Lab 06 Fine-Tuning a Speech Recognition for French

22210802 Haeeul HWANG

```
%%capture

%pip install datasets
%pip install accelerate -U
%pip install transformers[torch] -U
%pip install evaluate
%pip install jiwer
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Common voice dataset

```
from datasets import load_dataset

cv_dataset = load_dataset("fsicoli/common_voice_17_0", "fr", split='test')
```

숨겨진 출력 표시

```
cv_dataset
```

```
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant'],
    num_rows: 16159
})
```

```
# save the dataset in the drive

save_path_json = '/content/drive/My Drive/cv_dataset.json'

cv_dataset.to_json(save_path_json)

print(f"Dataset saved to {save_path}")
```

```
Creating json from Arrow format: 100%  17/17 [00:00<00:00, 81.90ba/s]
Dataset saved to /content/drive/My Drive/cv_dataset.csv
```

```
# filter the rows with empty 'gender' value

cv_dataset = cv_dataset.filter(lambda x : x != '', input_columns=["gender"])
cv_dataset
```

```
Filter: 100%  16159/16159 [00:00<00:00, 302486.63 examples/s]
Dataset({
    features: ['client_id', 'path', 'audio', 'sentence', 'up_votes', 'down_votes', 'age',
'gender', 'accent', 'locale', 'segment', 'variant'],
    num_rows: 1851
})
```

Text processing

1. Why do we need to remove punctuation from the gold transcription ?

- The model is trained to predict the characters in the gold transcription, but the gold transcription contains punctuation. The model is not trained to predict punctuation, so it is better to remove it from the gold transcription. Also, it is usually not necessary to include special characters in the transcription to understand the meaning of a speech signal.

```
# text processing
import re

special_chars = r'[^a-zA-Zàâäçèéêëîïôœùûüÿ\'\’\s]'

def remove_special_characters(batch):
    batch["sentence"] = re.sub(special_chars, '', batch["sentence"]).lower() + " "
    return batch
```

```
cv_dataset = cv_dataset.map(remove_special_characters)
```

```
Map: 100%  1851/1851 [00:00<00:00, 6521.90 examples/s]
```

```
cv_dataset[:5]['sentence']
```

```
['un vrai travail intéressant va enfin être mené sur ce sujet  ',
 'une réforme profonde est nécessaire  ',
 'pas si nombreuses que ça  ',
 'un comité interministériel du handicap s’est tenu il y a quelques semaines  ',
 'la parole est à monsieur alain ramadier pour soutenir l’amendement numéro cent vingthuit  ']
```

```
# vocabulary

vocab_list = list(set(" ".join(cv_dataset['sentence'])))

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

```
{'é': 0,
 'x': 1,
 'p': 2,
 'j': 3,
 'î': 4,
 'w': 5,
 'o': 6,
 'n': 7,
 'è': 8,
 'c': 9,
 'y': 10,
 'œ': 11,
 'â': 12,
 'ê': 13,
 'd': 14,
 ' ': 15,
 'a': 16,
 's': 17,
 'z': 18,
 'l': 19,
 'q': 20,
 'u': 21,
 'k': 22,
 'g': 23,
 'ë': 24,
 'i': 25,
 'b': 26,
 'v': 27,
 'm': 28,
 'f': 29,
 'e': 30,
 'h': 31,
 'ü': 32,
 'ä': 33,
 "'": 34,
 'à': 35,
 'r': 36,
 'ù': 37,
 'ç': 38,
 '’': 39,
 'ï': 40,
 'û': 41,
 'ô': 42,
 't': 43}
```

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

```
# add an "unknown" and a "padding" token
vocab_dict["[UNK]"] = len(vocab_dict)
vocab_dict["[PAD]"] = len(vocab_dict)
len(vocab_dict)
```

```
46
```

```
# save the vocabulary as a json file
import json
with open('vocab.json', 'w') as vocab_file:
    json.dump(vocab_dict, vocab_file)
```

Tokenizer, Feature extractor, Processor

2. wav2vec2 is often described as an “accoustic model”. Can you explain why ? Why should we also consider a “language model” ?

- While wav2vec2 is primarily focused on the acoustic modeling of speech, a language model is important for capturing the higher-level structure and semantics of the language. By combining the acoustic model with a language model, we can improve the performance of speech recognition systems by incorporating both the acoustic and linguistic information.

```
from transformers import Wav2Vec2CTCTokenizer

# use the json file to instantiate an object of the Wav2Vec2CTCTokenizer class
tokenizer = Wav2Vec2CTCTokenizer("./vocab.json", unk_token="[UNK]", pad_token="[PAD]", word_delimiter_token="|")
```

```
from transformers import Wav2Vec2FeatureExtractor

# the feature extractor is responsible for processing the audio files
feature_extractor = Wav2Vec2FeatureExtractor(feature_size=1, sampling_rate=16000, padding_value=0.0, do_normalize=True, return_attention_mask=False)
```

```
from transformers import Wav2Vec2Processor

# the processor is responsible for processing the transcriptions
processor = Wav2Vec2Processor(feature_extractor=feature_extractor, tokenizer=tokenizer)
```

Audio resampling

```
cv_dataset[0]['audio']
```

```
{'path':
'/root/.cache/huggingface/datasets/downloads/extracted/4aadb4dc625d3b2d15fbb7331f6b404c799ec1433251dc58d39dfa9b67a3d22d/fr_test_0/common_voice_fr_172
 'array': array([ 0.00000000e+00, -1.01153664e-13,  5.07202228e-15, ...,
        -2.62631238e-05, -7.72800286e-06, -4.08383894e-05]),
 'sampling_rate': 48000}
```

```
from scipy.signal import resample

# resample the audio files to 16kHz

def convert_sampling_rate(audio, original_rate, new_rate):
    resampling_ratio = new_rate / original_rate
    return resample(audio, int(resampling_ratio * len(audio)))

def re_sampling_rate(batch):
    new_array = convert_sampling_rate(batch['audio']['array'], batch['audio']['sampling_rate'], 16000)
    batch['resampled_audio'] = {"path": batch['audio']['path'], "array":new_array, "sampling_rate": 16000}
    return batch
```

```
cv_dataset = cv_dataset.map(re_sampling_rate, remove_columns=['audio']).rename_column("resampled_audio", "audio")
```

```
Map: 100%  1851/1851 [00:38<00:00, 59.60 examples/s]
```

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

# check the audio file and its transcription

rand_int = random.randint(0, len(cv_dataset))
print(cv_dataset[rand_int]["sentence"])

audio = cv_dataset[rand_int]["audio"]["array"]
ipd.Audio(data=np.asarray(audio), autoplay=True, rate=16000)
```

```
mais le premier est surtout beaucoup plus cher que le second
```

0:00 / 0:05

```
rand_int = random.randint(0, len(cv_dataset))

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

```
Target text: un chien
Input array shape: (39168,)
Sampling rate: 16000
```

```
# prepare the dataset with resampled audio files

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"]).input_values[0]
    batch["input_length"] = len(batch["input_values"])

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

```
cv_dataset = cv_dataset.map(prepare_dataset)
```

```
Map: 100%  1851/1851 [01:42<00:00, 22.77 examples/s]
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
```

```
# filter all sequences that are longer than 4 seconds out of the dataset

max_input_length_in_sec = 4.0
cv_dataset = cv_dataset.filter(lambda x: x < max_input_length_in_sec * processor.feature_extractor.sampling_rate, input_columns=["input_length"])
```

```
Filter: 100%  1851/1851 [00:00<00:00, 68609.60 examples/s]
```

5. Split the data in a train and a test sets (using a 80-20 split).

```
cv_dataset = cv_dataset.train_test_split(test_size=0.2)
cv_dataset
```

```
DatasetDict({
    train: Dataset({
```

```
        features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio',
'input_values', 'input_length', 'labels'],
        num_rows: 265
    })
    test: Dataset({
        features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio',
'input_values', 'input_length', 'labels'],
        num_rows: 67
    })
})
```

Data collator

```
# data collator

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.PaddingStrategy`, `optional`, defaults to :obj:`True`):
            Select a strategy to pad the returned sequences (according to the model's padding side and padding index)
            among:
            * :obj:`True` or :obj:`'longest'`: Pad to the longest sequence in the batch (or no padding if only a single
              sequence if provided).
            * :obj:`'max_length'`: Pad to a maximum length specified with the argument :obj:`max_length` or to the
              maximum acceptable input length for the model if that argument is not provided.
            * :obj:`False` or :obj:`'do_not_pad'` (default): No padding (i.e., can output a batch with sequences of
              different lengths).
    """

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

    def __call__(self, features: List[Dict[str, Union[List[int], torch.Tensor]]]) -> Dict[str, torch.Tensor]:
        # 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), -100)

        batch["labels"] = labels

        return batch
```

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

WER metric

3. What is WER and how is it defined ?

- The Word Error Rate (WER) is a metric used to evaluate the performance of a speech recognition system. It is defined as the minimum number of insertions, deletions, and substitutions required to transform the hypothesis into the reference transcription, divided by the number of words in the reference transcription.

```
from evaluate import load

# evaluation metric

wer_metric = load('wer')
```

```
Downloading builder script: 100%  4.49k/4.49k [00:00<00:00, 367kB/s]
```

```
def compute_metrics(pred):
    pred_logits = pred.predictions
    pred_ids = np.argmax(pred_logits, axis=-1)

    pred.label_ids[pred.label_ids == -100] = processor.tokenizer.pad_token_id

    pred_str = processor.batch_decode(pred_ids)
    # we do not want to group tokens when computing the metrics
    label_str = processor.batch_decode(pred.label_ids, group_tokens=False)

    wer = wer_metric.compute(predictions=pred_str, references=label_str)

    return {"wer": wer}
```

Model training

4. What hyper-parameters can/should be set to control training ?

- The hyper-parameters that can be set to control training are the learning rate, the batch size, the number of epochs, the weight decay, the warmup steps, the gradient clipping, the dropout rate, the attention dropout rate, the layer dropout rate, the number of layers, the number of attention heads, the hidden size, the intermediate size, the number of training etc.

```
from transformers import Wav2Vec2ForCTC

# load the pretrained Wav2Vec2 checkpoint
model = Wav2Vec2ForCTC.from_pretrained(
    "facebook/wav2vec2-large-xlsr-53-french",
    ctc_loss_reduction="mean",
    pad_token_id=processor.tokenizer.pad_token_id,
    vocab_size=len(processor.tokenizer),
    ignore_mismatched_sizes=True)
```

```
config.json: 100%                                                    1.29k/1.29k [00:00<00:00, 115kB/s]
pytorch_model.bin: 100%                                              1.26G/1.26G [02:04<00:00, 11.0MB/s]
Some weights of the model checkpoint at facebook/wav2vec2-large-xlsr-53-french were not used when
- This IS expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model trained
- This IS NOT expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model that
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
- lm_head.weight: found shape torch.Size([49, 1024]) in the checkpoint and torch.Size([48, 1024])
- lm_head.bias: found shape torch.Size([49]) in the checkpoint and torch.Size([48]) in the model
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
```

```
model.freeze_feature_encoder()
```

```
from transformers import TrainingArguments

training_args = TrainingArguments(
  output_dir="./",
  group_by_length=True,
  per_device_train_batch_size=4,
  evaluation_strategy="steps",
  num_train_epochs=20,
  gradient_checkpointing=True,
  save_steps=200,
  eval_steps=200,
  logging_steps=200,
  learning_rate=1e-4,
  weight_decay=0.005,
  warmup_steps=1000,
  save_total_limit=2,
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1474: FutureWarning: `evaluation_strategy` is deprecated and will be removed in
  warnings.warn(
```

```
from transformers import Trainer

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

```
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
```

[1340/1340 18:50, Epoch 20/20]

Step	Training Loss	Validation Loss	Wer
200	3.046600	2.964749	1.000000
400	2.942100	2.830410	1.000000
600	2.264700	0.911038	0.653226
800	0.980300	0.591548	0.540323
1000	0.662400	0.527328	0.505376
1200	0.485500	0.508780	0.491935

```
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
TrainOutput(global_step=1340, training_loss=1.5882406747163231, metrics={'train_runtime':
1131.1306, 'train_samples_per_second': 4.686, 'train_steps_per_second': 1.185, 'total_flos':
5.4241586704825344e+17, 'train_loss': 1.5882406747163231, 'epoch': 20.0})
```

```
def map_to_result(batch):
  with torch.no_grad():
    input_values = torch.tensor(batch["input_values"], device="cuda").unsqueeze(0)
    logits = model(input_values).logits

  pred_ids = torch.argmax(logits, dim=-1)
  batch["pred_str"] = processor.batch_decode(pred_ids)[0]
  batch["text"] = processor.decode(batch["labels"], group_tokens=False)

  return batch
```

```
results = cv_dataset["test"].map(map_to_result, remove_columns=cv_dataset["test"].column_names)
print("Test WER: {:.3f}".format(wer_metric.compute(predictions=results["pred_str"], references=results["text"])))
```

```
Parameter 'function'=<function map_to_result at 0x7dbfd300a4d0> of the transform datasets.arrow_d
WARNING:datasets.fingerprint:Parameter 'function'=<function map_to_result at 0x7dbfd300a4d0> of t
```

Map: 100% 67/67 [00:04<00:00, 14.25 examples/s]

```
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:91: UserWarning: None of the in
  warnings.warn(
Test WER: 0.634
```

```
results[1]
```

```
{'pred_str': 'remplissez le s’il vous plait',
 'text': "remplissezle s'il vous plait"}
```

```
def predict(batch, model):
    model = model.cuda()
    with torch.no_grad():
        input_values = torch.tensor(batch["input_values"], device="cuda").unsqueeze(0)
        logits = model(input_values).logits

    pred_ids = torch.argmax(logits, dim=-1)
    batch["pred_str"] = processor.batch_decode(pred_ids)[0]
    batch["text"] = processor.decode(batch["labels"], group_tokens=False)
    model.to('cpu')
    input_values.to('cpu')

    return batch
```

Fine-tuning

6. Calculate the learning and testing errors obtained for 3 different sets of hyper-parameters. What can you conclude about the model's ability to generalize ?

- Epoch: 20, Weight Decay: 0.005; WER: 0.653
- Epoch: 20, Weight Decay: 0.05; WER: 0.616
- Epoch: 30, Weight Decay: 0.05; WER: 0.589

The model's ability to generalize improves with higher weight decay and more training epochs, as evidenced by the progressively lower WER values. Therefore, the best hyperparameter combination in this case is 30 epochs with a weight decay of 0.05, which achieves the lowest WER of 0.589.

```
def finetune_hyper(num_train_epochs=20, learning_rate=1e-4, weight_decay=0.005, warmup_steps=1000, train_set=cv_dataset["train"]):
    model = Wav2Vec2ForCTC.from_pretrained(
        "facebook/wav2vec2-large-xlsr-53-french",
        ctc_loss_reduction="mean",
        pad_token_id=processor.tokenizer.pad_token_id,
        vocab_size=len(processor.tokenizer),
        ignore_mismatched_sizes=True
    )

    model.freeze_feature_encoder()

    training_args = TrainingArguments(
    output_dir="./",
    group_by_length=True,
    per_device_train_batch_size=8,
    evaluation_strategy="steps",
    num_train_epochs=num_train_epochs,
    gradient_checkpointing=True,
    save_steps=500,
    eval_steps=500,
    logging_steps=500,
    learning_rate=learning_rate,
    weight_decay=weight_decay,
    warmup_steps=warmup_steps,
    save_total_limit=2,
    )

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

    trainer.train()

    return model.to('cpu')

def evaluate(model, test_set=cv_dataset["test"]):
    results = test_set.map(lambda batch : predict(batch, model), remove_columns=test_set.column_names)
    return wer_metric.compute(predictions=results["pred_str"], references=results["text"])
```

```
# epoch : 20, weight decay : 0.005
model = finetune_hyper(num_train_epochs=20, learning_rate=1e-4, weight_decay=0.005, warmup_steps=100)
wer1 = evaluate(model)
print(wer1) # 0.6532258064516129
```

숨겨진 출력 표시

```
# epoch : 20, weight decay : 0.05
model = finetune_hyper(num_train_epochs=20, learning_rate=1e-4, weight_decay=0.05, warmup_steps=100)
wer2 = evaluate(model)
print(wer2) # 0.6155913978494624
```

```
Some weights of the model checkpoint at facebook/wav2vec2-large-xlsr-53-french were not used when
- This IS expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model trained
- This IS NOT expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model that
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
- lm_head.weight: found shape torch.Size([49, 1024]) in the checkpoint and torch.Size([48, 1024])
- lm_head.bias: found shape torch.Size([49]) in the checkpoint and torch.Size([48]) in the model
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1474: FutureWarning: `evalu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
```

[680/680 15:26, Epoch 20/20]

Step	Training Loss	Validation Loss	Wer
500	3.439900	0.559807	0.537634

```
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
```

Map: 100% 67/67 [00:44<00:00, 1.53 examples/s]

```
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:91: UserWarning: None of the in
  warnings.warn(
0.6155913978494624
```

```
# epoch : 30, weight decay : 0.05
model = finetune_hyper(num_train_epochs=30, learning_rate=1e-4, weight_decay=0.05, warmup_steps=100)
wer3 = evaluate(model)
print(wer3) # 0.5887096774193549
```

```
Some weights of the model checkpoint at facebook/wav2vec2-large-xlsr-53-french were not used when
- This IS expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model trained
- This IS NOT expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model that
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec
- lm_head.weight: found shape torch.Size([49, 1024]) in the checkpoint and torch.Size([48, 1024])
- lm_head.bias: found shape torch.Size([49]) in the checkpoint and torch.Size([48]) in the model
You should probably TRAIN this model on a down-stream task to be able to use it for predictions a
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1474: FutureWarning: `evalu
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
```

[1020/1020 23:19, Epoch 30/30]

Step	Training Loss	Validation Loss	Wer
500	3.620400	0.612926	0.545699
1000	0.444400	0.478746	0.470430

```
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156:
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.c
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/conv.py:306: UserWarning: Plan failed wi
  return F.conv1d(input, weight, bias, self.stride,
```

Map: 100% 67/67 [00:44<00:00, 1.53 examples/s]

```
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:91: UserWarning: None of the in
  warnings.warn(
0.5887096774193549
```

7. Is the data-set gender-balanced ? Why is this important ?

- The data-set is not gender-balanced. This is important because the model might be biased towards the majority class.

```
# split the test set into male and female

male_masculine = np.arange(len(cv_dataset["test"]))[np.array(cv_dataset["test"]['gender']) == 'male_masculine']
test_m = cv_dataset["test"].select(male_masculine)
female_feminine = np.arange(len(cv_dataset["test"]))[np.array(cv_dataset["test"]['gender']) == 'female_feminine']
test_f = cv_dataset["test"].select(female_feminine)
```

```
print(test_m)
print(test_f)
```

```
Dataset({
    features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio', 'input_
    num_rows: 53
})
Dataset({
    features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio', 'input_
    num_rows: 14
})
```

```
# wer for male speakers
print(evaluate(model, test_m))

# wer for female speakers
print(evaluate(model, test_f))
```

```
Map: 100%                                    53/53 [00:34<00:00,  1.53 examples/s]
0.5813148788927336
Map: 100%                                    14/14 [00:09<00:00,  1.50 examples/s]
0.5783132530120482
```

8. Keeping the test set unchanged, train a model on 50% of the train set (with the best set of hyper-parameters). Report the test error distinguishing male and female speakers.

```
# split the train set into male and female
male_masculine = np.arange(len(cv_dataset["train"]))[np.array(cv_dataset["train"]['gender']) == 'male_masculine']
train_m = cv_dataset["train"].select(male_masculine)
female_feminine = np.arange(len(cv_dataset["train"]))[np.array(cv_dataset["train"]['gender']) == 'female_feminine']
train_f = cv_dataset["train"].select(female_feminine)
```

```
print(train_m)
print(train_f)
```

```
Dataset({
    features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio', 'input_
    num_rows: 213
})
Dataset({
    features: ['client_id', 'path', 'sentence', 'up_votes', 'down_votes', 'age', 'gender', 'accent', 'locale', 'segment', 'variant', 'audio', 'input_
    num_rows: 52
})
```

9. Train a model considering only female speakers and another model considering only male speakers. Report the test error distinguishing male and female speakers. What can you conclude ?

The difference in WER between male (63.67%) and female (69.88%) speakers suggests a gender bias in the model's performance. The model generalizes poorly to female voices, likely due to the lack of exposure to female speech data during training. This discrepancy highlights the importance of having a diverse training dataset that includes samples from different genders to ensure better generalization and fairness.

```
# train a model for male speakers
model = finetune_hyper(num_train_epochs=30, learning_rate=1e-4, weight_decay=0.05, warmup_steps=800, train_set=train_m)
# wer for male speakers
print(evaluate(model, test_m))  # 0.6366782006920415
# wer for female speakers
print(evaluate(model, test_f))  # 0.6987951807228916
```

```
Some weights of the model checkpoint at facebook/wav2vec2-large-xlsr-53-french were not used when initializing Wav2Vec2ForCTC: ['wav2vec2.encoder.pos
- This IS expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model trained on another task or with another architecture (e.g. i
- This IS NOT expected if you are initializing Wav2Vec2ForCTC from the checkpoint of a model that you expect to be exactly identical (initializing a
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec2-large-xlsr-53-french and are newly initialized: ['
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Some weights of Wav2Vec2ForCTC were not initialized from the model checkpoint at facebook/wav2vec2-large-xlsr-53-french and are newly initialized bec
- lm_head.weight: found shape torch.Size([49, 1024]) in the checkpoint and torch.Size([48, 1024]) in the model instantiated
- lm_head.bias: found shape torch.Size([49]) in the checkpoint and torch.Size([48]) in the model instantiated
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1474: FutureWarning: `evaluation_strategy` is deprecated and will be removed in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156: UserWarning: `as_target_processor` is deprecated and
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.checkpoint: the use_reentrant parameter should be pas
  warnings.warn(
[810/810 18:34, Epoch 30/30]

| Step | Training Loss | Validation Loss | Wer |
|---|---|---|---|
| 500 | 6.646600 | 2.865411 | 1.000000 |

/usr/local/lib/python3.10/dist-packages/transformers/models/wav2vec2/processing_wav2vec2.py:156: UserWarning: `as_target_processor` is deprecated and
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:464: UserWarning: torch.utils.checkpoint: the use_reentrant parameter should be pas
  warnings.warn(
Map: 100%                                    53/53 [00:34<00:00,  1.53 examples/s]
/usr/local/lib/python3.10/dist-packages/torch/utils/checkpoint.py:91: UserWarning: None of the inputs have requires_grad=True. Gradients will be None
```