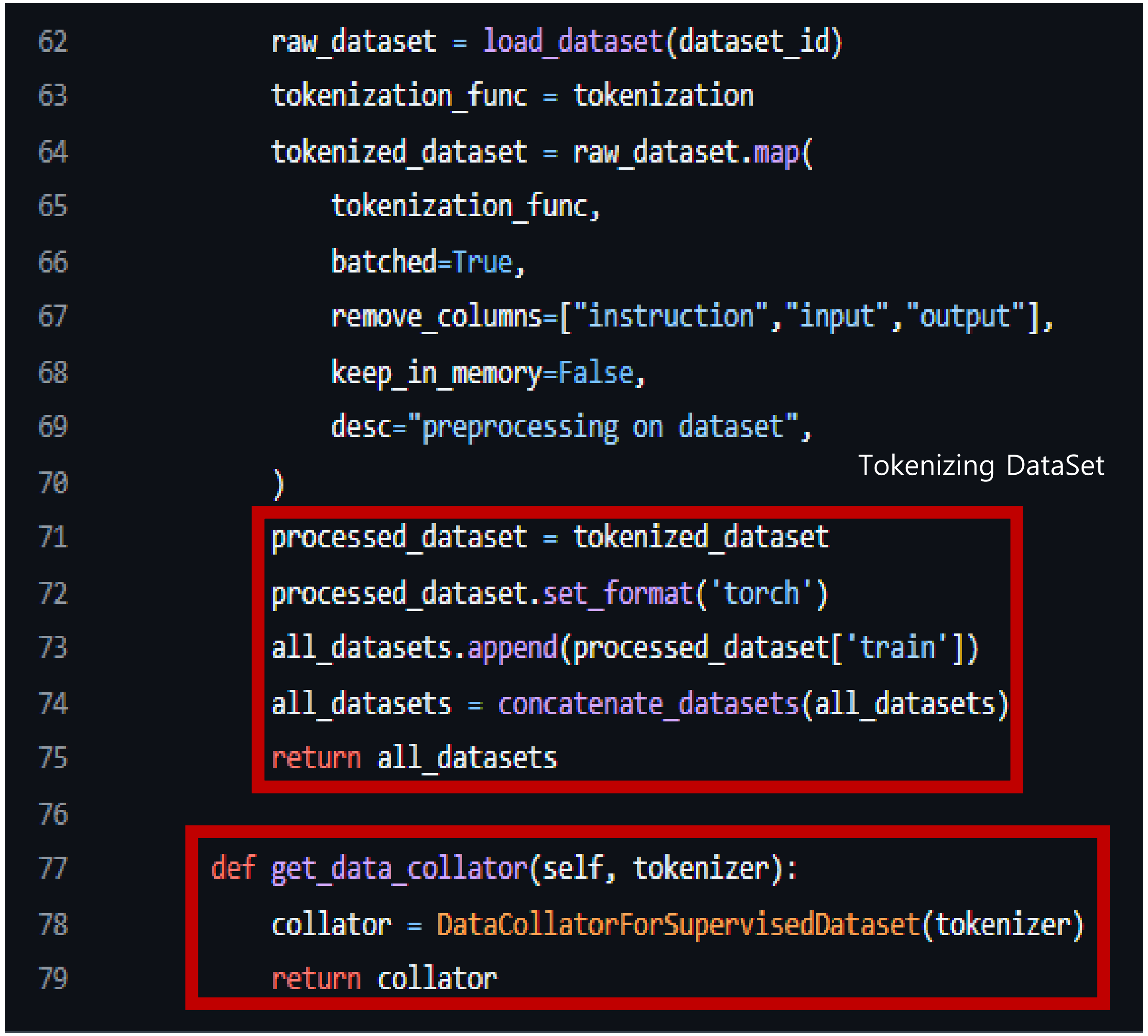
- 배치 및 Collator



```python
from dataclasses import dataclass
from typing import Dict, Sequence

import torch
import transformers


@dataclass
class DataCollatorForSupervisedDataset:
    tokenizer: transformers.PreTrainedTokenizer

    def __call__(self, instances: Sequence[Dict]) -> Dict[str, torch.Tensor]:
        input_ids, labels = tuple(
            [instance[key] for instance in instances] for key in ("input_ids", "labels")
        )
        input_ids = torch.nn.utils.rnn.pad_sequence(
            input_ids, batch_first=True, padding_value=self.tokenizer.pad_token_id
        )
        labels = torch.nn.utils.rnn.pad_sequence(
            labels, batch_first=True, padding_value=-100
        )
        return dict(
            input_ids=input_ids,
            labels=labels,
            attention_mask=input_ids.ne(self.tokenizer.pad_token_id),
        )
```

패딩을 통해 배치의 데이터 길이 최적화

- 자동화 생성



```python
        raw_dataset = load_dataset(dataset_id)
        tokenization_func = tokenization
        tokenized_dataset = raw_dataset.map(
            tokenization_func,
            batched=True,
            remove_columns=["instruction","input","output"],
            keep_in_memory=False,
            desc="preprocessing on dataset",
        )
        processed_dataset = tokenized_dataset
        processed_dataset.set_format('torch')
        all_datasets.append(processed_dataset['train'])
        all_datasets = concatenate_datasets(all_datasets)
        return all_datasets

    def get_data_collator(self, tokenizer):
        collator = DataCollatorForSupervisedDataset(tokenizer)
        return collator
```

Tokenizing DataSet

GitHub 주소: https://github.com/EDDI-RobotAcademy/SI-Follow-LLAMA/tree/main/dataset