Problem Illustration

English meaning: Want steak or salmon for dinner?

iPhone voice message app voice to text output:

English meaning: I want to eat this Sam for dinner

Correct output: 今晚想吃steak还是salmon

Use Case

- Problem:
  - Most speech recognition apps today only accurate in recognizing single language
- Stakeholders:
  - Mandarin-English bilinguals, Chinese international students
- Use Scenario:
  - Voice-texting with a mix of Chinese and English
- Goal:
  - Design an app that provides real-time voice-to-text recognition for speeches mixed with English and Mandarin

Use-case Requirements

- Reasonable output
  - Audio >> Mandarin-English-mixed text transcript
  - **Word error rate (WER) < 10%**
  - Popular speech recognition apps (Google, Apple …) WER 5%~10%
- Reasonable vocab recognition
  - Recognize daily words in English and Mandarin
  - **Recognize 60K most frequent English words and Mandarin words**
  - Vocab size used by many research papers
- Real-time text output
  - matches human normal speaking rate (100 words per second)
  - **End to end latency within 1 second**

Use-case Requirements

- Noise tolerance
  - Recognition remains accurate when input audio is noisy
  - Signal to noise ratio (SNR) 16-24dB (decibels) is usually considered poor
  - **Remain < 10% word error rate when SNR in the audio is higher than 25dB**
- Cross-device support
  - App should be **usable on common laptop OS (MacOS, Windows, Linux)**
  - Shall be usable on the newest Chrome browser (version released after 1/12) on a laptop
- Reasonable speech length
  - Support recognition on long continuous speech (**up to 1 minute audio input**)

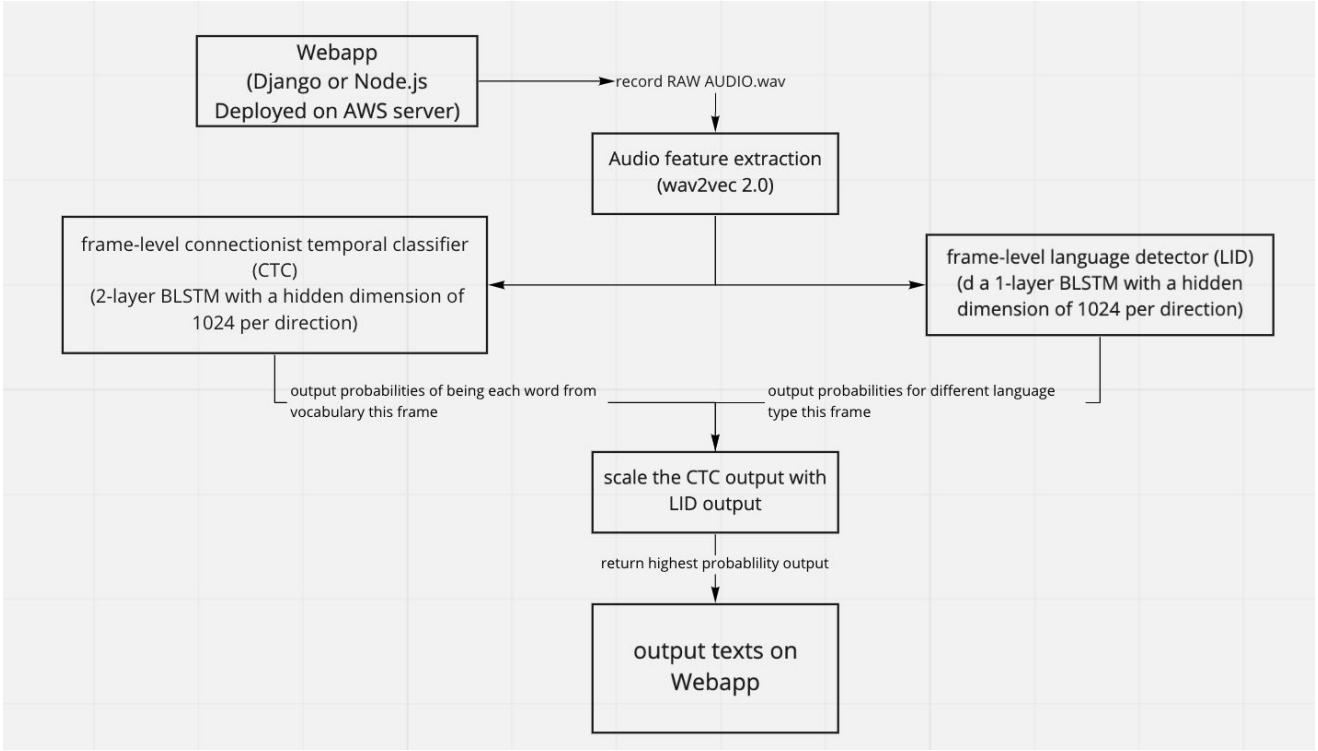
Technical Challenges

- Shortage of public code-switching codebase
  - Mitigation:
    - experiment with a model combination described in an **academic paper**
    - try **combining existing language detection models and speech recognition models**
- Lack of available code-switching datasets
  - Existing code-switching papers used datasets that cost beyond our budget
  - Mitigation: found a free Mandarin-English **100+ hours audio+transcript dataset**

Technical Challenges

- Uncertainty in model training time and cloud computing resources
  - Existing papers do not show the duration of their training phase
  - Full dataset used in relevant academic papers are hundreds of hours
  - Mitigation: **always test models on small datasets** first before running it on large datasets
- Restraint on deployment server computation power
  - Most speech recognition models today rely on GPUs, hard to get many
  - Mitigation:
    - set the audio **recording time limit to 1 minute** (<10MB audio data)
    - may need faculty help to request GPU resources

Solution Approach



Tools

- Django framework for web app
  - Backend support Python
  - Existing package for sending and reading audio stream to wav file
- Tensorflow
  - Neural network initialization and training
- Transformer library
  - Existing speech recognition models
  - Existing feature extraction code such as wav2vec 2.0
- Google Colab
  - Model training and tuning
- AWS server
  - App deployment

Testing & Verification

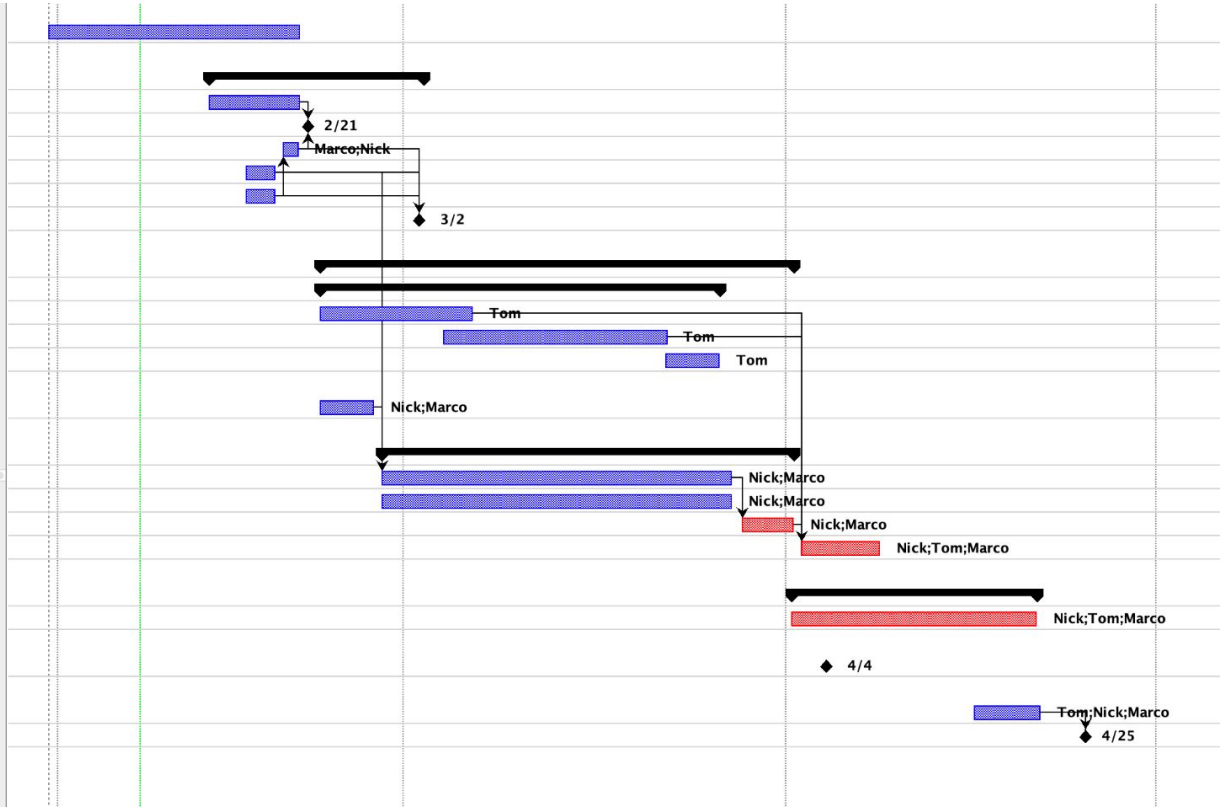
- Latency test
  - Measure the **average time taken for transcription to begin once the first audio is recorded**
  - Should be < 2000ms
  - Based on result tune the time interval we pack an audio stream to send to server
- Throughput test
  - Measure the **average number of transcribed words per second** when speaking at 100 words per second
  - Should be > 1 word per second
- Error rate test
  - Measure the **average word error rates** for audio test dataset
  - Should be < 10%

Testing & Verification

- Noise test
  - Using matlab to compute and group test audio datasets by **SNR (25dB, 30dB, 40dB, 50dB)**
  - measure **average translation error rate** on each group
  - Should be < 10%
- Vocab test
  - measure the average translation error rate when **audio includes random vocabs in English and Chinese**
  - Should be < 10%
- Browser test
  - measure average translation error rate of our app when running on Chrome browser on **Linux, MacOS, Windows** devices

Schedule

Research	16.8 days?	1/31/22 8:00 AM
Design	14.4 days?	**2/13/22 8:00 AM**
Design Presentation Work	6.4 days?	2/13/22 8:00 AM
Design Presentation	0 days?	2/21/22 8:00 AM
Detailed Architecture	1.3 days?	2/19/22 8:00 AM
LM Architecture 1	2.3 days?	2/16/22 8:00 AM
LM Architecture 2	2.4 days?	2/16/22 8:00 AM
Design Document	0.8 days?	3/2/22 8:00 AM
Implementation	31.2 days?	**2/22/22 8:00 AM**
Web App	26.4 days?	2/22/22 8:00 AM
UI Frontend	10.4 days?	2/22/22 8:00 AM
Backend	14.7 days?	3/4/22 8:00 AM
Live Voice Translation	4 days?	3/22/22 8:00 AM
Dataset Compilation	4 days?	2/22/22 8:00 AM
Language Model	27.2 days?	**2/27/22 8:00 AM**
CTC Training	23.2 days?	2/27/22 8:00 AM
LID Training	23.2 days?	2/27/22 8:00 AM
Testing	3.7 days?	3/28/22 11:00 AM
System Testing	5.6 days?	4/2/22 8:00 AM
Integration	15.8 days?	**4/1/22 12:00 PM**
LM/Web App Integration	15.8 days?	4/1/22 12:00 PM
Midpoint Demo	0.8 days	4/4/22 8:00 AM
Final Presentation Work	4.8 days	4/16/22 8:00 AM
Final Presentation	0 days?	4/25/22 8:00 AM



Task Division

Webapp Development (Honghao)

1. Web app frontend UI development
    a. Audio recording
    b. Text transcription display
2. Server deployment
    a. Setup app frontend interface and server communication
    b. Added trained model into backend logic

Language Detection and Speech Recognition Models (Marco, Nicholas)

1. Dataset preprocessing for training and validation
2. Training
    a. Literature review for various models and parameter tuning techniques
    b. Developing training pipelines for models
3. Validation
    a. Developing evaluation scripts for multiple models
    b. Compare model performance (average processing speed per word and accuracy)