Spanish Text-to-Speech with SpeechT5 + VoxPopuli + Own Voice Synthesis!

Miya Ding, Trevor Freed, Heqi Qiao

HUGGING FACE

Procedure Overview

1. Extract a small dataset in Spanish.
2. Fine-tune SpeechT5 with Spanish audio-text pairs.
3. Inference time generate speech from text in Spanish.
4. Add Miya’s voice dataset, extract embeddings, inference.

Core Components

- SpeechT5
  - Pre-existing transformer based architecture
  - Pre-trained on speech and text data, joint representation
  - Downstream: ex. Generates natural sounding speech from text in English
- SpeechBrain
  - Speaker Embeddings
- **VoxPopuli**
  - Multi-language audio-transcription paired dataset, we extracted Spanish, preprocess text
  - Multi-speaker, around 5 minute of audio/speaker along with text
- **Own Voice Synthesis**
  - Convert Miya's Voice and Text into matching format

```
# Load the CSV file
csv_file_path = 'SpanishVoiceWav/normalized_text.csv'  # Replace with the actual path to your CSV file
df = pd.read_csv(csv_file_path)

# Directory containing the WAV files
directory_path = 'SpanishVoiceWav'

# Get a sorted list of all WAV files in the directory
wav_files = sorted([f for f in os.listdir(directory_path) if f.endswith('.wav')])

# Initialize lists for each column
audio_id_list = []
language_list = []
audio_list = []
raw_text_list = []
normalized_text_list = []
gender_list = []
speaker_id_list = []
is_gold_transcript_list = []
accent_list = []

# Define the features schema with ClassLabel for language
features = Features({
    'audio_id': Value(dtype='string'),
    'language': ClassLabel(names=['en', 'de', 'fr', 'es', 'pl', 'it', 'ro', 'hu', 'cs', 'nl', 'fi', 'hr', 'sk', '
    'audio': Audio(sampling_rate=16000),
    'raw_text': Value(dtype='string'),
    'normalized_text': Value(dtype='string'),
    'gender': Value(dtype='string'),
    'speaker_id': Value(dtype='string'),
    'is_gold_transcript': Value(dtype='bool'),
    'accent': Value(dtype='string')
})

# Finding the index for 'es' in the ClassLabel
spanish_index = features['language'].str2int('es')

# Loop through each sorted file in the directory
for idx, filename in enumerate(wav_files):
    file_path = os.path.join(directory_path, filename)

    # Load the audio file with librosa
    audio_array, sr = librosa.load(file_path, sr=16000, mono=True)

    # Store the audio array in the audio_list
    audio_list.append({'array': audio_array, 'sampling_rate': sr})
```

Links

SpeechT5: https://huggingface.co/learn/audio-course/chapter6/pre-trained_models

SpeechBrain: https://huggingface.co/speechbrain/spkrec-xvect-voxceleb

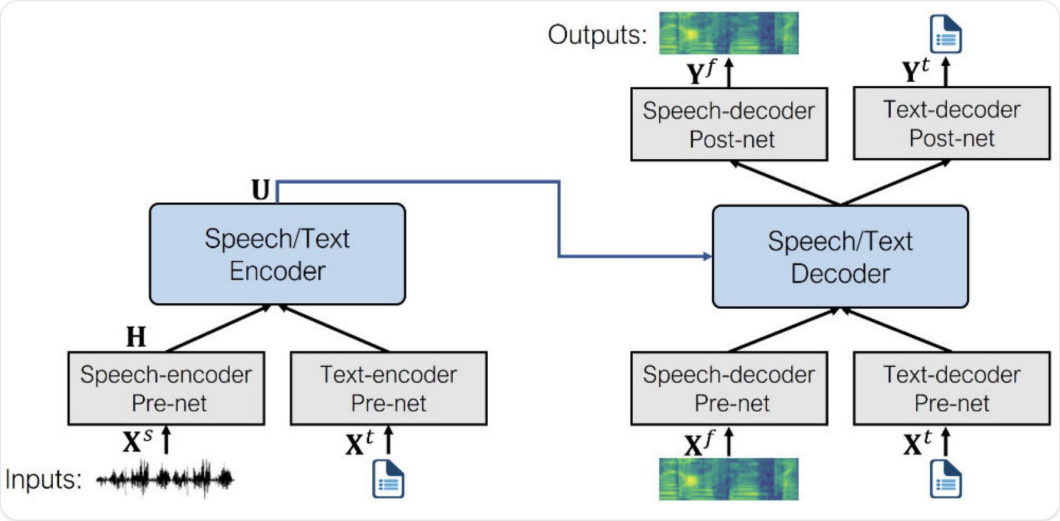
TTS Code Adapted from: https://huggingface.co/docs/transformers/en/tasks/text-to-speech

VoxPopuli Dataset: https://huggingface.co/datasets/facebook/voxpopuli

Main notebook: https://github.com/HQQHQ/FinetuneSpeechT5-Spanish/blob/main/TTS.ipynb

Play & Evaluation (for Richa): https://github.com/HQQHQ/FinetuneSpeechT5-Spanish/blob/main/Play_and_Evaluate.ipynb

SpeechT5 Model Architecture



SpeechT5 learns “joint contextual representations for speech and text data via a shared encoder-decoder structure”

What are Speaker Embeddings?

spkrec-**xvect**-voxceleb model from SpeechBrain, 512 element vector, SpeechT5 also pre-trained on x-vectors.
A numerical representation that captures the unique vocal characteristics of a speaker, such as tone, pitch, accent, and other speech-specific features that are distinct to that individual, **almost** irrespective of the linguistic content.

For example, we extracted Miya voice's embeddings from the SpeechBrain model

Potential Issues:

Lack of Language Specificity: significant differences in pronunciation and intonation. Result in speech features (embeddings) that are not accurate or representative.

```python
import os
import torch
import speechbrain
from speechbrain.pretrained import EncoderClassifier

spk_model_name = "speechbrain/spkrec-xvect-voxceleb"

device = "cuda" if torch.cuda.is_available() else "cpu"
speaker_model = EncoderClassifier.from_hparams(
    source=spk_model_name,
    run_opts={"device": device},
    savedir=os.path.join("/tmp", spk_model_name),
)


def create_speaker_embedding(waveform):
    with torch.no_grad():
        speaker_embeddings = speaker_model.encode_batch(torch.tensor(waveform))
        speaker_embeddings = torch.nn.functional.normalize(speaker_embeddings, dim=2)
        speaker_embeddings = speaker_embeddings.squeeze().cpu().numpy()
    return speaker_embeddings
```

Zero-Shot or Non-Zero-Shot Learning?

Both!

1. **Zero-Shot:** Using SpeechBrain model pre-trained on English to directly extract Spanish audio-embeddings.

2. **Non-Zero-Shot:** Later fine-tuned SpeechT5 with Spanish data

Fine-Tuning SpeechT5!

Adapting to New Data: Fine-tuning adjusts the pre-trained model's weights so that it can better understand and generate the Spanish language audio characteristics.

Learning Task-Specific Features: it also learns to better perform specific TTS task.

Improving Model Generalization: generalize better across various accents, dialects, and speaking styles within the Spanish-speaking population.

Fine-Tuning

```python
from transformers import Seq2SeqTrainingArguments

training_args = Seq2SeqTrainingArguments(
    output_dir="speecht5_finetuned_voxpopuli_nl",
    per_device_train_batch_size=4,
    gradient_accumulation_steps=8,
    learning_rate=1e-5,
    warmup_steps=500,
    max_steps=4000,
    gradient_checkpointing=True,
    fp16=True,
    evaluation_strategy="steps",
    per_device_eval_batch_size=2,
    save_steps=1000,
    eval_steps=1000,
    logging_steps=25,
    report_to=["tensorboard"],
    load_best_model_at_end=True,
    greater_is_better=False,
    label_names=["labels"],
    push_to_hub=True,
)
```

```python
from transformers import Seq2SeqTrainer

trainer = Seq2SeqTrainer(
    args=training_args,
    model=model,
    train_dataset=dataset["train"],
    eval_dataset=dataset["test"],
    data_collator=data_collator,
    tokenizer=processor,
)
```

```
max_steps is given, it will override any value given in num_t
```

```python
trainer.train()
```

```
/home/zeus/miniconda3/envs/cloudspace/lib/python3.10/site-pac
  warnings.warn(
```

[4000/4000 2:23:16, Epoch 17/18]

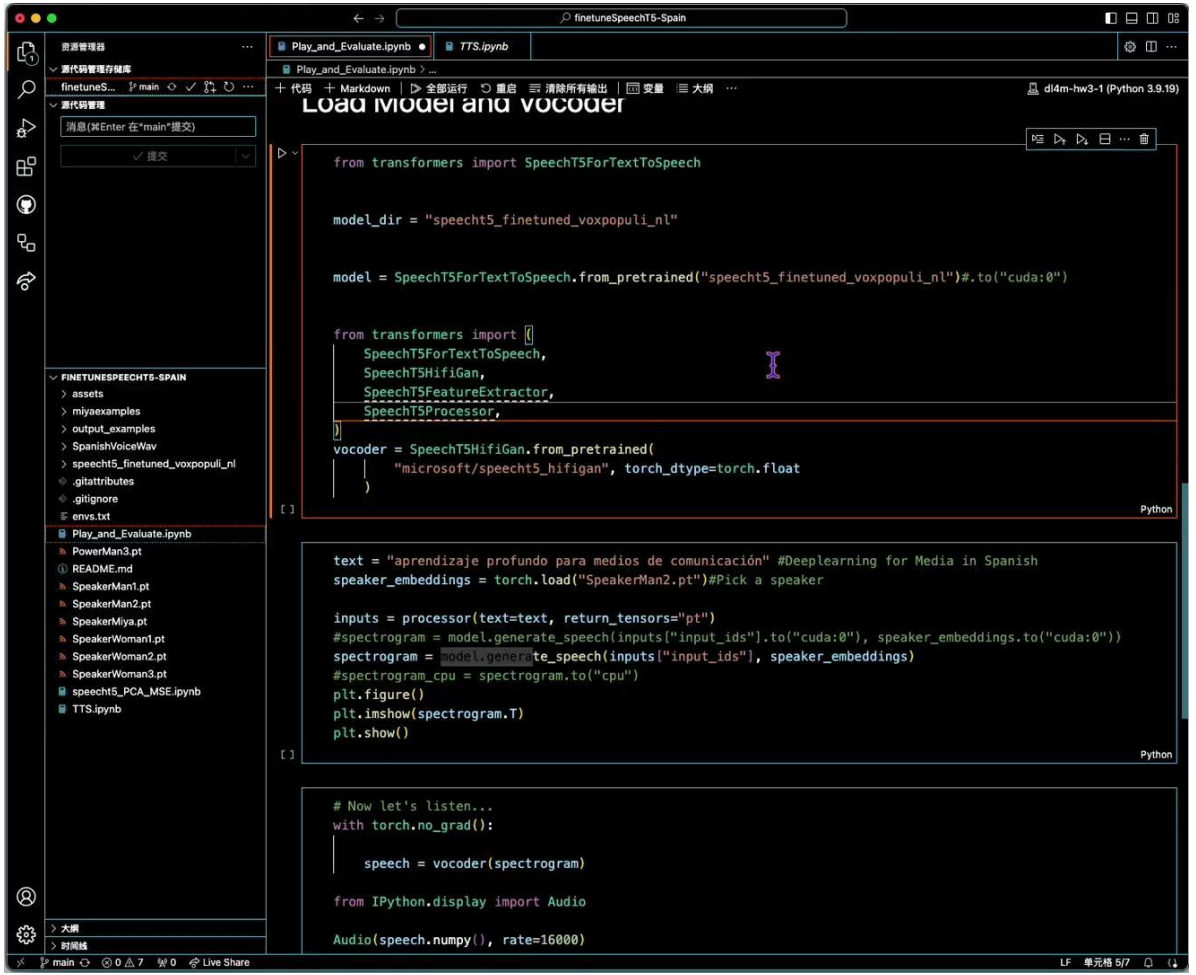
Step	Training Loss	Validation Loss
1000	0.506000	0.464025
2000	0.485400	0.450527
3000	0.480000	0.447352
4000	0.481400	0.445818

Using the Seq2SeqTrainingArguments class from the Transformers library to set up the training configurations.

Load and store the best model at the end

Upload the model file using the git lfs

Inference!


finetuneSpeechT5-Spain
Play_and_Evaluate.ipynb
TTS.ipynb
Load model and vocoder
from transformers import SpeechT5ForTextToSpeech
model_dir = "speecht5_finetuned_voxpopuli_nl"
model = SpeechT5ForTextToSpeech.from_pretrained("speecht5_finetuned_voxpopuli_nl")#.to("cuda:0")
from transformers import (
SpeechT5ForTextToSpeech,
SpeechT5HifiGan,
SpeechT5FeatureExtractor,
SpeechT5Processor,
)
vocoder = SpeechT5HifiGan.from_pretrained(
"microsoft/speecht5_hifigan", torch_dtype=torch.float
)
Python
text = "aprendizaje profundo para medios de comunicación" #Deeplearning for Media in Spanish
speaker_embeddings = torch.load("SpeakerMan2.pt")#Pick a speaker
inputs = processor(text=text, return_tensors="pt")
#spectrogram = model.generate_speech(inputs["input_ids"].to("cuda:0"), speaker_embeddings.to("cuda:0"))
spectrogram = model.generate_speech(inputs["input_ids"], speaker_embeddings)
#spectrogram_cpu = spectrogram.to("cpu")
plt.figure()
plt.imshow(spectrogram.T)
plt.show()
Python
Now let's listen...
with torch.no_grad():
speech = vocoder(spectrogram)
from IPython.display import Audio
Audio(speech.numpy(), rate=16000)
FINETUNESPEECHT5-SPAIN
assets
miyaexamples
output_examples
SpanishVoiceWav
speecht5_finetuned_voxpopuli_nl
.gitattributes
.gitignore
envs.txt
Play_and_Evaluate.ipynb
PowerMan3.pt
README.md
SpeakerMan1.pt
SpeakerMan2.pt
SpeakerMiya.pt
SpeakerWoman1.pt
SpeakerWoman2.pt
SpeakerWoman3.pt
speecht5_PCA_MSE.ipynb
TTS.ipynb

Novelty

As mentioned earlier, we created our own dataset of Miya's Spanish voice recordings and corresponding texts, following the same structure as the original dataset.

We managed to extract Miya's speaker embeddings and generate audio that resembles her voice from any simple given texts. Here are some examples:

Me llamo Trevor
(My name is Trevor)

Hugging Face
(Huggingface)

Me llamo Lixin
(My name is Lixin)

Lixin gustan las chicas
(Lixin likes girls)

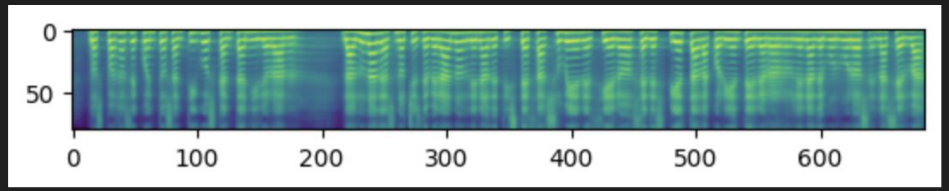
Evaluation: Fine-tuning

- Lectura en Español, by José Martí
  - Made up by ChatGPT??
  - Spanish analogy to The Rainbow Passage 🌈
- Over 250+ characters, performance declines 🫠
- Under 250 characters, good enough for Google Translate 🤓

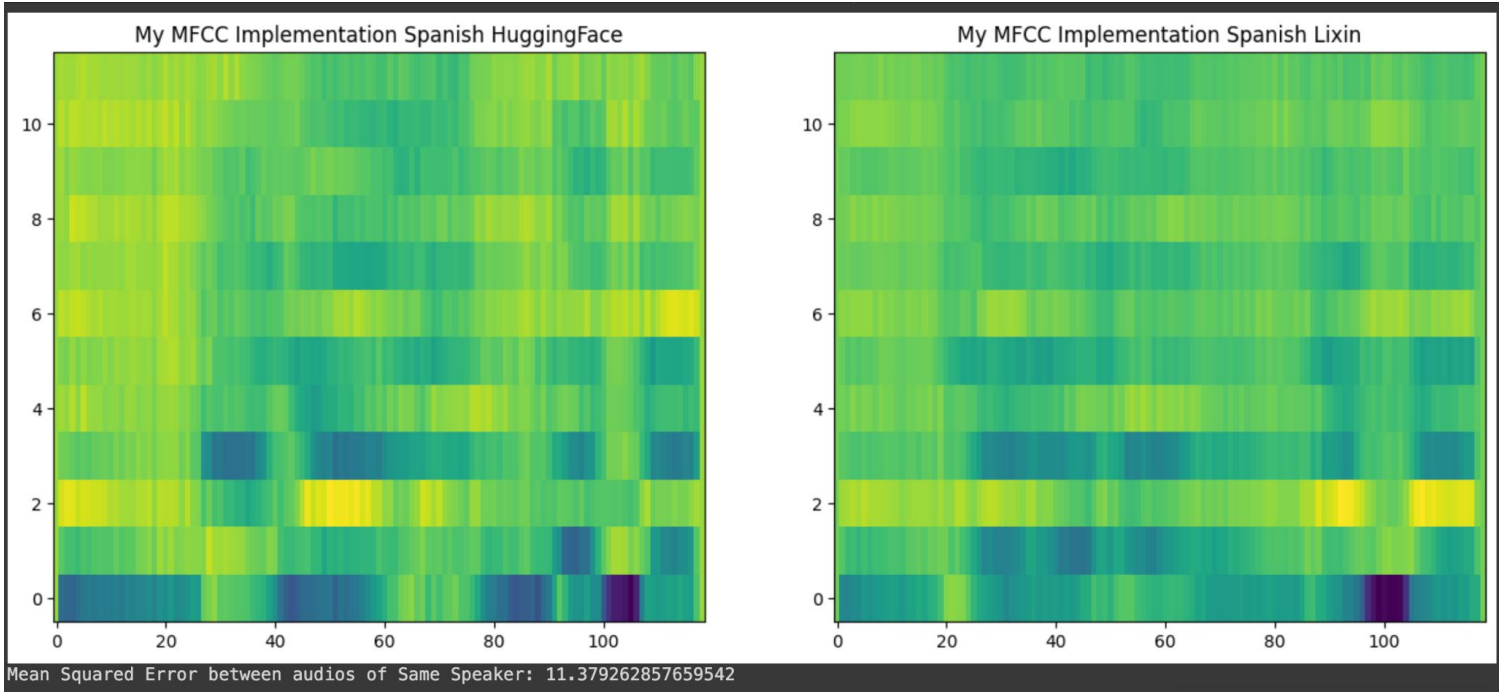
Evaluation: Fine-tuning

```
text = "Los pájaros de cuentos antiguos cantan y lloran. La luna de las noches de verano, llena de tristeza, sube en el cielo.\
Se despierta la corriente del río, llena de murmuros, en la selva." # Lectura en Español by José Martí excerpt

speaker_embeddings = torch.load("SpeakerWoman1.pt") # Pick a speaker

inputs = processor(text=text, return_tensors="pt")

spectrogram = model.generate_speech(inputs["input_ids"], speaker_embeddings)

plt.figure()
plt.imshow(spectrogram.T)
plt.show()
```

✓ 2.1s



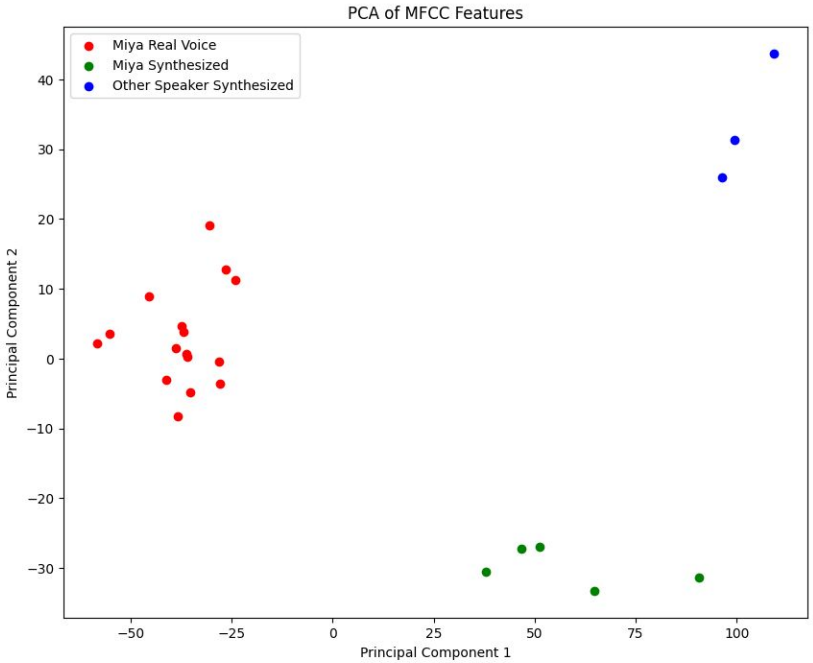
Evaluation: MFCC across same speaker' audio



Evaluation: Own Voice Synthesis

- For 2-3 word inputs, pretty good!
- For longer inputs…. 😬

Visual Evaluation: Own Voice Synthesis



Centroid MSE Distances: Miya Real v.s. Synth: 5102.77, Miya Real-Other Speaker Synth: 10129.69, Miya Synth-Other Synth: 2962.51

Feedback from Proposal

1) Evaluation method(s)?
    a) Use Google Voice Recognition Model 👍🏻
    b) Didn’t conduct Mean Opinion Score study

2) Bonus Personalization Objective: embed our own voices? YEAH!

Limitations

- The model needs further development to effectively extract and generalize speaker embeddings. Some are not audible at all.
  MegaTTS2, ByteDance
- The small scale(16 clips) and non-fluency of our Spanish speaking datasets limit the model's capability to only simple sentence generation, not suitable for complex tasks.
- Audible generated speech contains electrical-like noises, indicating a need for significant improvement in sound quality.