Librispeech ASR

Mohana Evuri - 23B1017
Abhinav B - 23B1018

Spring 2025

The Problem Statement

- **Input:**
  - Audio file (`.wav` format), with a person speaking English text. (We have used files of a line or two).
- **Output:**
  - The phonetic sequence that is formed by decomposing the words spoken in the audio file.
  - The words spoken in the audio file.
- For example, an audio file is given, with some speech in it:
- The output would be something like:
  - DH AH F AA DH ER AH N D S AH N HH AE D B IH N K L OW S K AH M P AE N Y AH N Z AH N D DH AH AY D IY AH AH V B IY IH NG L EH F T AH L OW N W IH DH DH AH R EH M N AH N T AH V AH T EY S T L AH S L AY F AA N HH IH Z HH AE N D Z W AA Z N AA T G R AE T AH F AY IH NG T UW DH AH Y AH NG M AE N HH UW HH AE D AO L W EY Z AH N D T AE S IH T L IY K AW N T AH D AH P AA N HH IH Z EH L D ER Z HH EH L P IH N M EY K IH NG DH AH B EH S T AH V AH P UH R B IH Z N AH S
  - THE FATHER AND SON HAD BEEN CLOSE COMPANIONS AND THE IDEA OF BEING LEFT ALONE WITH THE REMNANT OF A TASTELESS LIFE ON HIS HANDS WAS NOT GRATIFYING TO THE YOUNG MAN WHO HAD ALWAYS AND TACITLY COUNTED UPON HIS ELDER'S HELP IN MAKING THE BEST OF A POOR BUSINESS

The Problem Statement

- The aim of this project is to make a model, which can do **speech recognition (ASR) after training** on a dataset called "**Librispeech**". The training is done using a **Hidden Markov Model** (HMM), with the features extracted from the audio.
- The **Librispeech dataset** has speech recording of a **1000 hours**, analyzed using the **`Kaldi`** toolkit. In this project, we try to replicate this via a smaller set of 1 hour and by using a much lighter tool set of basic python libraries such as **`hmmlearn`**.

Motivation

- We have chosen this paper as both of us have never tried an AI/ML project regarding **speech recognition** (ASR), and this looked like a very fun topic to work on.
- We had previously worked on image processing with encoder-decoders, neural networks and other stuff, but not a **Hidden Markov Model** (HMM).
- This paper claims that training on this **Librispeech dataset yields better results** on other datasets such as the **Wall Street Journal (WSJ)** dataset, compared to the model trained on this dataset itself.
- Using HMMs to train an audio dataset needs a lot of attention on preprocessing too, which is kind of interesting and also makes it challenging.

Literature Review

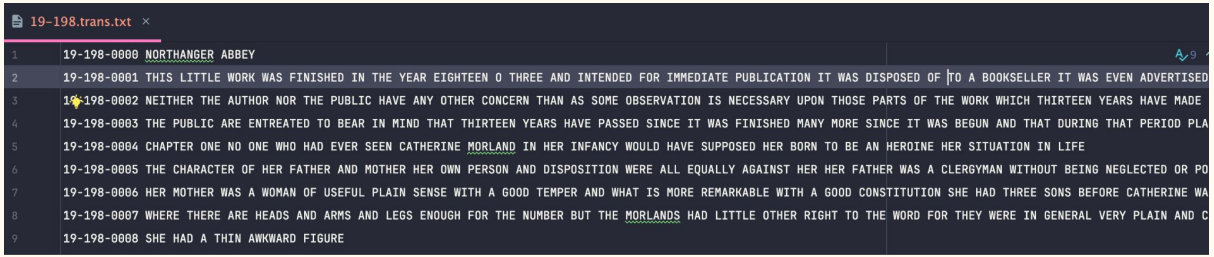
- **Automatic Speech Recognition** (ASR) is the technology that converts spoken language into written text. It processes audio input, extracts features (like MFCCs), models the acoustic patterns (using HMMs or deep learning), and decodes them using a language model to produce the most likely transcription.
- **Mel-Frequency Cepstral Coefficients (MFCCs)**: Mimic human hearing by mapping frequencies to the Mel scale.
- **Hidden Markov Models** are probabilistic models used in ASR to represent sequences of speech sounds:
  - Model speech as a sequence of hidden states (e.g., phonemes).
  - Each state emits observations (e.g., MFCCs) based on a probability distribution.
  - Transition probabilities define how likely one state is to follow another.

Literature Review

- **ASR – Prior Work** (Data collected from internet):
  - Template & DTW-based (1950s–1980s): Early systems used pattern matching for isolated word recognition.
  - HMM-GMM (1990s–2000s): Statistical models like Hidden Markov Models with Gaussian Mixture Models dominated ASR.
  - DNN-HMM Hybrids (2010s): Deep neural networks replaced GMMs for better acoustic modeling.
  - End-to-End & Self-Supervised (2016–present): Models like DeepSpeech, Wav2Vec 2.0, and Whisper enable high-accuracy transcription with minimal feature engineering.

Dataset

- The dataset we used, as our project name suggests, is the Librispeech dataset.
- We used a 1 hour version of the original 1000 hour dataset (This was told to be done in the original problem statement given to us).
- We have 44 phonetics, out of which we get 39 features (stored after training).



Method/Technique - Preprocessing

The preprocessing part does the feature extraction.

- Read each WAV at its native sampling rate.
- For the processed audio, we compute 13-dim MFCC vectors.
- For the transcript, convert words into phoneme sequences
- Achieved via the `soundfile`, `librosa` , `scipy.wavfile` and the `g2p_en` libraries.

Method/Technique - Dataset Assembly

- Shuffle WAV list; accumulate until ~3600s (1 hour) of audio.
- For each file
  - Load MFCCs, load transcript, convert to phonemes
  - Build a map { phoneme → index }, encode phoneme sequences as integer lists.
  - Save JSONs and numpy arrays for training.

Method/Technique - Training

- **MFCC Normalization & Segmentation**: Normalize MFCCs (zero mean, unit variance); estimate phoneme durations via forced alignment or uniform splits.
- **HMM Initialization**: Create left-to-right HMMs (3–5 states/phoneme), initialize Gaussian mixtures (e.g., k-means), and set uniform transitions with self-loop bias.
- **Training with Baum–Welch**: Run EM (10–20 iterations) to refine HMM parameters per phoneme; optionally tie states or build triphone models.
- **Decoding with Viterbi**: Normalize test MFCCs, build composite HMMs from phoneme models, and apply Viterbi search with beam pruning to decode state paths.
- **Post-Processing**: Collapse state sequences into phoneme strings; optionally apply duration penalties or a phoneme-level language model. Use an inverse G2P dictionary and greedy/beam search (up to 5-gram) to reconstruct word sequences.

Results

For the Phoneme Match, we achieved **92.81%** accuracy.

Analysis

- The results for the phonetics have achieved a great accuracy.
- Construction of the words into sentences is fine for small words, but not great for larger words or words with silent letters.

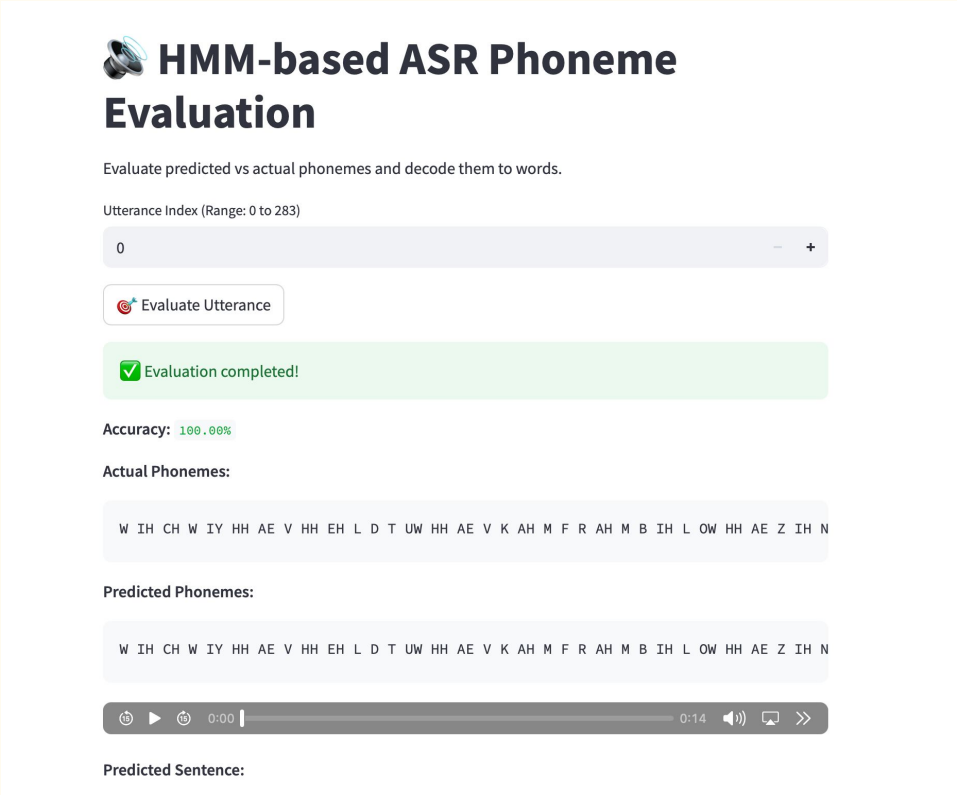
Error Analysis

We have faced an issue, where the model predicts till a particular phoneme, and justs repeats that value over and over. We could correct that by adjusting the alignment MFCC features properly.

Learnings

- **Recap**: Developed an HMM-based ASR system using MFCCs, phoneme models, and Viterbi decoding.
- **Insight**: HMMs can align and decode speech well but lack robustness to variability.
- **Future Work**: Improve context modeling, integrate stronger language models, and enhance noise/speaker robustness, along with improving the final text detection model

A Live Demo (using `streamlit`) !



Summary and Conclusion

- Built a small-scale ASR system using ~1 hour of LibriSpeech, MFCCs, and phoneme-level HMMs with Viterbi decoding.
- GMM-HMMs are interpretable and data-efficient, capturing basic spectral and temporal patterns.
- Phoneme recognition is reasonable, but word-level performance is limited without a strong language model.
- Simple inverse-CMU mapping helps recover words, but suffers from ambiguity and OOV issues.