Task 1

Fine-tuning TTS for English with a Focus on Technical Vocabulary

-by Tejasva Maurya

1. Introduction

This report evaluates the performance of the fine-tuned SpeechT5 model against established pre-trained models like Mozilla TTS and Coqui TTS. We will assess its proficiency in pronouncing technical terms, analyze Mean Opinion Scores (MOS) for user satisfaction, and measure inference speed, showcasing the model's potential for producing intelligible and expressive speech in specialized contexts.

2. Dataset Description

This dataset is a custom-built audio dataset, primarily created by recording my own voice to capture technical jargon and specific sentences relevant to speech recognition and text-to-speech applications. To further enhance its robustness and adaptability, I have integrated additional publicly available datasets.

Data Source:

- **Primary data:** Custom self-recorded audio, focusing on technical jargon and specialized phrases.
- **Supplementary data:** Additional datasets from Hugging Face and the NPTEL2020 Indian English Speech Dataset (from AI4Bharat), sourced from GitHub (https://github.com/AI4Bharat/NPTEL2020-Indian-English-Speech-Dataset?tab=readme-ov-file)

Details: Combination of multiple datasets, including custom recordings and public datasets. Includes downloads and sample data (pure-set).

Customization: The dataset has been fine-tuned by adding a few personally recorded audio files, enhancing its relevance for technical speech recognition tasks.

This combined dataset is designed to support speech recognition models in accurately handling technical vocabulary, especially in the context of Indian-English accents Content.

Examples:

"API stands for Application Programming Interface" → Processed_DATASET/data/processed_full_forms/full_forms-06.wav

"SaaS stands for Software as a Service" → Processed_DATASET/data/processed_full_forms/full_forms-07.wav

"We trained the neural network using backpropagation to minimize the loss function." → Processed_DATASET/data/processed_Sentences/Sentences-01.wav

3. Fine-Tuning Process of the SpeechT5 TTS Model

The fine-tuning process consisted of several stages, including data preparation, text normalization, speaker embedding extraction, model setup, and training. Here's a breakdown of each step:

1. **Initialization and Setup**
   The Hugging Face SpeechT5ForTextToSpeech model and SpeechT5Processor were

initialized using the pre-trained "microsoft/speecht5_tts" checkpoint. This model, designed for text-to-speech conversion, was selected for fine-tuning on the target dataset.

2. **Loading and Processing the Dataset**
   The dataset, stored in a CSV file, was loaded using the datasets library. It contained both audio samples and corresponding text transcriptions. All audio data was resampled to a consistent 16 kHz sampling rate to maintain uniformity across the dataset.
3. **Text Normalization**
   Text input was standardized by normalizing it—converting to lowercase, removing unnecessary punctuation (excluding apostrophes), and eliminating extra spaces. Additionally, numeric characters and underscores were replaced with their respective word forms (e.g., "0" to "zero") to enhance clarity in speech generation, especially for technical terms.
4. **Speaker Embedding Extraction**
   Speaker embeddings were extracted using SpeechBrain's pre-trained EncoderClassifier model (speechbrain/spkrec-xvect-voxceleb) to capture speaker-specific characteristics. These embeddings were added alongside the audio and text data to allow the model to mimic specific speaker styles.
5. **Data Filtering and Splitting**
   Long sequences with more than 200 tokens were filtered out to avoid complications during training. The dataset was then split into training and testing sets with a 90-10 ratio.
6. **Custom Data Collator**
   A custom data collator was implemented to handle the padding of input text, audio, and speaker embeddings. Special care was taken to correctly mask padded tokens to ensure they didn't affect loss calculation during training.
7. **Training Setup**
   The model was configured with specific hyperparameters, including a learning rate of 0.0001, a batch size of 4, gradient accumulation steps of 8, and gradient checkpointing for memory optimization. Training was capped at 1,000 steps with evaluation every 100 steps, and the best model was pushed to the Hugging Face Hub.
8. **Fine-Tuning the Model**
   During training, the model processed both text and audio data, using the speaker embeddings to generate speech that reflected the specific characteristics of each speaker.
9. **Post-Training Inference and Speech Generation**
   After fine-tuning, the model was used for speech generation from text. The SpeechT5HifiGan vocoder was employed to convert the model's output into high-fidelity waveforms.

This multi-step process improved the model's ability to generate high-quality, speaker-specific speech from text inputs, making it suitable for technical and specialized use cases.

4. Training hyperparameters

The following hyperparameters were used during training:

- learning_rate: 0.0001
- train_batch_size: 4
- eval_batch_size: 2
- seed: 42

- gradient_accumulation_steps: 8
- total_train_batch_size: 32
- optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- lr_scheduler_type: linear
- lr_scheduler_warmup_steps: 100
- training_steps: 1000
- mixed_precision_training: Native AMP

5. Evaluation Results

Inference Time (in Seconds)

Finetuned Model	**Base Model**
3.1275	4.5659

Training results

Training Loss	**Epoch**	**Step**	**Validation Loss**
0.6122	0.3168	100	0.5289
0.5468	0.6337	200	0.4885
0.5207	0.9505	300	0.4745
0.5086	1.2673	400	0.4729
0.5012	1.5842	500	0.4638
0.4982	1.9010	600	0.4564
0.4888	2.2178	700	0.4528
0.4862	2.5347	800	0.4515
0.4866	2.8515	900	0.4454
0.4753	3.1683	1000	0.4451

Technical Terms List and Pronunciation Outputs

Technical Term	**Pronunciation Output (**in Pronunciations Output folder**)**
Authentication	authentication.wav
BERT	BERT.wav
Boosting	Boosting.wav
Cache	Cache.wav
Classification	classification.wav

Containerization	Containerization.wav
CUDA	CUDA.wav
GPU	GPU.wav
Gradient	Gradient.wav
GraphQL	GraphQL.wav
Implementing	implementing.wav
Infrastructure	Infrastructure.wav
Kubernetes	Kubernetes.wav
Load Balancing	Load Balancing.wav
LSTM	LSTM.wav
Microservices Architecture	Microservices Architecture.wav
MongoDB	MongoDB.wav
Multithreading	Multithreading.wav
Neural Network	Neural Network.wav
Non-linearity	non-linearity.wav
NoSQL	NoSQL.wav
OAuth	OAuth.wav
Oversampling	Oversampling.wav
ROC Curve	ROC Curve.wav
Scalability	Scalability.wav
SVM	SVM.wav
TPU	TPU.wav
Transformers	Transformers.wav
VPN	VPN.wav
Word2Vec	Word2Vec.wav
XGBoost	XGBoost.wav