```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)
```

```
Thu Dec  7 17:43:56 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   49C    P8    10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

We need the `librosa` package to load audio files and the `jiwer` to evaluate our fine-tuned model using the word error rate (WER) metric [1].

```
%%capture
!pip install datasets==1.18.3
!pip install transformers==4.17.0
!pip install jiwer
```

Prepare Data, Tokenizer, Feature Extractor

New Section

Create Wav2Vec2CTCTokenizer

Let's start by loading the dataset and taking a look at its structure.

```
from datasets import load_dataset, load_metric

timit = load_dataset("timit_asr")
```

```
Downloading:                                                    7.06k/? [00:00<00:00, 442kB/s]
Downloading:                                                    2.64k/? [00:00<00:00, 142kB/s]
Downloading and preparing dataset timit_asr/clean (download: 828.75 MiB, generated: 7.96
Downloading: 100%                                               869M/869M [02:45<00:00, 6.90MB/s]
```

```
Dataset timit_asr downloaded and prepared to /root/.cache/huggingface/datasets/timit_asr
```

```
timit
```

```
DatasetDict({
    train: Dataset({
        features: ['file', 'audio', 'text', 'phonetic_detail', 'word_detail',
'dialect_region', 'sentence_type', 'speaker_id', 'id'],
        num_rows: 4620
    })
    test: Dataset({
        features: ['file', 'audio', 'text', 'phonetic_detail', 'word_detail',
'dialect_region', 'sentence_type', 'speaker_id', 'id'],
        num_rows: 1680
    })
})
```

```
timit = timit.remove_columns(["phonetic_detail", "word_detail", "dialect_region", "id", "ser
```

Let's write a short function to display some random samples of the dataset and run it a couple of times to get a feeling for the transcriptions.

```
from datasets import ClassLabel
import random
import pandas as pd
from IPython.display import display, HTML

def show_random_elements(dataset, num_examples=10):
    assert num_examples <= len(dataset), "Can't pick more elements than there are in the dat
    picks = []
    for _ in range(num_examples):
        pick = random.randint(0, len(dataset)-1)
        while pick in picks:
            pick = random.randint(0, len(dataset)-1)
        picks.append(pick)

    df = pd.DataFrame(dataset[picks])
    display(HTML(df.to_html()))


show_random_elements(timit["train"].remove_columns(["audio", "file"]), num_examples=10)
```

	text
0	What targets have we successfully knocked out?
1	Both figures would go higher in later years.
2	Jeff's toy go-cart never worked!
3	Does Hindu ideology honor cows?
4	Should giraffes be kept in small zoos?
5	And their chroniclers are not the dramatic poets but the prose novelists.
6	It is a big project, not to be taken lightly.
7	We apply auditory modeling to computer speech recognition.
8	Don't ask me to carry an oily rag like that.
9	Those who are not purists use canned vegetables when making stew.

```
import re
chars_to_ignore_regex = '[\,\?\.\!\-\;\:\"]'

def remove_special_characters(batch):
    batch["text"] = re.sub(chars_to_ignore_regex, '', batch["text"]).lower() + " "
    return batch


timit = timit.map(remove_special_characters)
```

```
WARNING:datasets.fingerprint:Parameter 'function'=<function remove_special_characters at
```

4620/? [00:00<00:00, 10758.98ex/s]

1680/? [00:00<00:00, 7869.35ex/s]

```
show_random_elements(timit["train"].remove_columns(["audio", "file"]))
```

	text
0	often you'll get back more than you put in
1	to be passive to be girlishly shy was palpably absurd
2	don't ask me to carry an oily rag like that
3	the hallway opens into a huge chamber
4	irish youngsters eat fresh kippers for breakfast
5	don't ask me to carry an oily rag like that
6	there are canoes ideal for fishing in protected waters or for camping trips
7	don't ask me to carry an oily rag like that
8	turbulent tides rose as much as fifty feet
9	trespassing is forbidden and subject to penalty

```
def extract_all_chars(batch):
  all_text = " ".join(batch["text"])
  vocab = list(set(all_text))
  return {"vocab": [vocab], "all_text": [all_text]}
```

```
vocabs = timit.map(extract_all_chars, batched=True, batch_size=-1, keep_in_memory=True, remc
```

100% 1/1 [00:00<00:00, 15.26ba/s]

100% 1/1 [00:00<00:00, 23.64ba/s]

Now, we create the union of all distinct letters in the training dataset and test dataset and convert the resulting list into an enumerated dictionary.

```
vocab_list = list(set(vocabs["train"]["vocab"][0]) | set(vocabs["test"]["vocab"][0]))
```

```
vocab_dict = {v: k for k, v in enumerate(vocab_list)}
vocab_dict
```

```
{'n': 0,
 'f': 1,
 'w': 2,
 "'": 3,
 's': 4,
 'k': 5,
 'c': 6,
 'l': 7,
 'e': 8,
 'z': 9,
 'u': 10,
 'v': 11,
 'y': 12,
 'a': 13,
 'x': 14,
 'm': 15,
 'q': 16,
 'j': 17,
 'd': 18,
 'h': 19,
 't': 20,
 'i': 21,
 'o': 22,
 'b': 23,
 'g': 24,
 'r': 25,
 ' ': 26,
 'p': 27}
```

```
vocab_dict["|"] = vocab_dict[" "]
del vocab_dict[" "]
```

```
vocab_dict["[UNK]"] = len(vocab_dict)
vocab_dict["[PAD]"] = len(vocab_dict)
len(vocab_dict)
```

```
30
```

Let's now save the vocabulary as a json file.

```
import json
with open('vocab.json', 'w') as vocab_file:
    json.dump(vocab_dict, vocab_file)
```

In a final step, we use the json file to instantiate an object of the `Wav2Vec2CTCTokenizer` class.

```
from transformers import Wav2Vec2CTCTokenizer

tokenizer = Wav2Vec2CTCTokenizer("./vocab.json", unk_token="[UNK]", pad_token="[PAD]", word_
```

Create Wav2Vec2 Feature Extractor

New Section

```
from transformers import Wav2Vec2FeatureExtractor

feature_extractor = Wav2Vec2FeatureExtractor(feature_size=1, sampling_rate=16000, padding_va


from transformers import Wav2Vec2Processor

processor = Wav2Vec2Processor(feature_extractor=feature_extractor, tokenizer=tokenizer)
```

Next, we can prepare the dataset.

```
timit["train"][0]["file"]
```

```
'/root/.cache/huggingface/datasets/downloads/extracted/404950a46da14eac65eb4e2a8317b137
2fb3971d980d91d5d5b221275b1fd7e0/data/TRAIN/DR4/MMDM0/SI681.WAV'
```

`Wav2Vec2` expects the input in the format of a 1-dimensional array of 16 kHz. This means that the audio file has to be loaded and resampled.

Thankfully, `datasets` does this automatically when calling the column `audio`. Let try it out.

```
timit["train"][0]["audio"]
```

```
{'path':
'/root/.cache/huggingface/datasets/downloads/extracted/404950a46da14eac65eb4e2a8317b1372
 'array': array([-2.1362305e-04,  6.1035156e-05,  3.0517578e-05, ...,
        -3.0517578e-05, -9.1552734e-05, -6.1035156e-05], dtype=float32),
 'sampling_rate': 16000}
```

```
import IPython.display as ipd
import numpy as np
import random

rand_int = random.randint(0, len(timit["train"]))

print(timit["train"][rand_int]["text"])
ipd.Audio(data=np.asarray(timit["train"][rand_int]["audio"]["array"]), autoplay=True, rate=1
```

```
     the proof that you are seeking is not available in books

           0:00 / 0:03
```

```
rand_int = random.randint(0, len(timit["train"]))

print("Target text:", timit["train"][rand_int]["text"])
print("Input array shape:", np.asarray(timit["train"][rand_int]["audio"]["array"]).shape)
print("Sampling rate:", timit["train"][rand_int]["audio"]["sampling_rate"])
```

```
     Target text: first add milk to the shredded cheese
     Input array shape: (38605,)
     Sampling rate: 16000
```

```
def prepare_dataset(batch):
    audio = batch["audio"]

    # batched output is "un-batched" to ensure mapping is correct
    batch["input_values"] = processor(audio["array"], sampling_rate=audio["sampling_rate"]).
    batch["input_length"] = len(batch["input_values"])

    with processor.as_target_processor():
        batch["labels"] = processor(batch["text"]).input_ids
    return batch
```

Let's apply the data preparation function to all examples.

```
timit = timit.map(prepare_dataset, remove_columns=timit.column_names["train"], num_proc=4)
```

```
max_input_length_in_sec = 4.0
timit["train"] = timit["train"].filter(lambda x: x < max_input_length_in_sec * processor.fea
```

```
      100%                                                  5/5 [00:00<00:00, 154.87ba/s]
```

Awesome, now we are ready to start training!

```
import torch

from dataclasses import dataclass, field
from typing import Any, Dict, List, Optional, Union

@dataclass
class DataCollatorCTCWithPadding:
    """
    Data collator that will dynamically pad the inputs received.
    Args:
        processor (:class:`~transformers.Wav2Vec2Processor`)
            The processor used for proccessing the data.
        padding (:obj:`bool`, :obj:`str` or :class:`~transformers.tokenization_utils_base.Pa
            Select a strategy to pad the returned sequences (according to the model's paddir
            among:
            * :obj:`True` or :obj:`'longest'`: Pad to the longest sequence in the batch (or
              sequence if provided).
            * :obj:`'max_length'`: Pad to a maximum length specified with the argument :obj:
              maximum acceptable input length for the model if that argument is not providec
            * :obj:`False` or :obj:`'do_not_pad'` (default): No padding (i.e., can output a
              different lengths).
    """

    processor: Wav2Vec2Processor
    padding: Union[bool, str] = True

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[st
        # split inputs and labels since they have to be of different lenghts and need
        # different padding methods
        input_features = [{"input_values": feature["input_values"]} for feature in features]
        label_features = [{"input_ids": feature["labels"]} for feature in features]

        batch = self.processor.pad(
            input_features,
            padding=self.padding,
            return_tensors="pt",
        )
        with self.processor.as_target_processor():
            labels_batch = self.processor.pad(
                label_features,
                padding=self.padding,
                return_tensors="pt",
            )

        # replace padding with -100 to ignore loss correctly
        labels = labels_batch["input_ids"].masked_fill(labels_batch.attention_mask.ne(1), -1

        batch["labels"] = labels

        return batch
```

```
data_collator = DataCollatorCTCWithPadding(processor=processor, padding=True)
```

Next, the evaluation metric is defined. As mentioned earlier, the predominant metric in ASR is the word error rate (WER), hence we will use it in this notebook as well.

```
wer_metric = load_metric("wer")
```

```
Downloading:                                                    4.48k/? [00:00<00:00, 266kB/s]
```

```
from transformers import TrainingArguments
output_dir = '/content/repo_name/'
training_args = TrainingArguments(
  output_dir=output_dir,
  group_by_length=True,
  per_device_train_batch_size=8,
  evaluation_strategy="steps",
  num_train_epochs=30,
  fp16=True,
  gradient_checkpointing=True,
  save_steps=500,
  eval_steps=500,
  logging_steps=500,
  learning_rate=1e-4,
  weight_decay=0.005,
  warmup_steps=1000,
  save_total_limit=2,
)
```

Now, all instances can be passed to Trainer and we are ready to start training!

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    data_collator=data_collator,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=timit["train"],
    eval_dataset=timit["test"],
    tokenizer=processor.feature_extractor,
)
```

```
 Using amp half precision backend
```

```
function ConnectButton(){
    console.log("Connect pushed");
    document.querySelector("#top-toolbar > colab-connect-button").shadowRoot.querySelector("#connect"
}
setInterval(ConnectButton,60000);
```

```
trainer.train()
```

```
The following columns in the training set  don't have a corresponding argument in `Wav2V
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:306: FutureWarning:
  warnings.warn(
***** Running training *****
  Num examples = 3978
  Num Epochs = 30
  Instantaneous batch size per device = 8
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 14940
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
```

[2501/14940 18:59 < 1:34:33, 2.19 it/s, Epoch 5.02/30]

Step	Training Loss	Validation Loss	Wer
500	3.499400	1.680385	0.964579
1000	0.845700	0.563984	0.545310
1500	0.438600	0.462594	0.458687
2000	0.297700	0.422367	0.422162

[76/210 00:21 < 00:37, 3.53 it/s]

```
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-500
Configuration saved in /content/repo_name/checkpoint-500/config.json
Model weights saved in /content/repo_name/checkpoint-500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-500/preprocessor_config.json
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-1000
Configuration saved in /content/repo_name/checkpoint-1000/config.json
Model weights saved in /content/repo_name/checkpoint-1000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-1000/preprocessor_config.json
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-1500
Configuration saved in /content/repo_name/checkpoint-1500/config.json
Model weights saved in /content/repo_name/checkpoint-1500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-1500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-500] due to args.save_total_lim
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
```

```
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-2000
Configuration saved in /content/repo_name/checkpoint-2000/config.json
Model weights saved in /content/repo_name/checkpoint-2000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-2000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-1000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
```

[9001/14940 1:09:59 < 46:11, 2.14 it/s, Epoch 18.07/30]

Step	Training Loss	Validation Loss	Wer
500	3.499400	1.680385	0.964579
1000	0.845700	0.563984	0.545310
1500	0.438600	0.462594	0.458687
2000	0.297700	0.422367	0.422162
2500	0.227100	0.435139	0.410861
3000	0.194400	0.429970	0.394666
3500	0.159300	0.498936	0.390600
4000	0.137700	0.412528	0.392737
4500	0.124900	0.434000	0.376886
5000	0.108900	0.429038	0.382951
5500	0.101700	0.418560	0.371718
6000	0.092500	0.530390	0.384122
6500	0.084200	0.504215	0.371718
7000	0.072900	0.509529	0.373716
7500	0.069600	0.473375	0.372407
8000	0.060100	0.441955	0.365447
8500	0.059300	0.507855	0.358142

[4/210 00:00 < 01:00, 3.40 it/s]

```
Saving model checkpoint to /content/repo_name/checkpoint-2500
Configuration saved in /content/repo_name/checkpoint-2500/config.json
Model weights saved in /content/repo_name/checkpoint-2500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-2500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-1500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
```

```
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-3000
Configuration saved in /content/repo_name/checkpoint-3000/config.json
Model weights saved in /content/repo_name/checkpoint-3000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-3000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-2000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-3500
Configuration saved in /content/repo_name/checkpoint-3500/config.json
Model weights saved in /content/repo_name/checkpoint-3500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-3500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-2500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-4000
Configuration saved in /content/repo_name/checkpoint-4000/config.json
Model weights saved in /content/repo_name/checkpoint-4000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-4000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-3000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-4500
Configuration saved in /content/repo_name/checkpoint-4500/config.json
Model weights saved in /content/repo_name/checkpoint-4500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-4500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-3500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-5000
Configuration saved in /content/repo_name/checkpoint-5000/config.json
Model weights saved in /content/repo_name/checkpoint-5000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-5000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-4000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
```

```
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-5500
Configuration saved in /content/repo_name/checkpoint-5500/config.json
Model weights saved in /content/repo_name/checkpoint-5500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-5500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-4500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-6000
Configuration saved in /content/repo_name/checkpoint-6000/config.json
Model weights saved in /content/repo_name/checkpoint-6000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-6000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-5000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-6500
Configuration saved in /content/repo_name/checkpoint-6500/config.json
Model weights saved in /content/repo_name/checkpoint-6500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-6500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-5500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-7000
Configuration saved in /content/repo_name/checkpoint-7000/config.json
Model weights saved in /content/repo_name/checkpoint-7000/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-7000/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-6000] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
Saving model checkpoint to /content/repo_name/checkpoint-7500
Configuration saved in /content/repo_name/checkpoint-7500/config.json
Model weights saved in /content/repo_name/checkpoint-7500/pytorch_model.bin
Feature extractor saved in /content/repo_name/checkpoint-7500/preprocessor_config.json
Deleting older checkpoint [/content/repo_name/checkpoint-6500] due to args.save_total_li
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:429: UserWarning: torc
  warnings.warn(
The following columns in the evaluation set  don't have a corresponding argument in `Wav
***** Running Evaluation *****
  Num examples = 1680
  Batch size = 8
```