AUTOMATIC SPEECH RECOGNIZATION - (English)

GROUP 2: APARANJITHA KOTHINTI | PARI DHANDAPANI | UDAYKUMAR BKVENKATA

Innovative AI project designed to push the boundaries of technology and drive transformative change in various industries.

PROBLEM STATEMENT

Developing an Automatic Speech Recognition (ASR) system aimed at accurately transcribing spoken language into written text across diverse domains and accents. The primary focus is on enhancing recognition performance in challenging acoustic environments and effectively managing out-of-vocabulary words.

This problem statement tackles two main challenges in ASR:

1. Speech Recognition Performance in Challenging Acoustic Conditions
2. Dealing with Out-of-Vocabulary (OOV) Words

Meta Information considered for model Training and testing:

- Data Language: English
- 300 hrs of Transcribed Data
- Format of data: Wav
- Sample Rate : 16k

Kaldi Installation

Clone repo from https://github.com/kaldi-asr/kaldi

Steps:

sudo apt update
sudo apt install -y cmake sox ffmpeg g++ automake autoconf libtool subversion git zlib1g-dev unzip gfortran python2.7 python3 gawk

ffmpeg: resample data to desired Hz

cd kaldi/tools
extras/check_dependencies.sh
extras/install_mkl.sh

RaspberryPI:
make -j 'nproc'

make
extras/install_irstlm.sh
./install_srilm.sh <name> <organisation> <email> <address>
Ex: ./install_srilm.sh a123 b123 acb@gmail.com 812345 (command i executed)

1. Installation of kaldi on Raspberry device with Ubuntu OS
2. Defined a new module/setup to fix srilm on Ubuntu
3. Patch ups on few utilities and libraries
4. Conversation of FLAC to WAV, as changing extension will throw error while MFCC feature extraction. Used Audiosegment to convert to bytes and export approach

Tools & Infra:

- Programming : Python, Pip
- ASR: Kaldi, AudioSegment
- Front end: Steamlint, Python
  Back end: fastAPI
- Infrastructure: Azure cloud
- Operating and system: Ubuntu OS in RaspberryPI5

Data Preparation

Automatic Speech Recognition (ASR) using Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) is a conventional method extensively employed in the field. Here's an overview of the approach involved in ASR using HMMs and GMMs:

1. Acoustic Modelling:

The acoustic model captures the relationship between speech features and units. In the HMM-GMM approach, GMMs are commonly used. Each phonetic unit is represented by a GMM, modelling the probability distribution of speech features for that unit.

Wave.scp

Text (transcripts)

utt2spk (utterance to speaker mapping)

2. Language Modelling:

It serves as a critical component in automatic speech recognition (ASR) for navigating linguistic constraints and enhancing recognition accuracy. These models encompass the statistical patterns of language, facilitating the identification of the most probable word sequences based on observed speech features. Both N-gram language models and more sophisticated alternatives such as hidden Markov models or neural networks can be employed for this purpose.

Lexican (pronounciation dictionary)

non-silence_phones

optional_silence

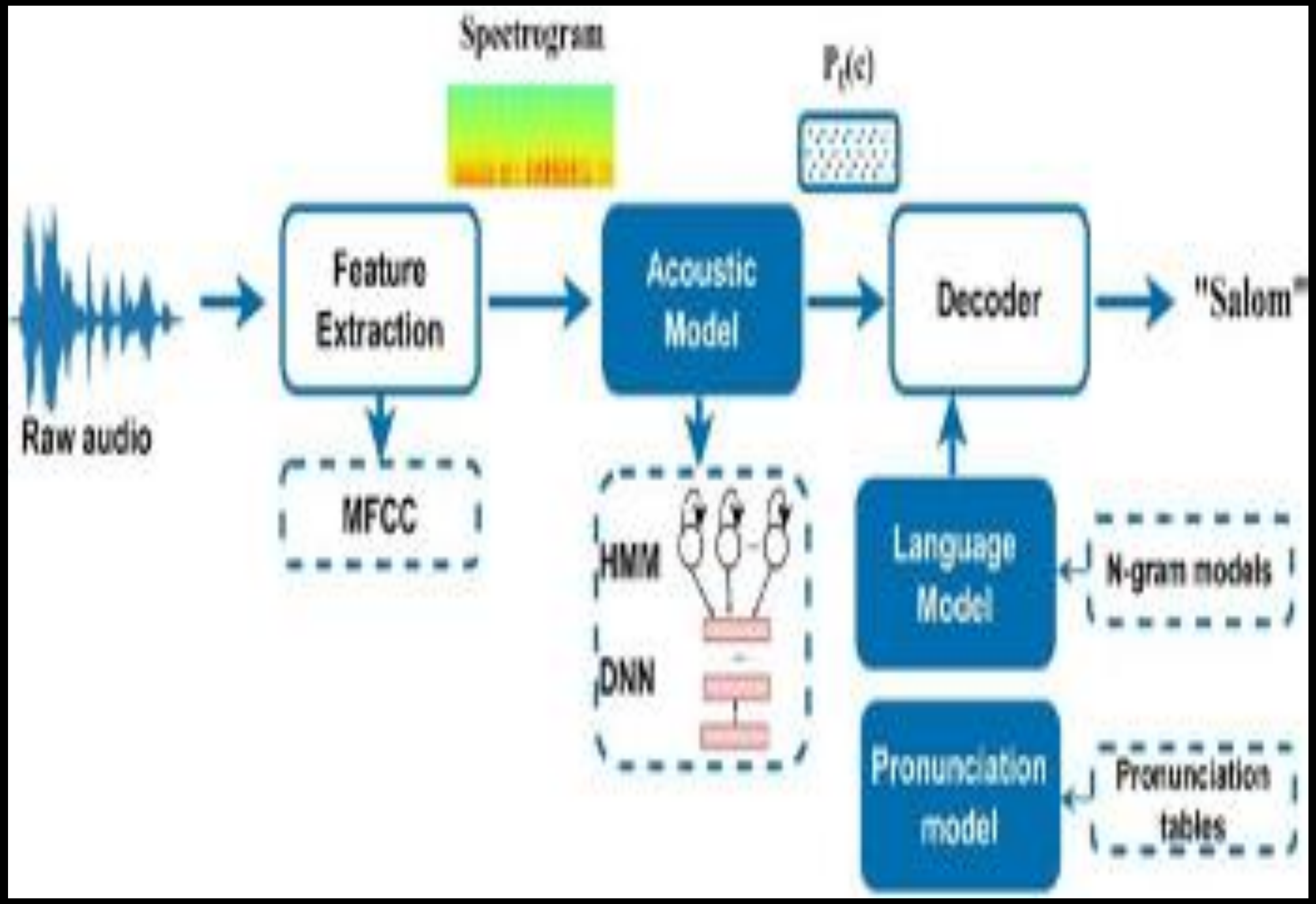
silence_phones

Pre Data preparation:

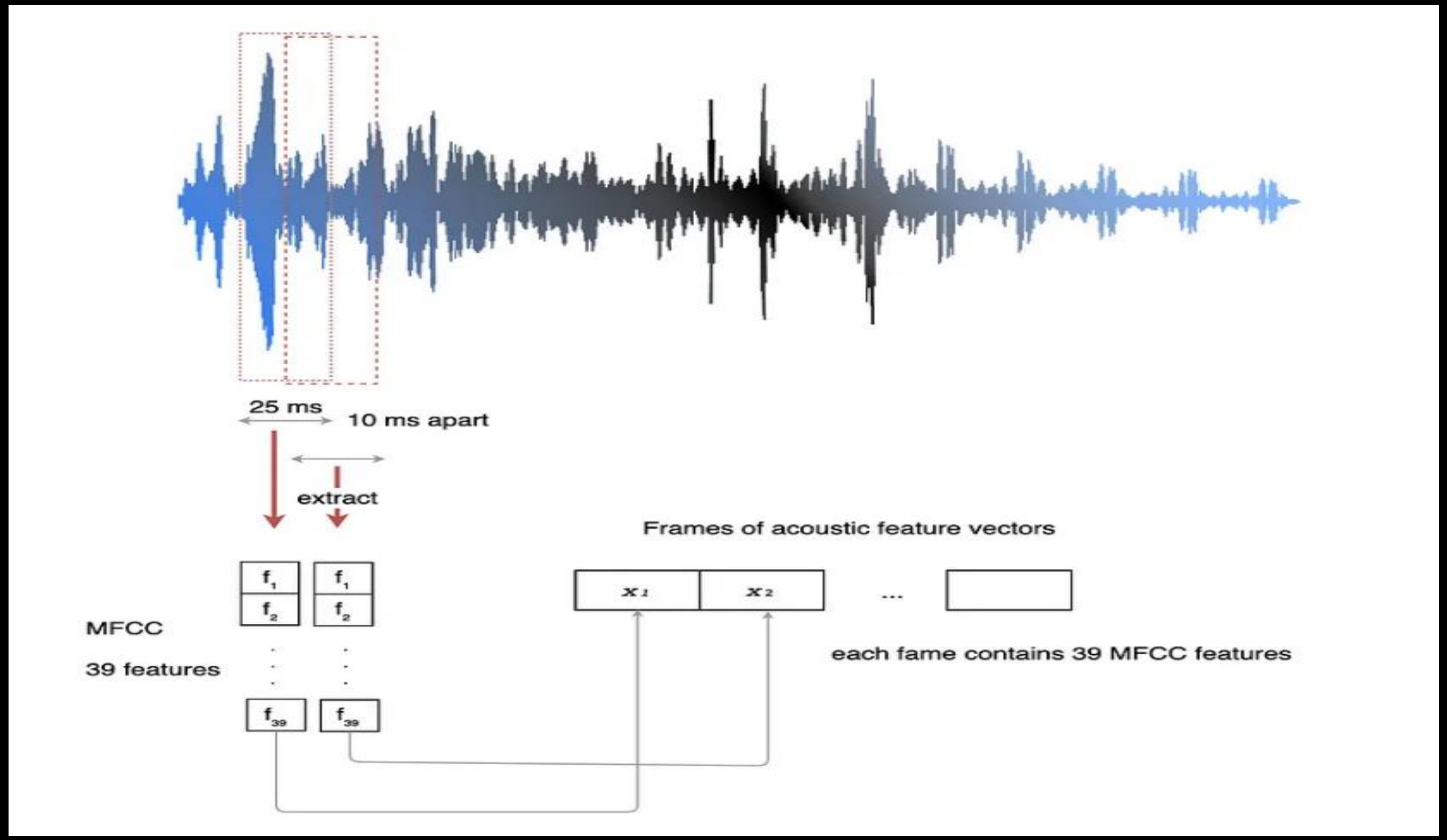
Conversion of FLAC files to WAV using AudioSegment library

Modelling

Automatic Speech Recognition (ASR) using Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) is a conventional method extensively employed in the field. Here's an overview of the approach involved in ASR using HMMs and GMMs:



- Feature Extraction – MFCC from the audio



- Used GMM-HMM Model (2000 HMM states)
- For LM, used SRILM with n-gram as 3

SRILM package with fix for Raspberrypi5:
https://drive.google.com/file/d/1j4pcfoXWMATdM7uF4nCJs-m7DBkjQb8p/view?usp=drive_link

Word Error Rate

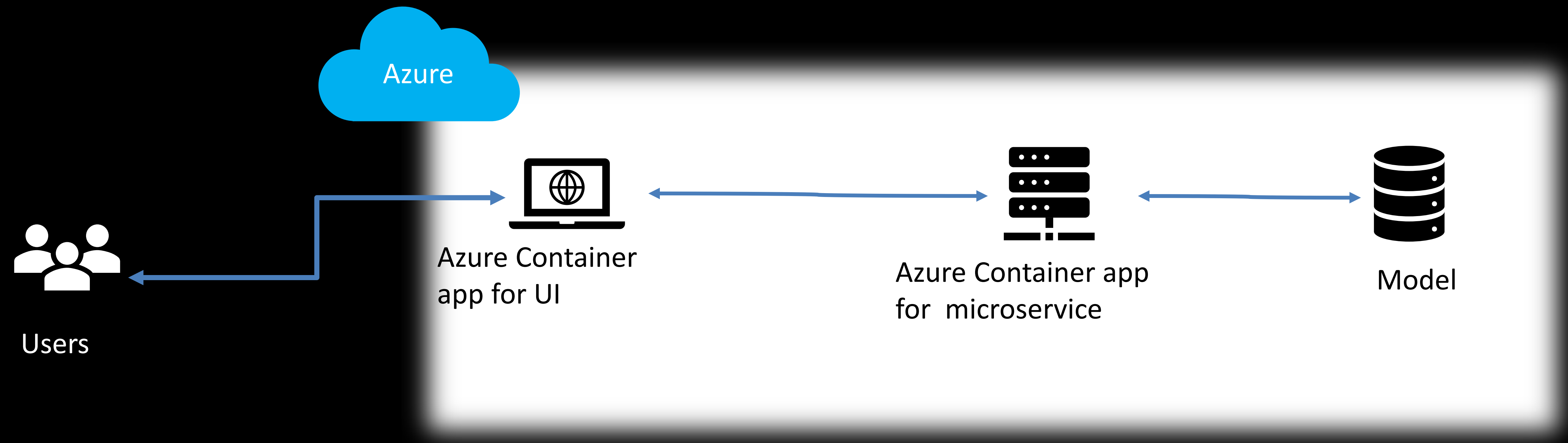
$$WER = \left(\frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\text{Total number of words on reference}} \right)$$

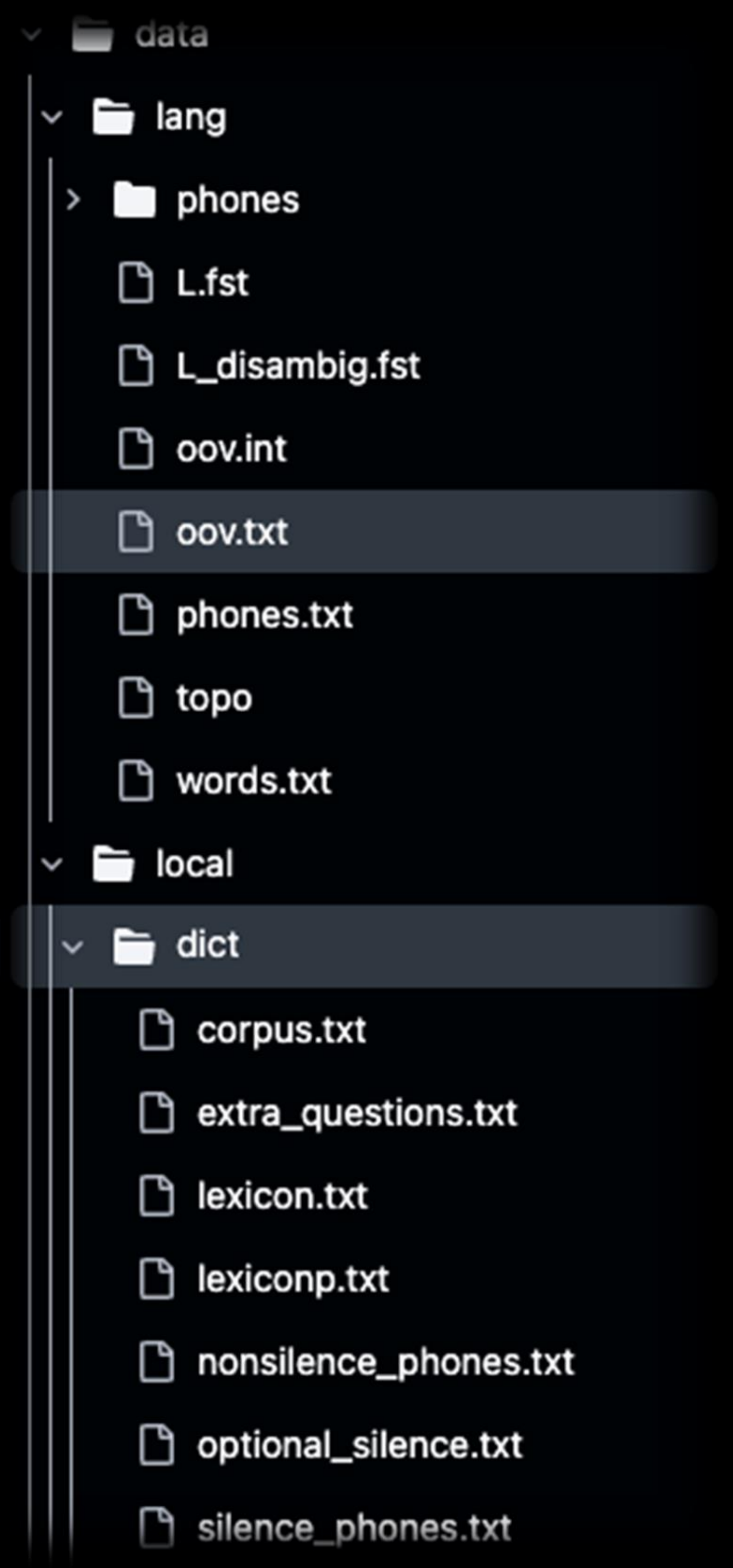
Example Original Audio:
The quick brown fox jumps over the lazy dog

Example Transcribed Audio:
The brown fox jumps again over the crazy dog

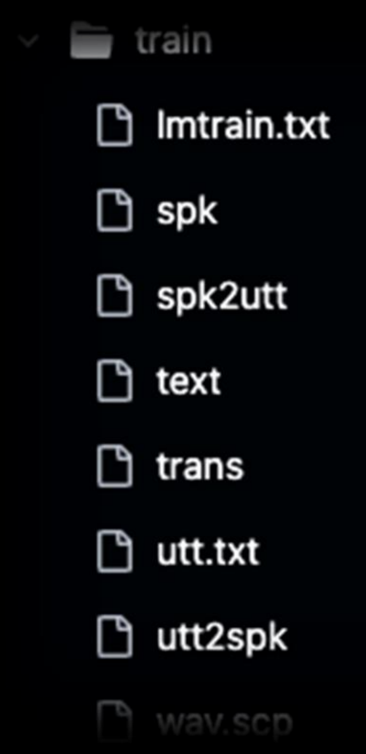
WER of the ASR model with test data = 29 %

ASR System Architecture





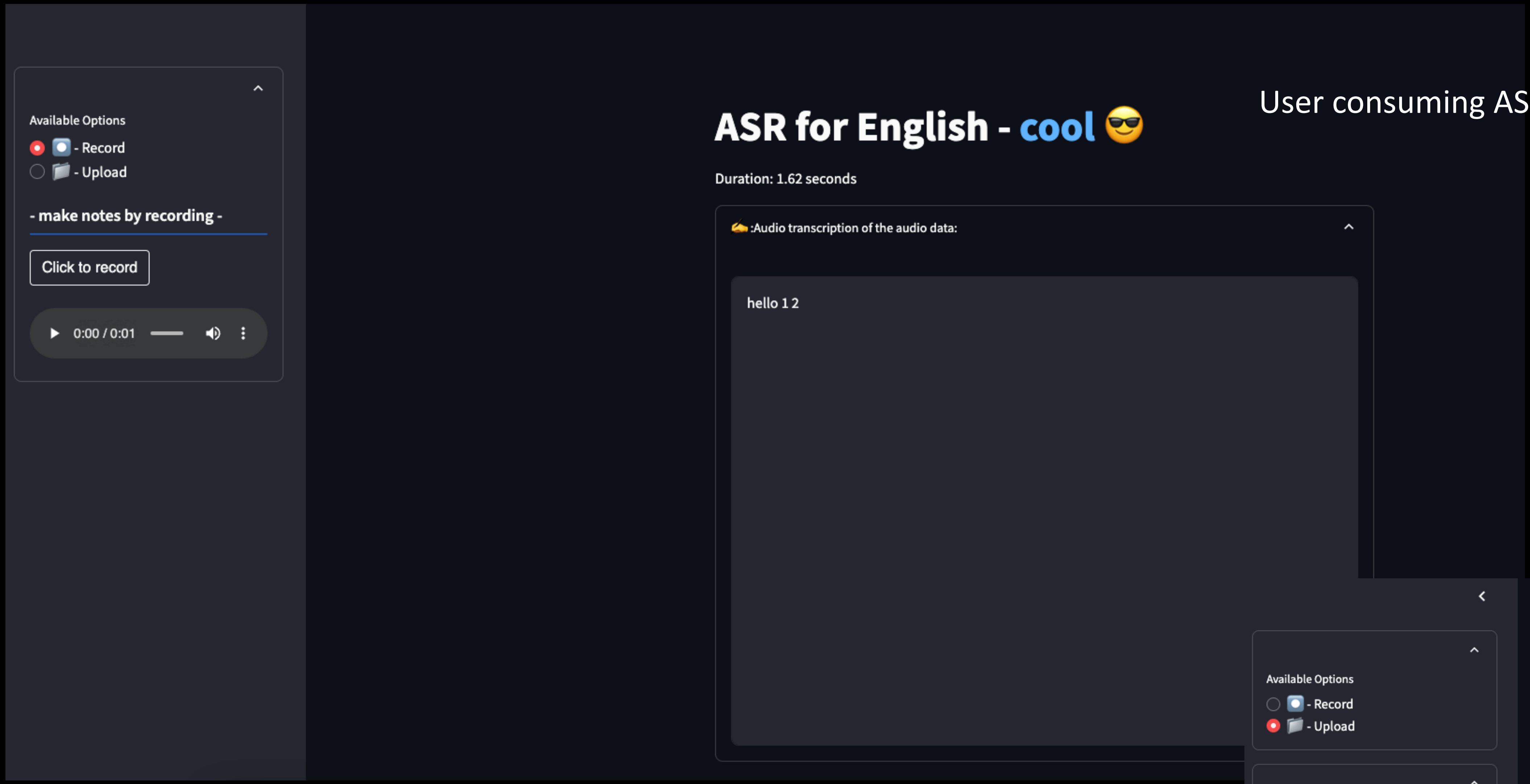
ASR DEMO



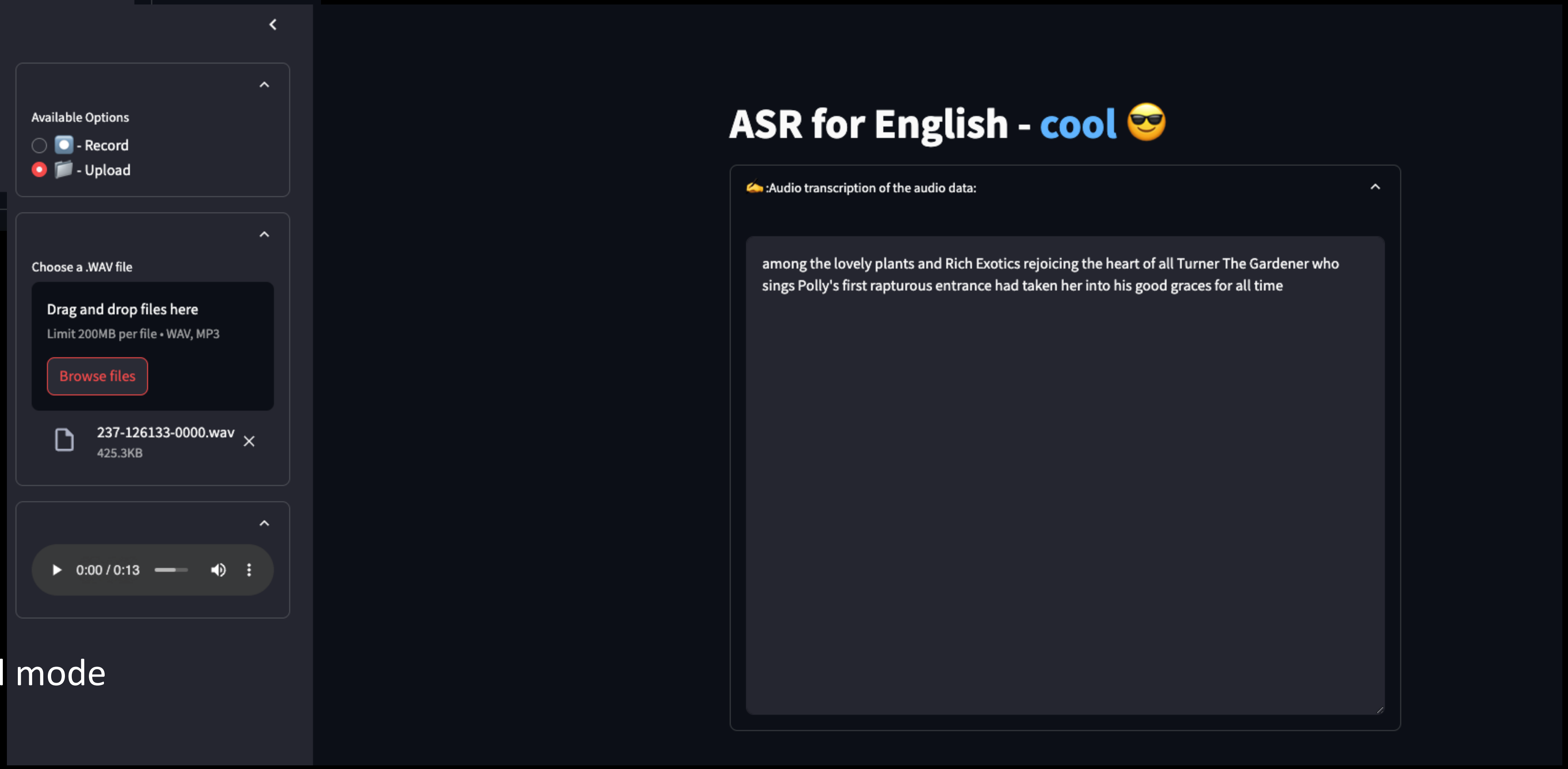
Git Hub Code Repo:

- ASR Model : uday160386/ai_ml_asr_text (github.com)
- ASR UI : uday160386/asr_capstone_en_ui (github.com)
- ASR Microservice: uday160386/asr_capstone_en_ms (github.com)

ASR System Screenshots



User consuming ASR system in a recording mode

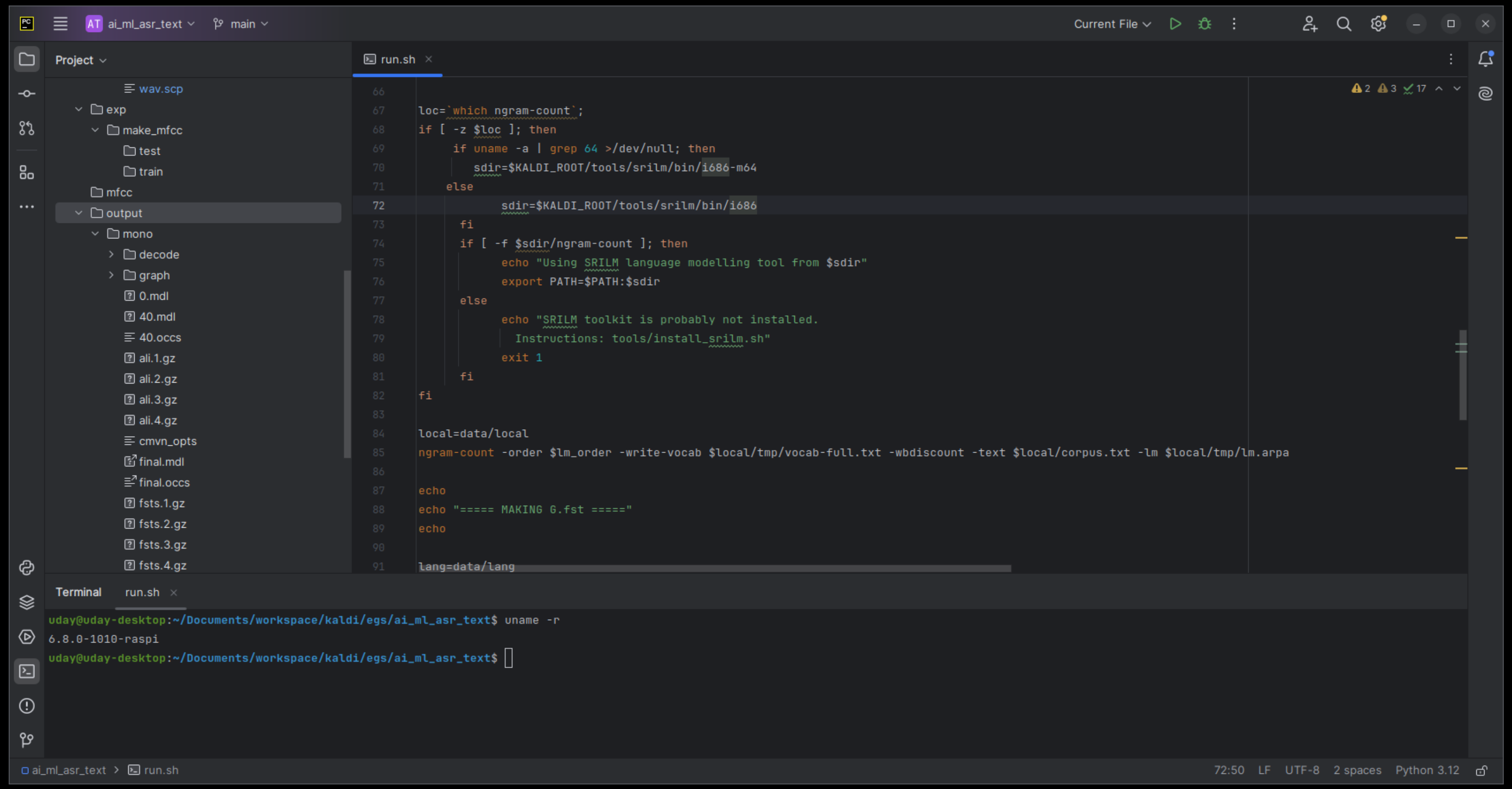


User consuming ASR system in a file upload mode

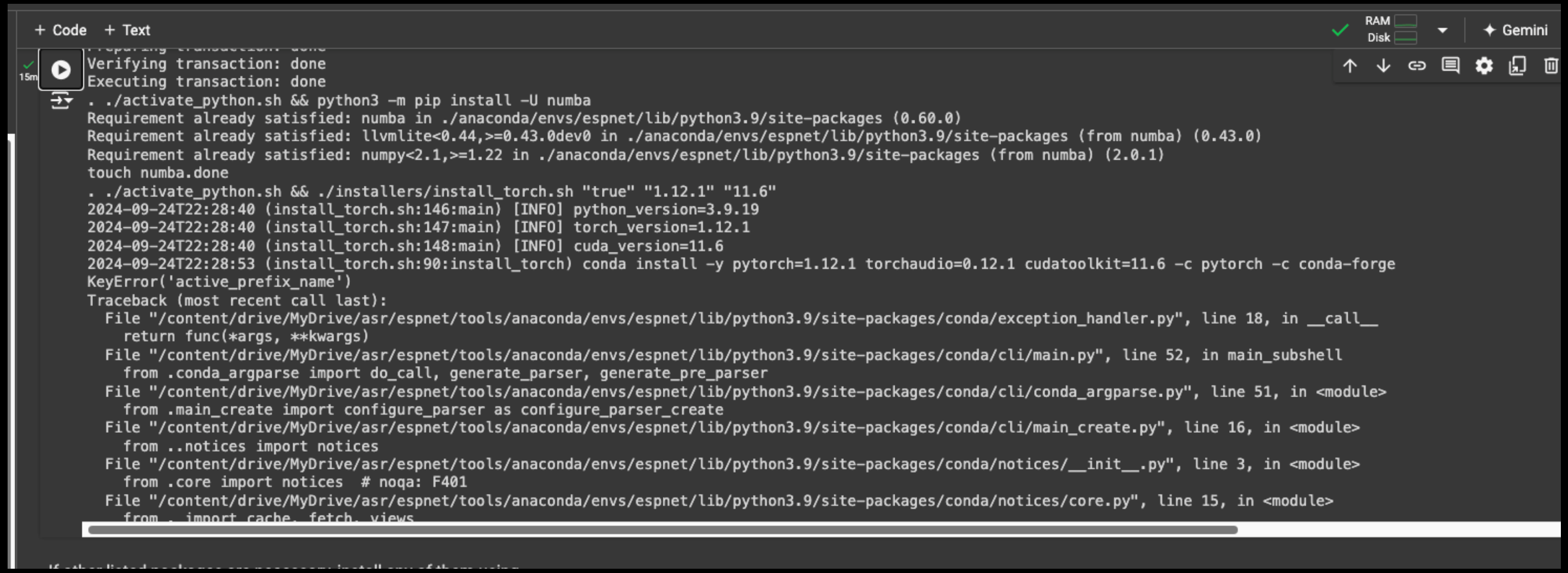
KEY MILESTONES

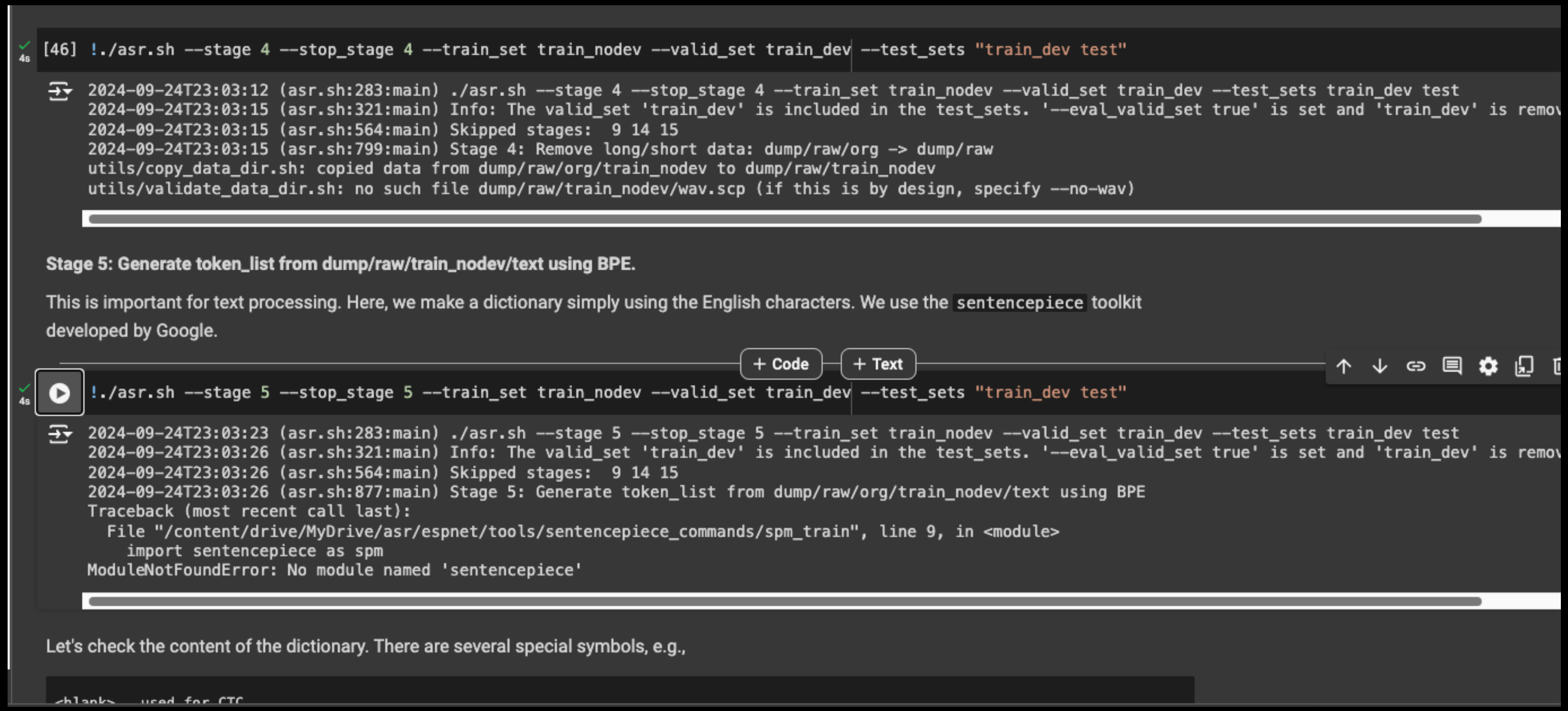
This project timeline are divided into key milestones, including platform design, development, testing, and deployment.

- Kaldi Installation Completed ✅
- Data Preparation Completed ✅
- Development of Model: ✅
- Training model: ✅
- Test model : ✅
- Demo: ✅



ASR using ESPnet

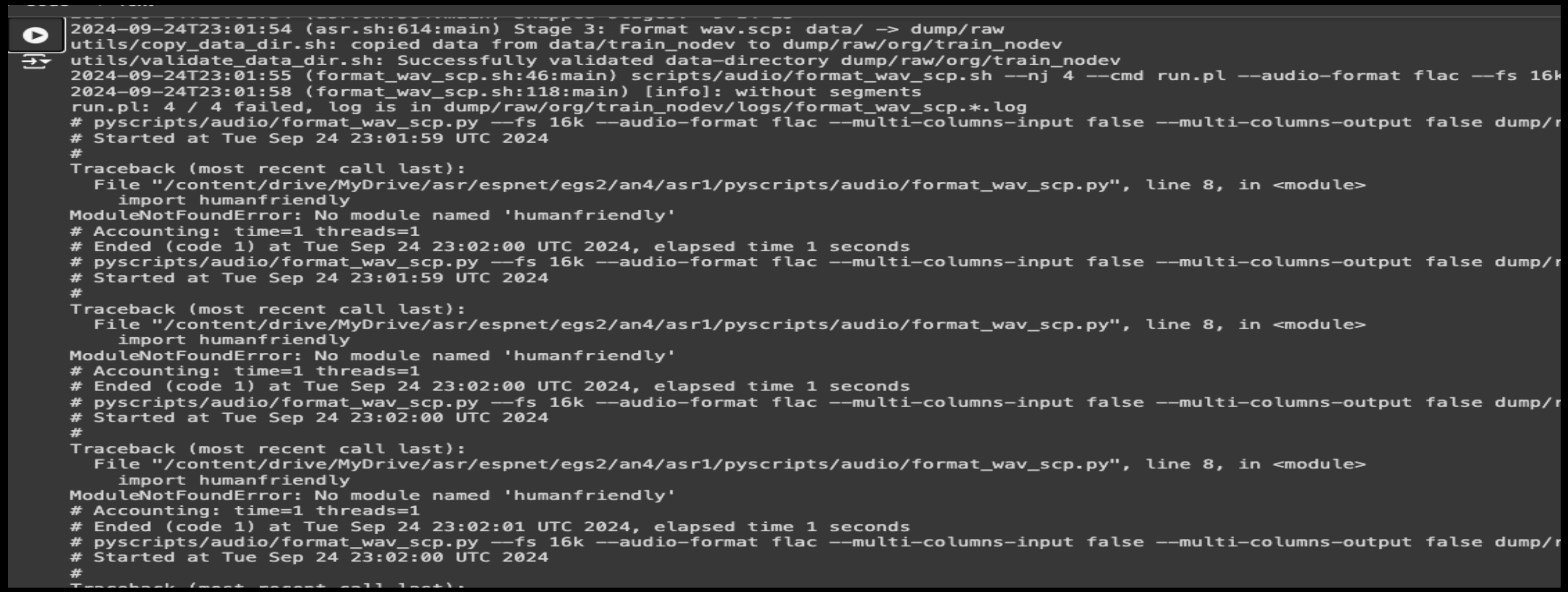


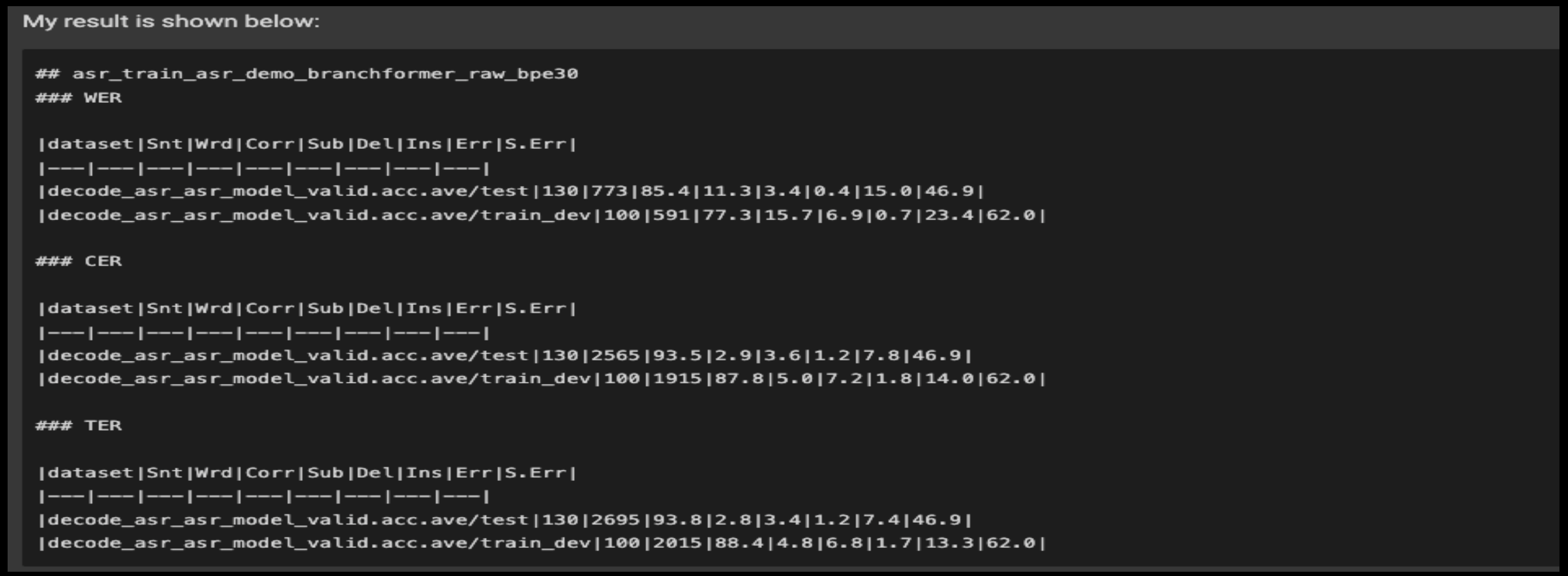


ESPnet example code:
https://colab.research.google.com/drive/1F5IXJqzljBrJr3N_6UgW-EPTmfZlOGby?usp=drive_link

Reason for Failure:

- Limited GPUS and frequent disconnects from Data
- Issues in installing dependency python packages
- **Blocker** : !./asr.sh --stage 3 --stop_stage 3 --train_set train_nodev --valid_set train_dev --test_sets "train_dev test" --nj 4

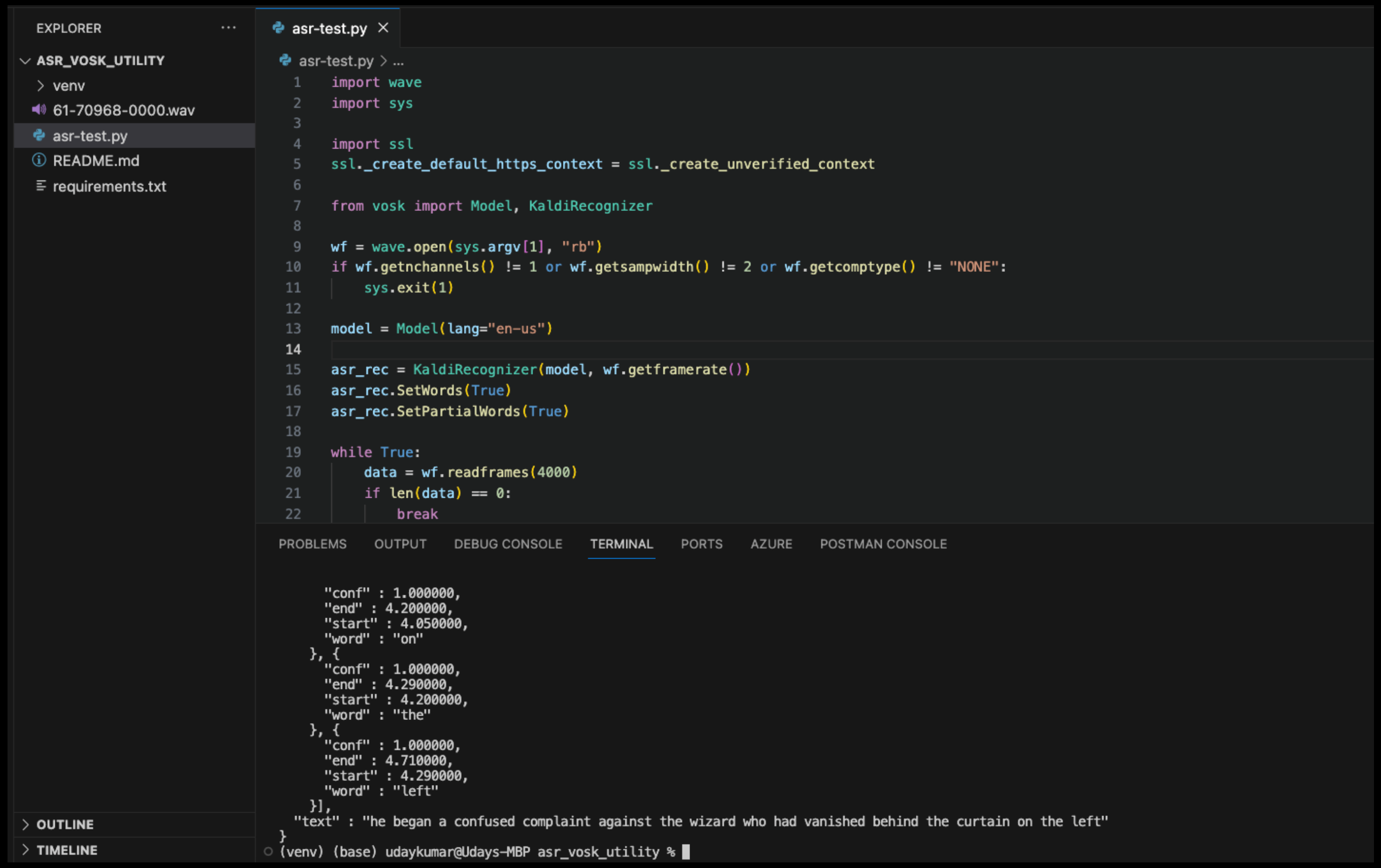




VOSK API

- Vosk is a speech recognition toolkit. The best things in Vosk are:
  Supports 20+ languages and dialects - English, Indian English, German, French, Spanish, Portuguese, Chinese, Russian, Turkish, Vietnamese, Italian, Dutch, Catalan, Arabic, Greek, Farsi, Filipino, Ukrainian, Kazakh, Swedish, Japanese, Esperanto, Hindi, Czech, Polish, Uzbek, Korean, Breton, Gujarati, Tajik.

- Works offline, even on lightweight devices - Raspberry Pi, Android, iOS
- Provides streaming API for the best user experience (unlike popular speech-recognition python packages)
- There are bindings for different programming languages, too - java/csharp/javascript etc.
- Allows quick reconfiguration of vocabulary for best accuracy.
- Supports speaker identification beside simple speech recognition.



Models used:
[RNNLM]

Source Code:

[asr_vosk_utility]