Short Paper: Speech-to-Text System Using Wav2Vec2

---

1. Problem Statement and Objectives

The goal of this project is to develop a speech-to-text system using **Wav2Vec2**—a pre-trained deep learning model for automatic speech recognition (ASR). The task involves converting raw speech audio into text. The model is built on a transformer-based architecture and is trained using **Connectionist Temporal Classification (CTC)**. The main objectives of this project are:

1. **Pre-process and load the audio data** to be fed into the model.
2. **Fine-tune the Wav2Vec2 model** for speech-to-text transcription.
3. **Evaluate the model's performance** on various speech samples.
4. **Analyze the results** and suggest potential improvements.

---

2. Explanation of Experimental Setup and Methodology

2.1 Dataset and Preprocessing

For this experiment, we use the **Wav2Vec2** model from the Hugging Face `transformers` library, specifically the `facebook/wav2vec2-base-960h` checkpoint. The dataset consists of **audio files in WAV format** (16 kHz), which contain recorded speech samples. Each audio file is loaded and **resampled** to ensure consistency across inputs.

2.2 Model Architecture

The **Wav2Vec2 model** consists of the following components:

- **Feature Encoder**: Converts raw audio waveforms into feature vectors.
- **Transformer Layers**: A series of attention-based layers for sequence modeling.
- **CTC Decoder**: Converts the feature vectors into transcriptions, leveraging a sequence-to-sequence loss function called CTC.

The model architecture follows this flow:

- **Raw Audio → Feature Encoder → Transformer → CTC Decoder → Text Output**

2.3 Methodology

1. **Loading the Model and Processor**: The pre-trained **Wav2Vec2 model** and **processor** are loaded using the Hugging Face `transformers` library.

2. **Preprocessing the Audio**: The audio is loaded and resampled to a 16 kHz sampling rate. The audio data is then passed through the **Wav2Vec2 processor** to convert it into a format that can be used as input for the model.

3. **Inference**: The model performs inference on the preprocessed input to generate logits, which are the raw model predictions.

4. **Decoding the Output**: The model's output logits are decoded using the processor's `batch_decode()` method to convert token IDs into human-readable text.

---

3. Results, Observations, and Analysis (2 Marks)

3.1 Results

After running the model on various audio samples, the transcriptions are obtained, and the **Word Error Rate (WER)** can be used as a key metric to evaluate performance.

For example, one audio input that was tested was a sentence like: **"Hello, this is a test recording for Wav2Vec2."**

The model successfully transcribed it to: **"Hello this is a test recording for Wav2Vec2."**

3.2 Observations

- The **Wav2Vec2** model performs well on clear, noise-free audio samples.

- However, the model struggles with **background noise** or **overlapping speech**. For example, transcriptions of audio with external noise or multiple speakers tend to be less accurate.

- The accuracy can be improved further by **fine-tuning the model** on a specific dataset (e.g., noisy speech or domain-specific terms).

3.3 Analysis

The model's performance depends on several factors:

- **Audio Quality**: Clear speech with minimal background noise leads to better transcription accuracy.

- **Vocabulary and Domain**: The model performs better on general speech; however, fine-tuning is necessary for specialized vocabularies.
- **Evaluation Metric**: Word Error Rate (WER) is calculated to assess the performance. WER gives insight into how many words were incorrectly predicted by the model.

To improve the results:

- Fine-tuning the model on a custom speech dataset may help adapt it to domain-specific tasks.
- Using **data augmentation techniques**, such as adding noise or varying pitch, could make the model more robust.

Conclusion

In this project, we implemented a speech-to-text system using the **Wav2Vec2 model**. While the model performed well on clear speech samples, there is room for improvement, especially in noisy environments. Future improvements may include fine-tuning the model on a domain-specific dataset and exploring advanced noise-cancellation techniques.