Voice-to-Text Model Training using Hugging Face Transformers

Overview

This project aims to train a voice-to-text model using the Hugging Face Transformers library. The model uses the Whisper architecture from OpenAI, designed for robust speech recognition tasks, efficient in handling different accents and speaking styles. The training leverages streaming datasets for efficient data handling and includes various optimization strategies to ensure fast and effective model training. The implementation leverages streaming data, data preprocessing, and efficient training strategies to build an effective transcription model.

System Architecture

1. **Data Ingestion**

    - **Dataset**: Mozilla Common Voice 11.0
    - **Data Loading**: Streaming mode using Hugging Face `datasets` library to handle large-scale datasets without local storage limitations.

2. **Data Preprocessing**

    - **Audio Processing**: Convert audio samples to log-Mel spectrogram features.
    - **Text Normalization**: Basic text normalization to handle case and punctuation.

3. **Model Architecture**

    - **Model**: *WhisperForConditionalGeneration* from Hugging Face.
    - **Configuration**: Includes handling special tokens, decoder configurations, and setting maximum lengths for generation.

4. **Training Pipeline**

    - **Trainer**: *Seq2SeqTrainer* for sequence-to-sequence training.
    - **Optimization**: Uses mixed precision training (FP16), gradient checkpointing, and gradient accumulation for efficient resource utilization.
    - **Evaluation**: Periodic evaluation using Word Error Rate (WER) metric.

Algorithms and Implementation

1. Data Loading and Preparation

Streaming Dataset Loading: The system loads the Common Voice dataset using streaming mode to handle large-scale audio data without overwhelming memory resources.

Data Processing for '*WhisperProcessor*': Audio data is processed to extract features required for model training.

Three steps can be distinguished in the voice to text pipeline –

1) An extractor of features that preprocesses the unprocessed audio inputs
2) The model that carries out the mapping from sequence to sequence
3) A tokenizer that converts the model outputs to text format after post-processing

WhisperFeatureExtractor and *WhisperTokenizer*, the corresponding feature extractor and tokenizer, are connected to the Whisper model in HuggingFace Transformers. These two objects are combined into a single class, the *WhisperProcessor*. Both the audio pre-processing and further text token processing can be done by calling the WhisperProcessor. This reduces the number of items we need to monitor during training to only the processor and the model.

Data Pre-processing: Input audio (48Khz) is down sampled at 16KHz which is compatible with the sampling rate for Whisper model. The 1-D audio array is computed to log Mel spectrogram by the feature extractor. Tokenizer encodes the transcription to respective label id.

2. Model and Training Configuration

Model Initialization: The pre-trained Whisper 'small' model is loaded and configured for training.

The Whisper checkpoints are available in five configurations of different sizes and here small model having 244M params, is being used.

Training Arguments: Training arguments are optimized for efficient training.

Data Collator: A custom data collator is implemented to handle padding and batching.

Trainer Callback: A callback is used to shuffle the dataset at the beginning of each epoch.

3. Evaluation

WER Metric: Word Error Rate (WER) is a primary and most commonly used metric, computed to evaluate the model's performance. Lower values indicate better performance.

WER is evaluated at different steps determined by the 'eval_steps' parameter in training argument. The best possible WER for this model was approximately 35.74%.

4. Training

Training the Model: The Seq2SeqTrainer is used to train the model with the specified configurations. Utilizes GPU acceleration with PyTorch and Huggingface's Accelerate library to speed up training.

After fine-tuning the model, a demo is built using Gradio for real time transcription using microphone as input audio.

The system leverages streaming datasets to handle large audio data efficiently, optimizes training parameters for faster training, and evaluates model performance using the Word Error Rate metric.