Egyptian Dialect Speech-to-Text System using Wav2Vec2

This document provides a comprehensive guide to the Speech-to-Text (STT) system designed specifically for transcribing Egyptian dialect speech. The system leverages the power of Wav2Vec2, a pre-trained model, to achieve accurate and reliable transcriptions.

Project Goal

The primary aim of this project is to develop a robust STT system that excels in recognizing and converting Egyptian dialect speech into text format. This is achieved by fine-tuning the Wav2Vec2 model on a large dataset of Egyptian dialect recordings.

Installation Guide

Prerequisites:

- Python 3.11
- Jupyter Notebook
- Essential Libraries: datasets, torchaudio, transformers, pandas, numpy, librosa, torch, etc.
- (Optional) Kaggle account for TPU access

Steps:

1. **Download the Code:** (Instructions on obtaining the code omitted)
2. **Install Dependencies:** Use pip install -r requirements.txt in your terminal.
3. **Data Acquisition:**
    - Download the dataset and extract it into designated folders (train and adapt).

Data and Resources

Dataset Details:

- **Name:** MTC-ASR-Dataset-16K
- **Content:** 100 hours of Egyptian dialect speech recordings
- **Testing Data:**

    - Duration: 3 hours
    - File format: 1726 audio files (16KHz WAV)
    - Recording environment: Clean and Noisy Audio
- **Dataset Structure:**
    - train/: Contains training audio data (WAV files)
    - adapt/: Contains adaptation audio data (WAV files)
    - train.csv: CSV file with wav_id and transcription columns for training data
    - adapt.csv: CSV file with wav_id and transcription columns for adaptation data

Data Access:

- Download the dataset from the provided link and extract it into the designated folders (train and adapt).

Understanding the Code

Code Organization:

- Preprocess_&_Prepare_Data.ipynb: Handles data preprocessing and feature extraction.
- Train.ipynb: Defines and trains the Wav2Vec2 model.
- Test.ipynb: Loads the trained model and evaluates its performance.

Key Script Breakdown:

- Preprocess_&_Prepare_Data.ipynb: Prepares audio files for training by performing actions like feature extraction and saving the processed data.
- Train.ipynb: Defines the model architecture, sets training parameters, trains the model, and saves checkpoints during the process.
- Test.ipynb: Loads the trained model, transcribes audio files, and assesses performance metrics like WER (Word Error Rate).

Running the System

Steps:

1. **Preprocess Data:** Execute all cells in Preprocess_&_Prepare_Data.ipynb on Kaggle TPU (if available).
2. **Transfer Preprocessed Data:** Download the processed data from Kaggle to your local machine.
3. **Train the Model:** Execute all cells in Train.ipynb on your local machine using a phased training approach due to the large dataset size.
4. **Evaluate Performance:** Execute all cells in Test.ipynb on your local machine to assess the model's accuracy.

Example Commands:

- Preprocess data: Run all cells in Preprocess_&_Prepare_Data.ipynb on Kaggle.
- Train the model: Run all cells in Train.ipynb on your local machine.
- Test the model: Run all cells in Test.ipynb on your local machine.

Model Details

Model Architecture:

- Leverages the pre-trained Wav2Vec2 model (facebookwav2vec2-xls-r-300m) for speech recognition.
- Fine-tuned specifically for the Egyptian dialect using a phased training approach on the provided dataset.

Training Process:

- The dataset is divided and trained in phases to handle its large size effectively.
- Further training is conducted on the entire dataset by randomly sampling approximately 40% of the data to improve WER (Word Error Rate).
- The final model is accessible on hugging face: 3BDOAi3/facebookwav2vec2-xls-r-300m-finetuned-with-MTC-Dataset.
- Checkpoints created during training are saved in a designated output directory.
- The best-performing model checkpoint can be found here: https://drive.google.com/drive/folders/1iaDpyfDGHSdddQlzKit0Tq6cujJXweT2?usp=sharing