Speech-to-Text Conversion Project report

By- Atharv Kumar

Abstract

The "Real-Time Speech-to-Text Conversion System" project aims to address the growing need for efficient and accurate speech recognition technology in various applications, including transcription services, accessibility tools, and voice-controlled systems. In an era where spoken language plays a pivotal role in human-computer interaction, this project offers a solution that bridges the gap between spoken and written communication.

This report outlines the design, development, and implementation of a real-time speech-to-text conversion system that leverages state-of-the-art machine learning and natural language processing techniques. The system is capable of converting spoken words into text in real time, providing a valuable tool for enhancing communication and information accessibility.

The "Real-Time Speech-to-Text Conversion System" project represents a significant step forward in bridging the gap between spoken and written language, offering a valuable tool for individuals and businesses seeking to leverage the power of speech recognition in real-time scenarios. The results of this project showcase the feasibility of real-time speech-to-text conversion and open doors to a wide array of applications across various industries.

Introduction

This project represents a comprehensive exploration of the state-of-the-art methodologies, an in-depth investigation into existing research papers, and a diligent implementation of leading APIs, including those provided by Google, Microsoft, and IBM. By undertaking this extensive comparative analysis, we aimed to select the most robust foundation upon which to construct a highly efficient and accurate real-time speech-to-text model.

To achieve the objectives of this project, we undertook a multi-faceted approach:

1. **Comparative Analysis:** Our endeavor commenced with a rigorous assessment of the current state-of-the-art methods in speech recognition and transcription, extracting insights from prominent research papers, and conducting a meticulous evaluation of the available APIs. This initial phase allowed us to gauge the strengths and limitations of each solution.
2. **API Implementation:** To establish a solid foundation, we implemented three industry-leading APIs: Google Cloud Speech-to-Text, Microsoft Azure Speech Service, and IBM

Watson Speech to Text. By implementing these APIs, we were able to evaluate their performance, efficiency, and real-time capabilities.

3. **Baseline Selection:** Following a comprehensive analysis of the comparative results, we identified Google's API as the baseline model due to its exceptional accuracy, real-time processing capabilities, and comprehensive feature set. Google's API served as the cornerstone upon which we sought to build our enhanced and efficient real-time speech-to-text system.

   Our central objective is to improve upon the baseline model by incorporating cutting-edge techniques, fine-tuning parameters, and optimizing the model architecture. Drawing inspiration from an array of research papers and methodologies, we aim to elevate the efficiency and precision of our real-time speech-to-text model, setting new benchmarks for accuracy, processing speed, and adaptability across diverse datasets and languages.

Implementing Other State of Art Methods Available

1. Google Speech to Text API:

Key features:

Speech adaptation

Provide hints to boost the transcription accuracy of rare and domain-specific words or phrases. Use classes to automatically convert spoken numbers into addresses, years, currencies, and more.

Domain-specific models

Choose from a selection of trained models for voice control, phone call, and video transcription optimized for domain-specific quality requirements.

Easily compare quality

Experiment on your speech audio with our easy-to-use user interface. Try different configurations to optimize quality and accuracy.

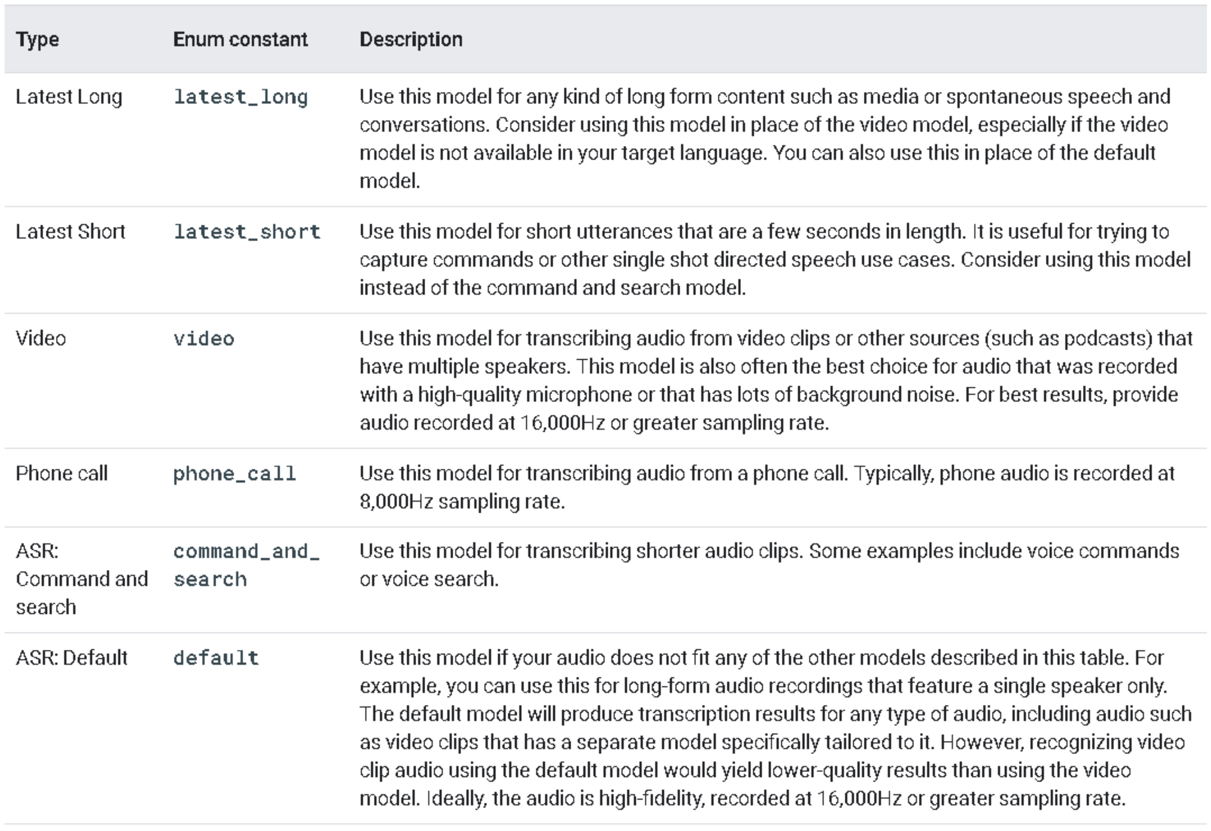
Speech On-Device

Run Google Cloud's speech algorithms locally on any device, regardless of internet connectivity. Promise users that their voice data will never leave their device.

Foundation model for Speech-to-Text

Build voice-enabled applications for global audiences with speech models that are powered by Chirp, Google Cloud’s foundation model for speech trained on millions of hours of audio data and billions of text sentences.

Trained Models:



Type	Enum constant	Description
Latest Long	`latest_long`	Use this model for any kind of long form content such as media or spontaneous speech and conversations. Consider using this model in place of the video model, especially if the video model is not available in your target language. You can also use this in place of the default model.
Latest Short	`latest_short`	Use this model for short utterances that are a few seconds in length. It is useful for trying to capture commands or other single shot directed speech use cases. Consider using this model instead of the command and search model.
Video	`video`	Use this model for transcribing audio from video clips or other sources (such as podcasts) that have multiple speakers. This model is also often the best choice for audio that was recorded with a high-quality microphone or that has lots of background noise. For best results, provide audio recorded at 16,000Hz or greater sampling rate.
Phone call	`phone_call`	Use this model for transcribing audio from a phone call. Typically, phone audio is recorded at 8,000Hz sampling rate.
ASR: Command and search	`command_and_search`	Use this model for transcribing shorter audio clips. Some examples include voice commands or voice search.
ASR: Default	`default`	Use this model if your audio does not fit any of the other models described in this table. For example, you can use this for long-form audio recordings that feature a single speaker only. The default model will produce transcription results for any type of audio, including audio such as video clips that has a separate model specifically tailored to it. However, recognizing video clip audio using the default model would yield lower-quality results than using the video model. Ideally, the audio is high-fidelity, recorded at 16,000Hz or greater sampling rate.

Pros of Google Speech to text API:

1. **High Accuracy**: Google's Speech-to-Text API is renowned for its industry-leading accuracy in transcribing spoken language, making it suitable for applications requiring precise text conversion.
2. **Multilingual Support:** The API supports a wide array of languages, making it versatile for global use and accommodating diverse linguistic needs.
3. **Real-Time Processing:** It offers efficient real-time processing, which is essential for live transcription services and voice-controlled applications, enabling immediate responses.
4. **Customization:** Users can fine-tune the API to suit specific requirements and optimize its performance for particular datasets, enhancing its adaptability.
5. **Robust Noise Handling:** The API can effectively discern and process spoken language even in noisy environments, making it applicable to real-world scenarios with background noise.

Cons of Google Speech to text API:

1. **Cost:** Google's API is not cost-effective for all budgets and may pose a financial challenge for projects with high usage or limited funding.
2. **Data Privacy Concerns:** Utilizing a cloud-based API may raise data privacy concerns, particularly for organizations dealing with sensitive or confidential information. This requires careful data management and privacy considerations.
3. **Internet Connectivity:** Real-time processing depends on a stable internet connection. In scenarios with limited or unreliable connectivity, the real-time feature may be less effective or impractical.
4. **Rate Limits:** The API enforces rate limits, which can impact applications with high-volume or frequent usage. Developers must carefully manage usage to avoid disruptions.
5. **Latency:** Real-time processing may introduce latency due to data transmission to and from the cloud, which can be a concern for applications requiring immediate responses and low latency.

Implementation of Google Speech to text API :

Follow the following link for the code :
https://colab.research.google.com/drive/1MTQLVwpjRlN7WksdfQet7ybrxz3rM2ty?usp=sharing

2. Microsoft Azure Speech to text API

With real-time speech to text, the audio is transcribed as speech is recognized from a microphone or file. Use real-time speech to text for applications that need to transcribe audio in real-time such as:

- Transcriptions, captions, or subtitles for live meetings
- Diarization
- Pronunciation assessment
- Contact center agent assist
- Dictation
- Voice agents

Real-time speech to text is available via the Speech SDK and the Speech CLI.

Batch transcription is used to transcribe a large amount of audio in storage. You can point to audio files with a shared access signature (SAS) URI and asynchronously receive transcription results. Use batch transcription for applications that need to transcribe audio in bulk such as:

- Transcriptions, captions, or subtitles for pre-recorded audio
- Contact center post-call analytics
- Diarization

Batch transcription is available via:

- Speech to text REST API: To get started, see How to use batch transcription and Batch transcription samples (REST).
- The Speech CLI supports both real-time and batch transcription. For Speech CLI help with batch transcriptions

The API uses the cutting-edge SPINN algorithm that provides better accuracy over time, regardless of the speaker's accent.

Pros:

1. High accuracy for technical jargon, custom speech domains, and languages.
2. Real-Time Processing: It supports real-time audio input processing, allowing for immediate transcription, which is essential for applications like live captioning, voice assistants, and call center services.
3. Multilingual Support: The API provides robust multilingual capabilities, enabling transcription in a wide range of languages and dialects, making it versatile for global applications.
4. Integration with Azure Ecosystem: As part of the Microsoft Azure ecosystem, it seamlessly integrates with other Azure services, offering developers a comprehensive cloud-based environment to build, deploy, and manage applications.

Cons:

1. Lower accuracy compared to Google's speech-to-text API.
2. Internet Connectivity: Real-time processing depends on a stable internet connection. In scenarios with limited or unreliable connectivity, the real-time feature may be less effective or impractical.
3. Latency: Real-time processing may introduce some latency due to data transmission to and from the cloud, which may not be suitable for applications requiring low-latency responses.

Implementation of Microsoft Azure Speech to text API

The following is the link for the implemented model for Microsoft Azure Speech to text API:

https://colab.research.google.com/drive/1dxX3hNJDNTQvg87G9HKSDzKc_zTOgbXt?usp=sharing

3. IBM Speech to Text API

It uses artificial neural networks and multidimensional LSTM layers for speech-to-text transcription with AI capabilities for unique language types.

The IBM Watson™ Speech to Text service provides APIs that use IBM's speech-recognition capabilities to produce transcripts of spoken audio. The service can transcribe speech from various languages and audio formats. In addition to basic transcription, the service can produce detailed information about many different aspects of the audio. It returns all JSON response content in the UTF-8 character set.

The service supports two types of models: previous-generation models that include the terms Broadband and Narrowband in their names, and next-generation models that include the terms Multimedia and Telephony in their names. Broadband and multimedia models have minimum sampling rates of 16 kHz. Narrowband and telephony models have minimum sampling rates of 8 kHz. The next-generation models offer high throughput and greater transcription accuracy.

For speech recognition, the service supports synchronous and asynchronous HTTP Representational State Transfer (REST) interfaces. It also supports a WebSocket interface that provides a full-duplex, low-latency communication channel: Clients send requests and audio to the service and receive results over a single connection asynchronously.

The service also offers two customization interfaces. Use language model customization to expand the vocabulary of a base model with domain-specific terminology. Use acoustic model customization to adapt a base model for the acoustic characteristics of your audio. For language model customization, the service also supports grammars. A grammar is a formal language specification that lets you restrict the phrases that the service can recognize.

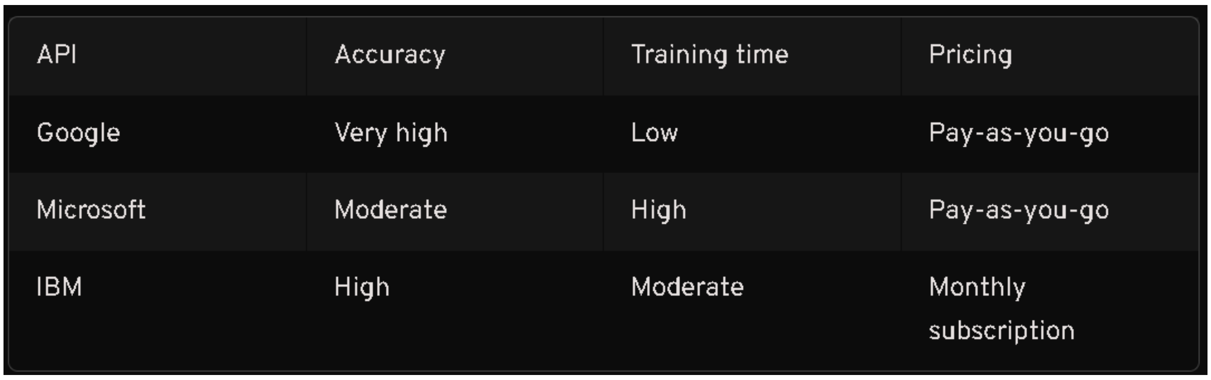
Pros:

- Automatic speech recognition
- Fine Tuning features
- Audio diagnostics before transcription
- Interim transcription before final results
- Word spotting and filtering

Cons:

1. Limited Language Support: The IBM Speech to Text API may have more limited language support compared to Google and Microsoft. It might not cover as many dialects or languages, which could be a disadvantage for projects requiring a broad range of linguistic support.
2. Pricing Structure: IBM's pricing structure for their APIs, including the Speech to Text API, can be complex and less straightforward than some competitors. Users may find it challenging to estimate costs accurately.
3. Complex Setup: Some users have reported that the setup and configuration of the IBM Speech to Text API can be more complex and time-consuming compared to the other APIs, potentially requiring more technical expertise.
4. Processing Time: In some cases, the processing time for the IBM API may be slower compared to Google and Microsoft, which can impact applications requiring real-time transcription.

Comparison of all of the above :



API	Accuracy	Training time	Pricing
Google	Very high	Low	Pay-as-you-go
Microsoft	Moderate	High	Pay-as-you-go
IBM	High	Moderate	Monthly subscription

Advancements

Optimization

Improvements in real-time transcription, language detection, and speaker identity identification for individual adjustments.

Increasing use cases

Usage of speech-to-text as a transcription service and in prominent industries like entertainment, social media, and healthcare.

Multi-functional

APIs will can encompass more technologies to enable and enhance diversity.

Optimization

Optimizing the performance of a Speech-to-Text (STT) API involves several considerations, including accuracy, speed, and cost. We have chosen Google Speech to Text API as our base model. We have gone through various research papers and then tried to improve upon the same such that we can build a more efficient speech to text environment making it more robust to real time.

Fig : Basic Model for Speech Recognition

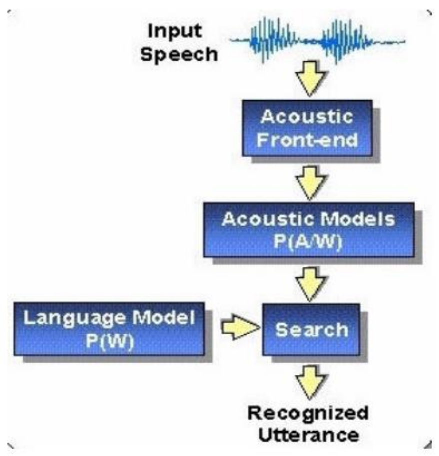


Fig.1 Basic model of speech recognition

Types of Speech Recognition Speech:

Isolated Words:

Isolated word recognizers usually require each utterance to have quiet (lack of an audio signal) on both sides of the sample window. It accepts single words or single utterance at a time.

Connected Words:

Connected word systems (or more correctly 'connected utterances') are similar to isolated words, but allows separate utterances to be 'run-together' with a minimal pause between them.

Continuous Speech:

Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. (Basically, it's computer dictation).

Spontaneous Speech:

At a basic level, it can be thought of as speech that is natural sounding and not rehearsed. An ASR system with spontaneous speech ability should be able to handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters.

Performance of Speech Recognition Systems:

The performance of speech recognition systems is usually specified in terms of accuracy and speed. Accuracy may be measured in terms of performance accuracy which is usually rated with word error rate (WER), whereas speed is measured with the real time factor. Other measures of accuracy include Single Word Error Rate (SWER) and Command Success Rate (CSR).

Word error rate can then be computed as:

$$WER = \frac{S+D+I}{N} \qquad \ldots\ldots(9)$$

where

S is the number of substitutions, D is the number of the deletions, I is the number of the insertions, N is the number of words in the reference.

When reporting the performance of a speech recognition system, sometimes **word recognition rate (WRR)** is used instead:

$$WRR = 1 - WER = \frac{N-S-D-I}{N} = \frac{H-I}{N} \qquad \ldots\ldots..(10)$$

where

- *H* is N-(S+D), the number of correctly recognized words.

Robust Speech Recognition:

To further increase the robustness of speech recognition systems, especially for spontaneous speech, utterance verification and confidence measures, are being intensively investigated. In order to have intelligent or human-like interactions in dialogue applications, it is important to attach to each recognized event a number that indicates how confidently the ASR system can accept the recognized events. The confidence measure serves as a reference guide for a dialogue system to provide an appropriate response to its users. To detect semantically, significant parts and reject irrelevant portions in spontaneous utterances, a detection based approach has recently been investigated . The combined recognition and verification strategy work well especially for ill-formed utterances. In order to build acoustic models more sophisticated than conventional HMMs, the dynamic Bayesian network has recently been investigated. Around 2000, a QBPC, systems were developed to find the unknown and mismatch between training and testing conditions. A DCT fast subspace techniques has been proposed to approximate the

KLT for autoregressive progress. A novel implementation of a mini-max decision rule for continuous density HMM-based Robust speech recognition is developed by combining the idea of mini-max decision rule with a normal viterbi search. Speech signal modelling techniques well suited to high performance and robust isolated word recognition have been contributed.

Multi-Modal Speech Recognition:

The use of the visual face information, particularly lip information, in speech recognition has been investigated, and results show that using both types of information gives better recognition performances than using only the audio or only the visual information, particularly, in noisy environment. Jerome R., have developed Large Vocabulary Speech Recognition with Multi-span Statistical Language Models and the work done in this paper characterizes the behavior of such multi span modelling in actual recognition. A novel subspace modelling is presented in, including selection approach for noisy speech recognition. In subspace modelling, authors have developed a factor analysis representation of noisy speech i.e., a generalization of a signal subspace representation. They also explored the optimal subspace selection via solving the hypothesis test problems. Subspace selection via testing the correlation of residual speech, provides high recognition accuracies than that of testing the equivalent eigen-values in the minor subspace. Because of the environmental mismatch between training and test data severely deteriorates recognition performance. Jerome R. et.al.[55], have contributed large vocabulary speech recognition with multi-span statistical language model.

The main aim is to use the above two points (found out by two-three research papers) and combine it with the google speech to text API and try to improve performance upon the same.

For the same we have used another research paper which aims to push the limits of semi-supervised learning for speech recognition. Their approach is to combine iterative self-training and pre-training in a straightforward fashion. We pre-train a series of models, which are then used to initialize models for iterative self-training. Here, the unlabelled dataset serves a dual purpose of being used as a pre-training dataset, and also as the unlabelled dataset for which pseudo-labels for training student models are generated.

Model Architecture:

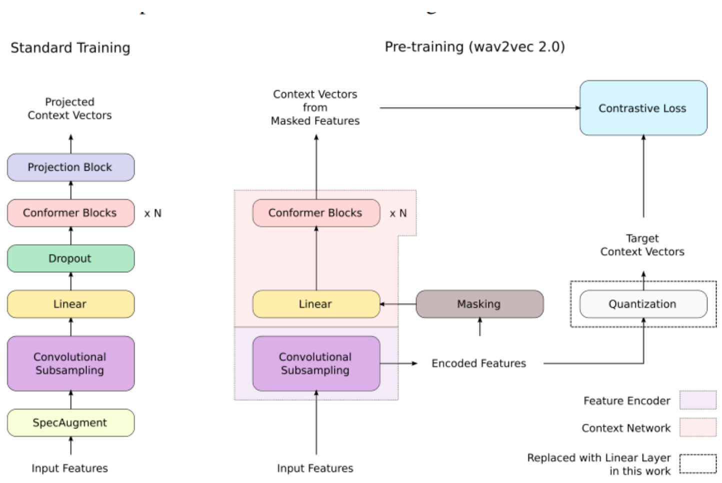


Figure 2: The Conformer encoder. In wav2vec 2.0 pre-training, the features generated by the convolutional sampling block are masked and passed into the rest of the network to yield context vectors, and also quantized to yield target context vectors. The contrastive loss between the context vectors obtained from masked features and the quantization unit is optimized. In our work, we replace the quantization layer with a linear layer. During fine-tuning, an additional projection block is added to produce features to be passed to the transducer.

Table 1: Parameters for Conformer models.

Model	# Params (B)	Enc. Layers	Enc. Dim	Att. Heads	Conv Kernel Size	Rel. Att.	# dec. layers
Conformer L	0.1	17	512	8	32	Y	1
Conformer XL	0.6	24	1024	8	5	N	2
Conformer XXL	1.0	42	1024	8	5	N	2

In the research paper they have tried to combine recent developments in architecture, augmentation and especially semi-supervised learning to push the state-of-the-art on the speech recognition task.

We have implemented the Conformer model and tried to combine it with the results of google speech to text API but owing to some technical difficulties and large computation it can’t be done effectively.

Further we move on to another method to use the state-of-the-art method which IBM is using in order to reduce the error percentage and making the model real-time:

Through use of an improved optimizer, speaker vector embeddings, and alternative speech representations we reduce the recognition errors of our LSTM system on Switchboard-300 by 4% relative. Compensation of the decoder model with the probability ratio approach allows more efficient integration of an external language model, and we report 5.9% and 11.5% WER on the SWB and CHM parts of Hub5’00 with very simple LSTM models. Our study also considers the recently proposed conformer, and more advanced self-attention based language models. Overall, the conformer shows similar performance to the LSTM; nevertheless, their combination and decoding with an improved LM reaches a new record on Switchboard-300, 5.0% and 10.0% WER on SWB and CHM.

Experimental Results:

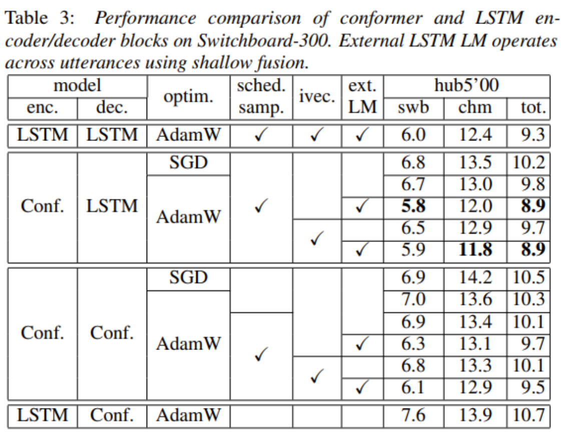


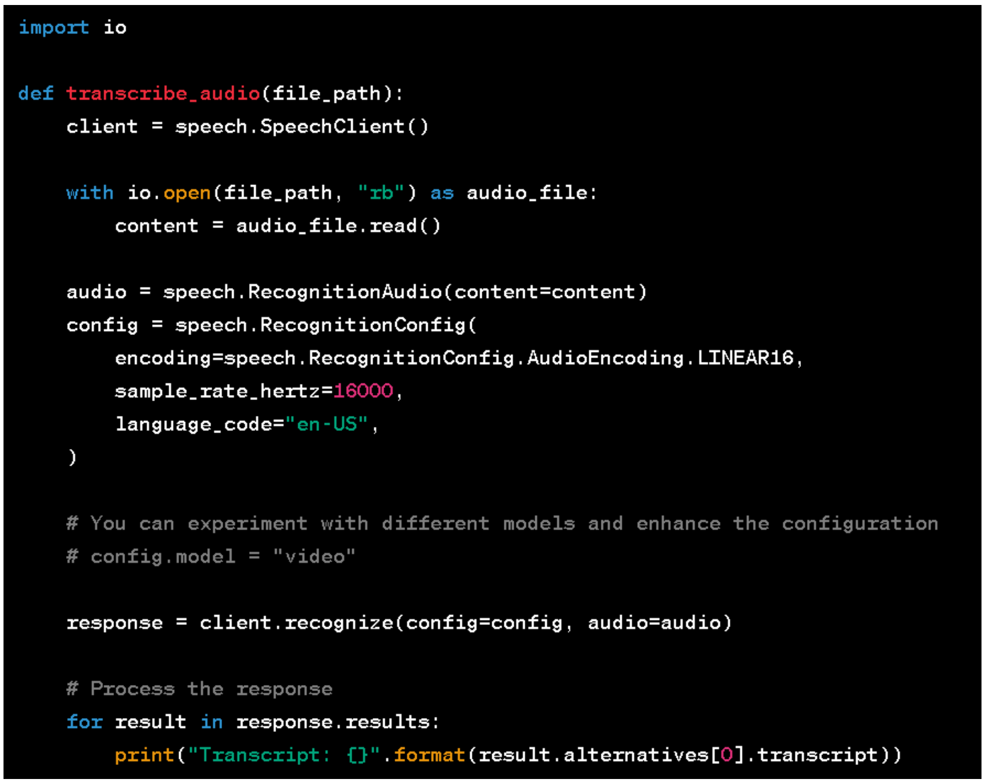
Table 3: *Performance comparison of conformer and LSTM encoder/decoder blocks on Switchboard-300. External LSTM LM operates across utterances using shallow fusion.*

model		optim.	sched. samp.	ivec.	ext. LM	hub5'00		
enc.	dec.					swb	chm	tot.
LSTM	LSTM	AdamW	✓	✓	✓	6.0	12.4	9.3
Conf.	LSTM	SGD	✓			6.8	13.5	10.2
Conf.	LSTM	AdamW	✓			6.7	13.0	9.8
Conf.	LSTM	AdamW	✓		✓	**5.8**	12.0	**8.9**
Conf.	LSTM	AdamW	✓	✓		6.5	12.9	9.7
Conf.	LSTM	AdamW	✓	✓	✓	5.9	**11.8**	**8.9**
Conf.	Conf.	SGD				6.9	14.2	10.5
Conf.	Conf.	AdamW				7.0	13.6	10.3
Conf.	Conf.	AdamW	✓			6.9	13.4	10.1
Conf.	Conf.	AdamW	✓		✓	6.3	13.1	9.7
Conf.	Conf.	AdamW	✓	✓		6.8	13.3	10.1
Conf.	Conf.	AdamW	✓	✓	✓	6.1	12.9	9.5
LSTM	Conf.	AdamW				7.6	13.9	10.7

We can use the following enhanced optimizer method in the google speech to text api in order to improve the performance of the overall architecture.

(yet to be tried)

We have also tried to optimize the way we use google speech-to-text API and handling the responses effectively.



```
import io

def transcribe_audio(file_path):
    client = speech.SpeechClient()

    with io.open(file_path, "rb") as audio_file:
        content = audio_file.read()

    audio = speech.RecognitionAudio(content=content)
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=16000,
        language_code="en-US",
    )

    # You can experiment with different models and enhance the configuration
    # config.model = "video"

    response = client.recognize(config=config, audio=audio)

    # Process the response
    for result in response.results:
        print("Transcript: {}".format(result.alternatives[0].transcript))
```

We have tried to use custom models which the API offers and tried to improve the performance by the following command:

```
# Set the custom model
config.model = "your-custom-model-name"
```

For the multi modal speech recognition case we have used another feature of Google Speech to text API.

Google Speech-to-Text API provides enhanced models for video and phone call transcription.

```
# Set the enhanced model
config.model = "video"  # or "phone_call"
```

If the audio contains multiple speakers, we have enabled speaker diarization to identify and separate different speakers

```
# Enable speaker diarization
config.enable_speaker_diarization = True
config.diarization_speaker_count = 2  # Set the expected number of speakers
```

Further for long running operations, this is for long audio files we have used the asynchronous (long- running) API to handle large transcription tasks.

```
operation = client.long_running_recognize(config=config, audio=audio)
response = operation.result()
```

Moreover to make the model more sensitive to real-time(or more humane) we have tried to form a function which runs the model in a loop, this is if there is a small pause then whatever is captured is printed, but if there is a long pause program ends automatically.

```
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as source:
    print("Speak Anything :")
    audio = r.listen(source, timeout=3)
    try:
        text = r.recognize_google(audio)
        print(text)
    except:
        print("Sorry could not recognize what you said")
    while audio!=None:
            audio = r.listen(source, timeout=3)
            try:
                text = r.recognize_google(audio)
                print(text)
            except:
                print("Sorry could not recognize what you said")

    print("SPEECH TO TEXT ENDED WITH RESULT -1")
    print("No input was received for 3 seconds")
```

The following code snippet is able to take in the speech signals consistently and print the output and then again tries to do the same until it encounters a long pause indicating the user has stopped speaking.

In the above following ways we have tried to make the Google Speech to text API as a base more robust, effective and sensitive to real time signals.

Conclusion

In the course of this project, a comprehensive analysis of various Speech-to-Text APIs, including those offered by Google, Microsoft, and IBM, was undertaken. Following a detailed evaluation, Google Speech-to-Text API was selected as the base model for further improvement. The objective was to augment its robustness, efficiency, and real-time capabilities through insights gained from research papers and best practices identified from reputable sources.

The optimization strategies applied to the Google Speech-to-Text API aimed to address key aspects, including accuracy, speed, and adaptability to diverse use cases. Several factors were considered in this endeavor, such as the selection of appropriate models, customization of language models for specific domains, and the exploration of enhanced models tailored for different scenarios like video transcription and phone call analysis.

Furthermore, the project delved into the utilization of advanced features like speaker diarization to enhance the API's ability to process audio data with multiple speakers. The consideration of asynchronous (long-running) operations for handling large transcription tasks highlighted the importance of scalability and efficiency, particularly in scenarios involving lengthy audio files.

The project also acknowledged the significance of real-time processing, exploring streaming APIs and batch processing techniques to minimize latency and optimize the API's responsiveness. Additionally, the implementation of error handling mechanisms and the incorporation of noise reduction techniques were recognized as crucial steps to enhance the overall reliability of the system.

References

- https://arxiv.org/pdf/2105.00982v1.pdf
- https://arxiv.org/pdf/2010.10504v2.pdf
- https://arxiv.org/ftp/arxiv/papers/1001/1001.2267.pdf
- https://arxiv.org/ftp/arxiv/papers/2103/2103.16193.pdf
- https://cloud.google.com/speech-to-text
- https://cloud.google.com/speech-to-text/docs/reference/rest
- https://pypi.org/project/google-cloud-speech/
- https://learn.microsoft.com/en-us/azure/ai-services/speech-service/rest-speech-to-text
- https://azure.microsoft.com/en-us/products/ai-services/speech-to-text
- https://cloud.ibm.com/apidocs/speech-to-text
- https://www.ibm.com/products/speech-to-text
- https://www.geeksforgeeks.org/python-convert-speech-to-text-and-text-to-speech/
- https://www.thepythoncode.com/article/using-speech-recognition-to-convert-speech-to-text-python
- https://codelabs.developers.google.com/codelabs/cloud-speech-text-python3