# **Customisable Speech-to-Text Model for Different Languages** **Individual Project Report**

#### Hema Suganya Ananthakumar

#### Student id: 23883511

**GitHub link -** <https://github.com/uwa-computer-science/project-5-customisable-speech-to-text-team2/tree/23883511_Dev>

### **Tasks**

In this project, my primary responsibilities included:

#### **Data Preprocessing:**

I was responsible for handling real-time audio streaming and noise reduction to ensure clean data input for language identification (LID) and speech-to-text (STT) pipelines.   
This pipeline ensures that the audio data is clean and prepared for accurate language identification and transcription. Below is the detailed data preprocessing pipeline:

#### **Audio Streaming**

* + Real-Time Audio Capture
  + Audio data is captured from a microphone or another input source using a streaming process. The input stream captures audio in small chunks.
  + The stream captures audio with a sampling rate of 16kHz (16,000 samples per second) in mono format to ensure uniformity across all audio input data.
  + Silence Detection: Silence is detected using a threshold based on the Root Mean Square (RMS) of the audio signal. This helps identify breaks or pauses in speech, which can be used to segment the audio into meaningful chunks for further processing.

#### **Noise Reduction**

* + Preprocessing for Clean Audio
  + The captured audio chunks are passed through a noise reduction algorithm. In this case, a library like noisereduce is used.
  + Noise Reduction Algorithm: The algorithm detects and reduces background noise by analyzing the noise profile during silent portions of the audio and subtracting it from the signal.
  + This ensures that the audio passed to the next stages (LID and STT) is clean and free from background noise, which is crucial for improving the accuracy of both language detection and transcription.

#### **Audio Resampling**

* + Resample Audio to 16kHz
  + If the input audio is not already at 16kHz, it is resampled to this standard sampling rate using libraries like torchaudio.transforms.Resample().
  + The resampling ensures compatibility with language identification and speech-to-text models that expect audio at this specific rate.

#### **Audio Conversion to Mono**

* + Ensure Mono Audio Format
  + If the audio is in stereo format (two channels), it is converted to mono. This is done by averaging the two channels into one.
  + Converting to mono reduces the complexity of processing and ensures that models trained on mono audio data receive input in the expected format.

#### **Silence-Based Audio Chunking**

* + Silence Detection and Splitting
  + Audio is segmented into chunks based on detected silences (pauses in speech). The silence detection process identifies points where the RMS (Root Mean Square) of the audio is below a set threshold.
  + When silence is detected for a predefined duration (e.g., 0.03 seconds), the current chunk is considered complete and is queued for processing. This approach ensures that the speech data is split into manageable chunks while eliminating long silent segments.
  + Chunking the audio helps in parallelizing the LID and STT tasks and ensures faster real-time processing.

#### **Queueing Audio Chunks**

* + Asynchronous Queueing
  + The processed audio chunks are placed in a queue for asynchronous processing. The use of a queue enables multiple audio chunks to be processed in parallel, speeding up the overall pipeline.
  + As soon as a chunk is ready, it is sent to the language identification and speech-to-text systems without waiting for the entire audio stream to finish.

#### **STT Model Testing:**

I conducted extensive model testing for the speech-to-text (STT) system, evaluating models such as Whisper (small, medium, large) and SpeechBrain. This testing was crucial in identifying the best-performing model based on performance metrics such as transcription accuracy, response time, and BLEU score generation.

#### **Contribution to LID Pipeline:**

I assisted in the language identification (LID) pipeline by extracting metadata from the Whisper model, which included the probabilities associated with different languages. This helped in selecting the appropriate language for transcription, improving the overall performance of the system.

### **Methods & Results**

#### **Problem Formulation**

Choosing the appropriate approach required an understanding of both audio processing and the challenges of real-time transcription in multiple languages. My decision to focus on Whisper and Speech Brain models was informed by experience and a review of relevant literature. The literature suggested that Whisper’s fine-tuned models (small, medium, large) could deliver high transcription accuracy, while Speech Brain offered flexibility for different languages.

This project deepened my understanding of tackling problems involving complex audio pipelines and integrating LID with STT. It highlighted the importance of balancing model performance with real-time efficiency.

### **Problem Solving**

**Data Preprocessing:**

I used noise reduction techniques (e.g., noisereduce library) to minimize background noise, improving transcription accuracy. I also implemented silence detection algorithms to split the audio stream, ensuring efficient processing. A key takeaway was learning how to handle large audio datasets in real-time, which involved optimizing the processing pipeline to handle streaming efficiently.

The unit STAT3405 [Bayesian] helped significantly in understanding how to structure my data pipeline, especially in dealing with real-time processing and model evaluation.

**STT Model Testing:**

I compared Whisper small, medium, and large models with Speech Brain. Whisper models generally provided superior performance in terms of transcription accuracy, especially for larger models. However, Speech Brain was more responsive for certain language-specific tasks.

The main challenge here was managing GPU resources to run these models efficiently. The ROC-AUC and BLEU scores were used to evaluate the model’s accuracy and robustness. MDS topics like feature selection and evaluation metrics played a significant role in selecting and evaluating these models.

**LID Pipeline Contribution:** I helped integrate Whisper’s metadata extraction into the LID process, extracting probabilities for different languages based on the audio inputs. This helped refine the language selection process and enhance overall accuracy.

### **Ethical, Responsible AI, and Broader Social Impact**

STT and LID systems can have significant social implications, particularly when wrong predictions occur. Incorrect transcriptions or language identification may lead to misunderstandings in critical environments such as healthcare, legal settings, or education. To mitigate these risks, it is essential to include human oversight in critical use cases and implement feedback loops that allow users to correct errors.

An alternative non-AI solution might involve more human-in-the-loop systems where transcription suggestions are verified by individuals.

### **Personal Reflection**

During this project, I faced both technical and non-technical challenges. The main technical challenge involved optimizing audio processing to handle large volumes of streaming data efficiently, which I overcame by implementing silence detection and noise reduction early in the pipeline.

Another challenge was testing various models like Whisper and Speech Brain across different hardware configurations. It was essential to balance model performance with system resources. I would refine the model selection further by conducting more targeted tests on specific languages if given more time.

**Flow Chart**

A diagram of a computer process

Description automatically generated with medium confidence

**References**

#### Hugging Face Platform

* Hugging Face, Inc. (2023). Hugging Face: Models and Datasets for Natural Language Processing and Machine Learning.
* Retrieved from <https://huggingface.co/>

#### Whisper Model

* Radford, A., et al. (2022). Whisper: Pre-trained Multilingual Speech-to-Text Transformer Model.
* Available at: <https://huggingface.co/openai/whisper-small>

#### Wav2Vec2 for Language Identification

* Baevski, A., et al. (2020). Wav2Vec2: A Framework for Self-Supervised Learning of Speech Representations.
* Available at: <https://huggingface.co/facebook/wav2vec2-large-960h>

#### SpeechBrain

* Ravanelli, M., et al. (2021). SpeechBrain: A General-Purpose Speech Toolkit.
* Available at: <https://huggingface.co/speechbrain>

#### BLEU Score Calculation

* Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (pp. 311–318).

#### Noisereduce Library

* Tim Sainburg. (2019). noisereduce: A Python package for reducing noise in signals using spectral gating. GitHub. Available at: <https://github.com/timsainb/noisereduce>

#### Spiking Neural Networks (SNN)

* Bellec, G., Scherr, F., Hajek, E., et al. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. Nature Communications, 11, 3625. DOI: <https://doi.org/10.1038/s41467-020-17440-7>