Customer Care Call Analytics Powered by ASR & LLM for Hindi Language

DISSERTATION

Submitted in partial fulfillment of the requirements of the

Degree: M.Tech in Artificial Intelligence & Machine Learning

By

Tanmay Jain

2022AA05306

Under the supervision of

Shakti P. Rath

(Principal Architect)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

(June, 2024)

Abstract

This dissertation focuses on developing a cost-effective and efficient Automatic Speech Recognition (ASR) system specifically designed for customer care centers serving the Hindi language. The primary aim is to address the challenges posed by low-resource languages, noisy environments, and significant variations in speech. This project utilizes a custom-trained ASR model, an open-source Speaker Diarization (SD) tool, and open-source Large Language Models (LLMs) to deliver comprehensive call analytics. All components are deployed on-premises, eliminating the need for any paid APIs.

The ASR system will be trained using open-source and internal custom data to support low-resource languages like Hindi. This approach ensures higher accuracy and relevance in transcriptions tailored specifically to the domain, which is crucial for the operational needs of customer care centers. The ASR model is designed to utilize advanced conformer-based architecture, ensuring the model can learn complex hidden patterns. A significant aspect of the system is its integration of SD, which accurately identifies and separates different speakers within a call. This feature enhances the clarity of transcriptions and ensures that dialogues are correctly attributed, facilitating better understanding and analysis of customer interactions. The diarization process is critical in multi-speaker environments typical of customer care settings, where distinguishing between customer and agent speech is essential for accurate data capture and subsequent analysis.

An LLM is employed for call analytics to enhance the utility of the ASR outputs further. The LLM processes the transcriptions to generate actionable insights, such as identifying common customer issues, sentiment analysis, and performance metrics for customer service representatives. This analytic capability is instrumental in improving service quality, training, and strategic decision-making within customer care operations. By avoiding the use of paid APIs and utilizing open-source technologies, this dissertation aims to deliver a robust, scalable ASR solution that can be adopted by customer care centers without incurring significant expenses.

In summary, this dissertation presents a comprehensive solution for enhancing customer care centers through a tailored ASR system that addresses the operational challenges of the Hindi language.

Key Words: Automatic Speech Recognition (ASR), Speaker Diarization (SD)**,** Customer Care Centers Analytics, Large Language Models (LLMs), Call Analytics

List of Abbreviations

AM	Acoustic Model
ASR	Automatic Speech Recognition
BPE	Byte Pair Encoding
CCAS	Customer Care Analysis System
CRM	Customer Relationship Management
CTC	Connectionist Temporal Classification
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
LLM	Large Language Model
LM	Language Model
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Coefficients
NFC	Normalization Form C
NFD	Normalization Form D
NLP	Natural Language Processing
NLU	Natural Language Understanding
Regex	Regular expressions
RNN	Recurrent Neural Network
Rnn-T	Recurrent Neural Network Transducer
SD	Speaker Diarization
STFT	Short-Time Fourier Transform
WER	Word Error Rate

List of Tables

List of Figures

Table of Contents

1. Chapter 1: Introduction

1.1. Background

Customer care centers are crucial touchpoints for businesses, providing support and resolving issues for customers. In India, these centers handle a significant volume of calls in various languages, each with its own set of dialects and accents. Efficiently managing these calls and deriving actionable insights is a major challenge. Traditional methods involve manually reviewing call recordings, which is time-consuming, error-prone, and not scalable.

As the volume of customer interactions continues to grow, the limitations of manual review become increasingly apparent. Companies struggle to keep up with the sheer amount of data, often missing out on critical insights buried within the conversations. This not only hampers the ability to respond promptly to customer needs but also affects the quality of service, revenue, and operational efficiency.

1.2. Problem Statement

The specific problem addressed in this research is the inefficiency and inadequacy of current methods for analyzing customer care call recordings for Indian languages, with a particular focus on Hindi. Existing approaches predominantly rely on manual review processes, which are not only labor-intensive but also lack scalability. These methods are constrained by the high volume of calls and the complex linguistic landscape of India, which includes a rich variety of accents, dialects, and regional variations.

The manual review process is time-consuming and prone to errors, leading to delays in generating actionable insights. This inefficiency results in missed opportunities for improving customer service and optimizing call center operations. Additionally, the reliance on manual methods means that insights are often not available in a timely manner, which affects the ability of businesses to make informed decisions based on current data.

Given these challenges, there is a clear need to develop a cost-effective, on-premise *Customer Care Analysis System (CCAS)* specifically tailored to handle the nuances of Hindi. Such a system should be capable of accurately processing and analyzing large volumes of call data, providing timely and precise insights into agent-customer interactions. By addressing these needs, the proposed CCAS aims to enhance the efficiency of customer care centers, improve the quality of service, and support better decision-making through advanced analytics.

1.3. Objectives of the Study

This study focuses on three key components: Speaker Diarization (SD), Automatic Speech Recognition (ASR), and Large Language Models (LLM) for customer care call analytics. Below is a detailed overview of the utility and requirements of each module:

1.3.1. Integrate Speaker Diarization

Speaker diarization in customer care can greatly enhance call analytics by accurately identifying and distinguishing between multiple speakers in a conversation (often an agent, and a customer). This separation enables more precise transcriptions, which are crucial for analyzing customer interactions and understanding customer sentiment. By isolating agent and customer speech, companies can better monitor call quality, assess agent performance, and provide targeted training. Additionally, it improves sentiment analysis accuracy, offering deeper insights into customer experiences and agent effectiveness.

1.3.2. Develop a Robust ASR Model

Developing a robust ASR model offers substantial benefits for customer care call analysis. ASR systems can convert spoken language into text, enabling easy review and documentation of customer interactions. This transcription capability allows businesses to assess customer sentiment, identify feedback as positive, negative, or neutral, and pinpoint areas for improvement. Moreover, ASR facilitates keyword extraction to uncover common issues and frequently discussed topics, aiding in trend tracking and data-driven decision-making. Automated reports and summaries generated from call transcripts save time and effort, while these transcripts serve as valuable resources for training and enhancing agent performance. Integrating ASR with analytics tools provides businesses with actionable insights, thereby improving overall customer service.

The following outlines the requirements for the ASR system:

a. **Implement Noise Robustness Techniques in ASR**

To ensure effective performance, incorporating noise robustness techniques into the ASR system is crucial, especially in customer care environments where audio quality can be inconsistent. Techniques like data augmentation, which introduces simulated noise during training, help the ASR system learn to recognize speech in various noisy conditions.

b. **Deploy the ASR System On-Premise**

Deploying the ASR system on-premise offers significant advantages, including enhanced data privacy and security by allowing full control over sensitive customer

data, which is essential for regulatory compliance. The on-premise deployment also permits deeper customization, making the system more adept at recognizing industry-specific jargon and regional accents. Integration with existing internal systems, such as Customer Relationship Management (CRM) and ticketing systems, becomes more seamless, fostering a cohesive workflow. Additionally, local deployment optimizes hardware resources, reduces latency, and provides offline capabilities for areas with unreliable connectivity. Customization for local languages and accents is more feasible, improving recognition accuracy.

c. Support Streaming Mode in ASR

Supporting streaming mode in ASR further enhances customer care analysis by enabling real-time transcription of interactions. This immediate access to conversation data allows for instant feedback and support suggestions, boosting response efficiency and customer satisfaction. Live transcriptions facilitate real-time sentiment analysis, helping tailor responses based on customer emotions. Continuous monitoring ensures service quality standards are met, and automated reporting generates insights to identify trends and areas for improvement. Streaming ASR is particularly effective for handling high call volumes, managing multiple conversations simultaneously, and integrating with CRM systems for accurate customer record-keeping. Real-time transcription data also supports the training of customer service representatives, providing immediate feedback on communication effectiveness and areas for improvement.

1.3.3. Integrate LLMs for Advanced Call Analytics

Integrating an LLM into a customer care analysis system can significantly enhance its capabilities. LLMs excel in Natural Language Understanding (NLU), allowing the system to better comprehend and process complex customer interactions and varied language styles. This improved understanding extends to sentiment analysis, where LLMs can detect customer emotions and satisfaction levels, providing valuable insights into customer experiences. Additionally, LLMs are adept at summarizing lengthy interactions, making it easier to extract key information and identify trends. By analyzing patterns and trends in customer interactions, LLMs can uncover emerging issues and areas for improvement. Furthermore, LLMs can personalize responses based on customer history and preferences, enhancing the overall customer experience. They are also useful for detecting anomalies in customer data, which can indicate systemic problems or areas needing further investigation.

We will incorporate LLMs for the following analysis:

a. Sentiment Analysis

i. Purpose: Determine the emotional tone of conversations to assess customer satisfaction.
ii. How It Helps:
    1. Real-time Feedback: By analyzing the sentiment of calls in real-time, customer service teams can immediately understand how customers feel about their service or product.
    2. Identify Trends: Regular sentiment analysis can highlight recurring issues or improvements over time, helping to gauge overall customer satisfaction and loyalty.
    3. Prioritize Responses: Calls with negative sentiment can be flagged for immediate follow-up, ensuring that unhappy customers receive prompt attention and resolution.

b. Problem Resolution Status
    i. Purpose: Identify whether the customer's issue was resolved during the call.
    ii. How It Helps:
        1. Measure Efficiency: Assessing whether issues are resolved in a single call helps evaluate the effectiveness of the support team and the efficiency of the resolution process.
        2. Customer Satisfaction: Unresolved issues often lead to customer dissatisfaction. Tracking resolution status ensures that problems are addressed, which can lead to higher customer satisfaction rates.
        3. Training Needs: If many calls end without resolution, it may indicate a need for additional training or resources for agents.

c. Keyword Detection
    i. Purpose: Recognize specific words or phrases, such as greetings or product-related terms, to ensure adherence to protocols.
    ii. How It Helps:
        1. Protocol Adherence: Ensures that agents are following established scripts or protocols, such as greetings, disclaimers, or upselling strategies.
        2. Quality Control: By monitoring for key phrases, the system can help ensure that all necessary information is covered during calls, improving the consistency of customer interactions.

d. Agent Behavior Analysis

i. Purpose: Assess the performance and behavior of customer service agents to ensure quality service.
ii. How It Helps:
    1. Performance Metrics: Analyze aspects such as tone, politeness, and adherence to procedures to evaluate agent performance.
    2. Identify Training Needs: Patterns in behavior or performance issues can help pinpoint areas where additional training or support may be needed.
    3. Recognition and Improvement: Positive behavior can be recognized and rewarded, while negative behavior can be addressed to improve overall service quality.

e. Enhancement Opportunities
    i. Purpose: Suggest ways to improve customer service and enhance call center operations.
    ii. How It Helps:
        1. Continuous Improvement: Insights from call analysis can highlight areas for improvement, such as reducing average handle time, improving first-call resolution rates, or enhancing agent communication skills.
        2. Customer Insights: Understanding customer needs and pain points can help in developing new strategies to enhance the customer experience.

f. New Product Features
    i. Purpose: Identify potential new product features or services based on customer feedback and interactions.
    ii. How It Helps:
        1. Product Development: Analyzing calls can reveal common requests or suggestions from customers, providing valuable input for product development teams.
        2. Competitive Edge: By aligning product features with customer expectations, companies can stay ahead of competitors and offer more tailored solutions.

Incorporating these elements into a customer call analysis system can significantly enhance the effectiveness of customer service operations, leading to better customer experiences, improved agent performance, and more strategic decision-making.

1.4. Scope of the Research

This study seeks to investigate and integrate open-source SD libraries to distinguish between speakers effectively. The development of a custom ASR model specifically for the Hindi language will be undertaken, with capabilities to manage dialectal and accent variations while maintaining robust performance in noisy environments. The system will be deployed on-premise, operating in streaming mode without dependency on paid APIs. Additionally, it will leverage LLMs for in-depth call analytics, providing valuable insights as outlined in section 1.3.3.

Limitations:

a. Speaker Diarization Resources: Most existing research and tools for speaker diarization are tailored to high-resource languages like English. As a result, many libraries primarily support English, leaving a scarcity of open-source libraries, tools for other languages, often with limited accuracy.

b. Challenges with Hindi: Hindi is considered a low-resource language compared to high-resource languages like English, which benefit from extensive datasets spanning hundreds of thousands of hours. Achieving high accuracy in Hindi can be challenging due to the limited data available. Moreover, advanced models such as conformers are data-hungry and require vast datasets to capture subtle nuances. Additionally, the diverse accents and dialects across different regions in India further complicate the creation of a robust dataset, potentially affecting the model's performance across various speakers.

c. Limitations of LLMs: While LLMs excel in English, their training in Hindi is comparatively limited. This can lead to reduced accuracy when processing Hindi data directly. To address this, we may need to translate Hindi data into English before feeding it to the LLMs for insights.

This research focuses on exploring open-source SD and LLM models at a fundamental level. The goal is to establish a functional pipeline for customer care analysis rather than achieving high accuracy in SD and LLM technologies. The primary objective is to develop an end-to-end customer care analysis workflow.

1.5. Significance of the Study

This study primarily impacts businesses by revolutionizing their approach to customer care through the integration of SD, ASR, and LLMs. Here's how:

a. Enhanced Customer Experience

By accurately attributing spoken content through SD, businesses can precisely analyze how each participant contributes to the interaction. This clarity helps in understanding customer needs more effectively and improving how issues are resolved, leading to a more satisfying customer experience.

b. Operational Efficiency

ASR automates the transcription of call data, converting spoken language into text quickly and accurately. This reduces manual effort and errors, allowing businesses to access and analyze performance metrics like response times and resolution rates more efficiently. Streamlined processes and quicker data access translate into significant operational efficiencies and cost savings.

c. Deep Insights and Personalization

LLMs provide advanced contextual analysis and sentiment evaluation, uncovering deeper insights into customer emotions and satisfaction levels. This ability to understand the nuances of customer interactions enables businesses to personalize their services and address specific concerns more effectively, enhancing overall service quality.

d. Cost Reduction

The automation of transcription and analysis through SD, ASR, and LLMs lowers administrative and labor costs associated with manual data processing. The reduced need for human intervention not only cuts operational expenses but also improves productivity.

e. Data Privacy and Compliance

On-premise deployment of these technologies ensures that customer data remains secure and complies with data protection regulations. This focus on data privacy helps build trust with customers and safeguards sensitive information.

f. Strategic Advantage

By leveraging the combined insights from SD, ASR, and LLMs, businesses can identify emerging trends, uncover new opportunities, and refine their strategies. This data-driven approach supports informed decision-making and strategic planning, positioning companies for long-term success and competitive advantage.

In summary, this study impacts businesses by enhancing customer experience, improving operational efficiency, providing deep insights, reducing costs, ensuring data privacy, and offering strategic advantages. The integration of SD, ASR, and LLMs creates a powerful framework for optimizing customer care and driving business growth.

2. Chapter 2: Literature Review

2.1. Theoretical Framework

The theoretical framework of this research encompasses several key areas: ASR, SD, noise robustness in ASR, and analytics with LLMs.

2.1.1. Speaker Diarization

SD is the process of segmenting an audio stream into homogeneous segments according to the identity of the speaker, effectively answering the question, "Who spoke when?" This involves identifying and distinguishing between different speakers in an audio recording, which is essential in multi-speaker environments such as meetings, interviews, and call centers.

The process typically involves several steps. First, the audio signal is divided into short, overlapping frames. Features such as Mel Frequency Cepstral Coefficients (MFCC) are extracted from these frames, capturing the essential characteristics of the audio signal. These features are then clustered, often using methods like Gaussian Mixture Model (GMMs) or more advanced techniques like Deep Neural Networks (DNN), to group frames that are likely spoken by the same speaker.

Next, a segmentation algorithm identifies boundaries where the speaker changes. This can be achieved using Hidden Markov Model (HMM) or Recurrent Neural Network (RNN). Finally, the segments are re-clustered to refine the speaker labels, ensuring that all segments attributed to the same speaker are grouped. The outcome is a timeline of the audio indicating segments of speech and the corresponding speakers.

2.1.2. Automatic Speech Recognition

ASR is the technology that converts spoken language into text. It has become integral to various applications, including virtual assistants, transcription services, and voice-controlled interfaces.

The process begins with capturing the audio signal, which is then divided into small segments called frames. Acoustic features, such as MFCCs, are extracted from these frames. These features are then input into an AM, which maps the features to phonemes, the basic sound units of a language.

The phonemes are then processed by an LM, which predicts the most likely sequence of words based on the sequence of phonemes and the context. The language model leverages statistical or neural network-based approaches to determine word

probabilities, ensuring that the recognized words form a coherent and contextually appropriate sequence.

Finally, a decoder integrates the outputs of the acoustic and language models to produce the final text transcription. Advanced ASR systems use deep learning architectures, such as and RNNs, and Transformers, to improve accuracy, especially in noisy environments or with diverse accents and languages.

2.1.3. Large Language Models

LLMs are a class of AI models designed to understand, generate, and manipulate human language at a high level. These models are typically based on deep learning architectures, with the Transformer model being a prominent example. LLMs, such as GPT-4, are trained on vast amounts of text data to learn the statistical patterns of language.

The training process involves feeding the model large corpora of text, enabling it to learn word distributions, syntax, semantics, and context. Transformers use attention mechanisms to weigh the importance of different words in a sentence, allowing the model to understand the context and relationships between words over long distances in text.

Once trained, LLMs can perform various tasks, including text generation, translation, summarization, and question-answering. They generate text by predicting the next word in a sentence based on the context of the preceding words, using the learned probabilities from the training data. This ability to generate coherent and contextually relevant text makes LLMs powerful tools for numerous applications in Natural Language Processing (NLP).

In summary, SD identifies who is speaking and when, ASR converts spoken language into written text, and LLMs enable advanced language understanding and generation, forming the backbone of modern NLP systems.

2.2. Review of Related Technologies

2.2.1. Existing SD Systems & Libraries

The following are the SD libraries and toolkits that we reviewed for the integration of the SD system:

a. pyAudioAnalysis

A Python library offering various audio analysis functionalities, including basic SD capabilities. It provides an accessible entry point for simple diarization tasks but may lack the advanced features required for complex, multi-speaker environments.

b. LIUM SpkDiarization

An older, yet still relevant toolkit specifically designed for SD. While it provides solid performance, it may require updates and customization to integrate with modern ASR systems and environments.

c. NeMo Toolkit

NVIDIA's NeMo toolkit provides state-of-the-art models for SD, leveraging deep learning techniques for high accuracy. It integrates seamlessly with other NeMo components, offering a comprehensive solution for complex, multi-speaker scenarios.

d. Opensource implementation

We explored open-source implementations of SD for the Hindi language and identified a robust solution. This repository presents a comprehensive approach to SD using advanced deep-learning techniques. It employs a two-stage method that combines speaker embedding extraction with clustering to achieve precise speaker identification. Additionally, the repository provides clear usage instructions, making it a valuable resource for researchers and practitioners in the field of speech processing.

e. Whisper

In addition to open-source solutions, we explored Whisper's SD for Hindi. Whisper uses advanced neural networks to differentiate speakers, handling diverse accents and dialects effectively accurately. Its user-friendly interface and thorough documentation make it a valuable tool for enhancing SD in Hindi language applications.

2.2.2. Existing ASR Systems & Toolkits

The following are the ASR systems and toolkits that we reviewed for the development of our ASR system:

a. Google Speech-to-Text

A cloud-based ASR service known for its high accuracy and extensive language support. It utilizes advanced machine learning algorithms to provide real-time transcription services. However, it is heavily reliant on internet connectivity, which can pose issues in areas with unstable internet. Additionally, it incurs significant costs, especially for large-scale deployments.

b. Microsoft Azure Speech Service

Provides comprehensive ASR capabilities with strong language support and customization features. It integrates well with other Azure services, making it a versatile choice for enterprises. However, it involves costs and data privacy concerns for cloud-based usage, similar to other commercial ASR services.

c. Amazon Alexa

Offers advanced voice recognition and processing capabilities, widely used in consumer devices. Its cloud dependency poses privacy and cost issues for large-scale, enterprise-level deployments. It is primarily designed for consumer interactions, which may limit its applicability in specialized enterprise settings.

d. Kaldi

An open-source ASR toolkit providing flexible and powerful tools for ASR research and development. It offers extensive customization and has been widely adopted in the research community. However, it requires significant expertise to set up and maintain, making it less accessible for organizations without specialized technical staff.

e. Mozilla DeepSpeech

An open-source ASR engine based on deep learning, providing a good balance between performance and ease of use. It supports offline processing and can be customized for specific languages and environments, although it may require substantial computational resources.

f. Whisper

Developed by OpenAI, Whisper is an advanced ASR model that excels in transcription accuracy and language support. It offers robust performance in diverse environments but may be complex and resource-intensive for on-premise deployment.

g. NeMo Toolkit

NVIDIA’s NeMo toolkit provides state-of-the-art ASR models, including conformer architectures, offering high accuracy and performance. It supports both training and inference, making it suitable for research and production use. Its flexibility allows for customization to specific needs, although it requires familiarity with NVIDIA’s ecosystem.

h. Wav2vec 2.0

A self-supervised ASR model developed by Facebook AI Research, known for its robustness and ability to learn from unlabeled data. This makes it particularly effective for low-resource languages, where labeled data is scarce. However, its training process is computationally intensive.

i. SpeechBrain

An open-source ASR toolkit designed for research and development, offering a flexible and extensible platform for building custom ASR systems. It supports a wide range of tasks and models but may require significant effort to customize for specific applications.

j. WeNet

An open-source ASR toolkit that emphasizes real-time ASR with a focus on providing efficient and accurate models suitable for deployment on edge devices. It offers a practical solution for low-latency applications, although its performance may vary depending on the complexity of the language and environment.

2.3. Review of Papers

Making a Case for Speech Analytics to Improve Customer Service Quality: Vision, Implementation, and Evaluation [1]

- This paper presents a strategic initiative that examines the use of speech analytics to improve customer service quality at call centers. It discusses the vision, implementation, and evaluation of speech analytics in enhancing customer service quality.

The Impact of Call Center Employees' Customer Orientation Behaviors on Service Quality [2]

- This study investigates the impact of call center employees' customer orientation behaviors on service quality. It highlights the importance of speech analytics in understanding customer interactions and improving service quality.

Text Summarization for Call Center Transcripts [3]

- Text summarization of call center transcripts is essential for detailed analysis but is challenging due to context switching, cross talk, and transcription errors from ASR systems. This work develops a summarization model for on-premise deployment at call centers by finetuning pre-trained open-source LLMs, using reference summaries generated by GPT-3.

Leveraging AI via speech-to-text and LLM integration for improved healthcare decision-making in primary care [4]

- Recent research highlights the potential of AI, particularly ASR and LLMs, to alleviate workloads and streamline documentation in healthcare. A proposed framework for general practices includes ASR for transcriptions, decision support systems, and automated prescription email generation. Qualitative and quantitative analyses

demonstrate improvements in reducing administrative burdens and enhancing medic-patient relationships, potentially improving diagnostic outcomes. This approach can be adapted for customer care call analysis to automate call summaries, improve documentation accuracy, support real-time decision-making, and enhance customer interactions.

2.4. Analysis of Existing SD Solutions

There are very few SD systems available for the Hindi language, with most open-source tools and libraries primarily supporting English. After extensive exploration of various open-source tools, we narrowed our focus to Whisper SD and another open-source SD implementation. Upon evaluation, we chose to proceed with the open-source solution due to its lower computational resource requirements, robust performance, and greater flexibility for fine-tuning and customization on our data. This choice ensures better accessibility for future adaptations and improvements.

2.5. Analysis of Existing ASR Solutions

Current customer care centers employ one of two methods:

2.5.1. Manual Process

In the manual review process, a designated individual listens to a subset of recordings from customer care agents and provides feedback to both the agents and the company. This approach has several limitations: it restricts the number of recordings that can be reviewed, which can lead to incomplete and potentially inaccurate insights. Consequently, this may result in suboptimal feedback for both the agents and the company, introduce potential biases, and incur high costs. The manual process, therefore, often fails to offer a comprehensive and unbiased evaluation of agent performance and customer interactions.

2.5.2. Commercial ASR Services

Commercial ASR services are used for call analysis, which are often costly, less accurate for Indian languages, and ineffective in noisy environments. These services typically rely on cloud-based solutions, raising concerns about data privacy and latency.

We evaluated several commercial solutions, and the following summarizes their respective strengths and limitations:

a. Google Speech-to-Text
    i. Strengths: Offers high accuracy and extensive language support, making it a reliable choice for diverse applications.

ii. Limitations: Dependent on internet connectivity and incurs significant costs for large-scale deployments.

b. Microsoft Azure Speech Service
    i. Strengths: Delivers comprehensive ASR capabilities with strong language support and customization features, alongside smooth integration with other Azure services.
    ii. Limitations: Involves costs and data privacy concerns associated with cloud-based solutions.

c. Amazon Alexa
    i. Strengths: Known for its advanced voice recognition and processing capabilities, widely used in consumer devices.
    ii. Limitations: Primarily designed for consumer interactions, with limited applicability in specialized enterprise settings.

2.5.3. Open-source toolkits

We also explored several open-source toolkits for ASR development. The strengths and limitations of these toolkits, based on our extensive evaluation, are summarized as follows:

a. Kaldi
    i. Strengths: Provides extensive customization options and is widely adopted in the research community, making it a powerful tool for ASR research.
    ii. Limitations: Requires substantial expertise for setup and maintenance.

b. Mozilla DeepSpeech
    i. Strengths: Supports offline processing and customization, making it suitable for specific languages and environments.
    ii. Limitations: Depreciated by Mozilla

c. Whisper
    i. Strengths: Demonstrates robust transcription accuracy and language support, excelling in diverse environments.
    ii. Limitations: Complex and resource-intensive for on-premise deployment.

d. NeMo Toolkit
    i. Strengths: Features state-of-the-art ASR models, including conformer architectures, offering high accuracy and performance. Seamlessly

integrate with Nvidia Graphics Processing Unit (GPU) and reach support from Nvidia.

ii. Limitations: Still in development, and requires familiarity with NVIDIA's ecosystem.

e. wav2vec 2.0
   i. Strengths: Effective for low-resource languages, providing robustness and the ability to learn from unlabeled data.
   ii. Limitations: The training process is computationally intensive.

f. SpeechBrain
   i. Strengths: Delivers a flexible and extensible platform for developing custom ASR systems.
   ii. Limitations: Requires considerable effort to customize for specific applications.

g. WeNet
   i. Strengths: Emphasizes real-time ASR with efficient and accurate models suitable for edge deployment.
   ii. Limitations: Performance may vary depending on the language and environmental complexity.

2.6. Research Gap Identification

Despite substantial advancements in ASR technologies, there are notable gaps that impact their effectiveness in practical applications. One significant gap is the need for customizable and cost-effective ASR systems. Many businesses require solutions that can be tailored to specific needs and deployed on-premise or in streaming mode. This would not only mitigate reliance on expensive cloud services but also ensure robust data privacy, a crucial concern for organizations handling sensitive customer information.

Another pressing challenge is handling noise and accents effectively. Current ASR systems often struggle to accurately transcribe speech in noisy environments or from speakers with diverse accents, particularly in low-resource languages. This limitation can undermine the accuracy of customer interaction analysis, as background noise can obscure important details and varied accents can lead to transcription errors. Addressing these gaps is essential for improving ASR systems' reliability and performance.

Enhancements in these areas would enable better integration with LLMs, which can provide advanced contextual understanding and sentiment analysis, further refining the accuracy and utility of customer care insights. Thus, advancing ASR technology to address these issues represents a critical research gap with significant implications for optimizing customer service analysis.

3. Chapter 3: Research Methodology

3.1. Research Design

1. The research will follow a phased approach, beginning with the development of an offline, on-premise, and streaming ASR model utilizing the open-source Nemo toolkit. This initial phase will focus on the design, training, and optimization of ASR models specifically trained for the Hindi language, ensuring robustness against dialectal and accent variations as well as noisy environments.
   Subsequent to the ASR model development, the research will integrate open-source SD techniques to accurately differentiate between multiple speakers in customer care scenarios. This integration aims to enhance the clarity and relevance of transcriptions by accurately attributing spoken content to the correct speakers.
   The final phase of the research will involve the incorporation of LLM based analytics. These LLMs will be integrated into the ASR system to provide advanced call analytics, including sentiment analysis, topic extraction, and insights into customer and agent behavior. This comprehensive analytic capability will offer valuable insights such as the sentiment of the conversation, the status of problem resolution, and the behavior of the agents.
   By following this phased approach, the research ensures a systematic development and integration process, aiming to deliver a robust, efficient, and comprehensive ASR system tailored for customer care centers operating within the Indian linguistic landscape. The deployment of the system on-premise will guarantee data privacy and security, while the use of open-source tools and techniques will ensure cost-effectiveness and scalability for future expansions.

2. The pipeline begins with the capture and pre-processing of the audio signal to reduce noise and normalize volume levels. The first stage involves SD, where the audio is segmented into segments associated with individual speakers. This is done using algorithms that detect speaker change points and cluster segments based on speaker similarity, often leveraging deep learning techniques like x-vectors for accurate speaker representation.
   Once the audio is segmented, each speaker segment is processed through an ASR system. In this stage, the ASR system, often based on advanced models like the Conformer, converts the audio into text. The Conformer model, which combines convolutional neural networks and transformers, excels in capturing both local and global dependencies within the audio, resulting in highly accurate transcriptions. The model directly maps the sequence of input features to a sequence of output characters or words, streamlining the ASR process.

The transcribed text from the ASR system is then processed by a LLM for further NLU tasks. The LLM can enhance the transcriptions by correcting grammatical errors, adding punctuation, and even providing contextual insights or summaries. The integration of LLMs ensures that the output text is not only accurate but also coherent and contextually appropriate.

3.2. Data Collection Methods

Data collection for this study involves sourcing speech samples from various publicly available datasets as well as internal company datasets. Specifically, the primary sources of data include the Mozilla Common Voice dataset, OpenSLR, and Ai4bharat, alongside proprietary datasets maintained internally by the company.
Following are the Data Sources that we used to train our ASR system:

3.2.1. Mozilla Common Voice Dataset

The Mozilla Common Voice project provides a large, diverse set of speech recordings contributed by volunteers from around the world. For this study, we specifically utilize the validated hours of Hindi speech data from this dataset to ensure the quality and reliability of the audio samples. The validated subset includes recordings that have undergone community verification, where volunteers listen to and confirm the accuracy of the recordings. This step is crucial to maintain the integrity and reliability of the ASR system.

3.2.2. Ai4bharat

Ai4bharat provides resources for Indian languages, including speech and text corpora. Their datasets are particularly useful for developing Language Model (LM) and ASR systems tailored to the linguistic diversity found in India. We have incorporated relevant datasets from Ai4bharat to enhance the robustness and accuracy of our ASR system for the Hindi language.

3.2.3. Internal Company Datasets

In addition to publicly available datasets, we utilize proprietary speech data collected internally by the company. These datasets are composed of customer care call recordings, which are instrumental in creating an ASR system that is specifically designed to function in customer service environments. The internal datasets include a variety of dialects and accents, providing a comprehensive training corpus for the ASR model.
The combination of these datasets provides a rich and diverse corpus of speech data, covering various dialects, accents, and recording conditions. By integrating data from

multiple sources, we aim to create a robust ASR system that can effectively handle the linguistic diversity and variability inherent in the Hindi language.
This multi-faceted approach to data collection ensures that the ASR system is well-equipped to deal with the challenges posed by real-world customer care environments, including background noise, speaker variation, and different speech patterns. The following table 1 shows the collected data details.

Dataset	Duration (hours)	Domain
Ai4Bharat Shrutilipi	1,620	FM: Air India Radio
Ai4Bharat IndicSUPERB	150	Generic
Mozilla Common Voice 18.0	15	Generic
Internal Data	1,000	Customer Care
Total	**2,785**	

Table 1: ASR Data Collected from Various Sources

3.3. Data Analysis & Processing Techniques

3.3.1. Data Analysis

In the analysis of both text and audio data, several issues were identified. The audio data exhibited problems such as corruption in some files, inconsistency in formats, varying stereo channels, and differing sample rates.
Similarly, the text data contained various undesired elements, including punctuations, special characters, numbers, English characters, junk characters, and Unicode characters.

3.3.2. Audio Preprocessing

A systematic approach was adopted for audio preprocessing. The audio data was converted to a uniform format of 16,000 Hz, mono channel, 16-bit, .wav files. This format was selected because the uncompressed .wav format allows the model to learn the maximum features from the audio. Additionally, audio segments were truncated to a maximum duration of 30 seconds. This duration was chosen to manage the GPU memory requirements during model training and feature learning effectively.

3.3.3. Text Cleaning & Preprocessing

The text data underwent extensive cleaning and preprocessing. All punctuation marks, special characters (e.g., commas, question marks, exclamation marks), and numbers were removed from the transcripts. Utterances containing numerical digits (e.g., 1, 2, 3) were excluded to ensure the model outputs numbers in word form, as training the model to recognize and output numerical digits in various combinations requires a substantial amount of data, which may not be available.

Further, the preprocessing involved limiting the vocabulary to Hindi characters, thereby removing any utterances containing non-Hindi characters. The training text was normalized from Normalization Form C (NFC) to Normalization Form D (NFD). In NFD, characters are in their decomposed form, which helped reduce the vocabulary size of the ASR system and improved its performance, especially for rare symbols that are combinations of two or more characters. Figure 1 shows an example of NFC and NFD Format.

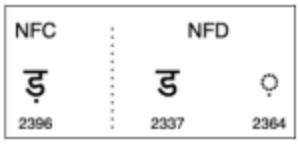


Figure 1: NFC and NFD Format

3.3.4. Data Utilization for Training

Following the preprocessing of both audio and text data, the duration of the processed audio data utilized for training is summarized in Table 1.

Dataset	Duration Before Processing (hrs)	Duration After Processing (hrs)	Data Lost (hrs)
Ai4Bharat Shrutilipi	1,620	1500	120
Ai4Bharat IndicSUPERB	150	140	10
Mozilla Common Voice 18.0	15	14	1
Internal Data	1,000	910	90
Total	**2,785**	**2564**	**221**

Table 2: ASR Data Analysis Before & After Processing

3.4. Experimental Setup

3.4.1. ASR System Training

The ASR system was trained using the NeMo toolkit, developed by NVIDIA for deep learning tasks such as ASR. NeMo provides a comprehensive framework for building, training, and fine-tuning state-of-the-art ASR models. It supports a modular approach, allowing for easy integration and customization of various components, including data preprocessing, model architecture, and training workflows. The toolkit also includes tools for efficient distributed training and inference optimization using the CUDA framework, which is essential for handling large-scale datasets and achieving high performance.

3.4.2. Audio File Processing

For the processing of audio files, we utilized Python packages such as Pydub and Soxi. These tools facilitated efficient handling and manipulation of audio data, which was essential for preparing the dataset for training. The audio preprocessing involved converting the audio data to a uniform format of 16,000 Hz, mono channel, 16-bit, .wav files.

3.4.3. Text Processing

Text processing involves several steps to ensure the integrity and consistency of the text data. Regular expressions (Regex) were employed to filter and remove non-Hindi language characters, punctuation marks, special characters, and numerical digits. This step was crucial to limit the vocabulary to Hindi characters, thus enhancing the model's ability to accurately recognize and transcribe spoken Hindi. Furthermore, the Indic-NLP Library was used to perform NFC to NFD normalization for the Hindi language.

3.5. Evaluation Metrics

We use Word Error Rate (WER) to measure the accuracy of the ASR system. WER is a critical metric for evaluating the performance of ASR systems. It measures the accuracy of a system by comparing its transcribed output to a reference transcript. It is defined as the ratio of the total number of errors (substitutions, insertions, and deletions) to the total number of words in the reference transcript. A lower WER signifies higher accuracy.

The formula for WER is as follows:

$$WER = (S + I + D)/ N$$

Where:

- S = Substitutions (incorrect words)
- I = Insertions (extra words)
- D = Deletions (missing words)
- N = Total words in the reference transcript

4. Chapter 4: System Design, Architecture, and Implementation

4.1. System Requirements

4.1.1. Functional Requirements

The system must meet the following functional requirements:

a. Speaker Diarization

The system must be capable of SD, effectively distinguishing and labeling different speakers during conversations.

b. Accurate Transcription

The system should provide precise and reliable transcription of customer calls, ensuring high fidelity in capturing spoken content in Hindi language or Devnagri script.

c. Transcription Insights

The system should leverage LLMs to generate actionable insights, such as sentiment analysis, problem resolution status, keyword detection, behavioral analysis, and new product features from the transcribed data.

4.1.2. Non-Functional Requirements

The system must also adhere to the following non-functional requirements:

a. On-Premise Deployment

The solution should be deployable on-premise within the customer care center's infrastructure to ensure data privacy and security.

b. High Scalability

The system should be designed to scale efficiently, accommodating varying volumes of data and user demand without degradation in performance.

c. Cost-Effectiveness

The system should be economically viable, providing a cost-effective solution that meets operational needs without compromising on performance or accuracy.

4.2. System Overview

Figure 2 illustrates the complete end-to-end workflow of our system. The process begins with the ingestion of call-recording audio files, which are first passed to the diarization subsystem. This component performs SD, effectively segmenting the audio into distinct speaker segments. In many instances, mono audio recordings include both

the agent's and the customer's voices, making it challenging to attribute spoken content to the correct speaker. SD addresses this challenge by differentiating between speakers.

Following the diarization process, the segmented audio files are forwarded to the ASR system. The ASR system transcribes the audio content into text. The resulting transcriptions are then input into the LLM for further analysis. The LLM processes the text to extract valuable insights and analytics from the recordings. The analytics produced by the LLM are subsequently stored and utilized for further analytical purposes, providing a comprehensive overview of the recorded interactions.

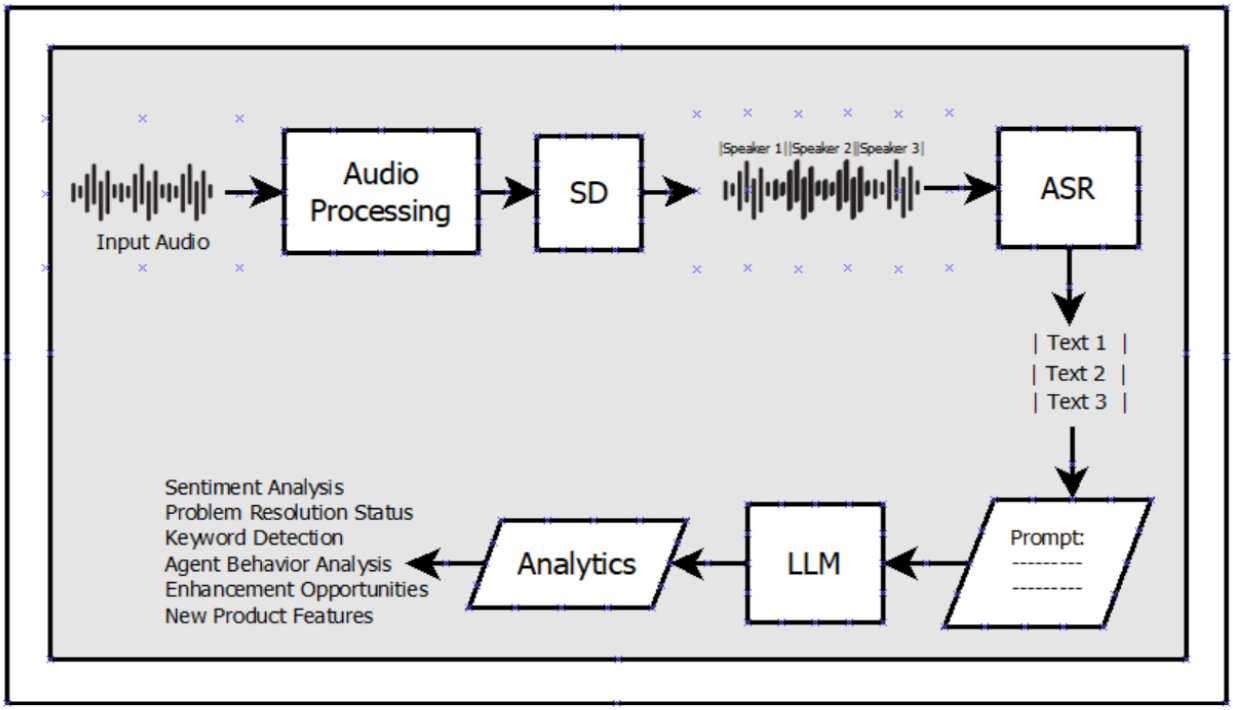


Figure 2: System Overview

4.3. Architecture Design

4.3.1. Architecture design of ASR

An ASR system begins with capturing the raw audio waveform as input. The initial preprocessing step involves splitting the audio into short frames and applying a window function to each frame to reduce spectral leakage. The frames are then passed through the MFCC block, where a Fourier Transform converts them to the frequency domain, followed by the application of a Mel Filter Bank to emphasize frequencies important for human speech. This is followed by a logarithm conversion and Discrete Cosine

Transform (DCT) to obtain the MFCCs, which are compact representations of the speech signal.
These MFCCs are fed into an AM, typically a neural network such as a Transformer-based model. The Acoustic Model (AM) maps the MFCC features to phonetic/ character or Byte Pair Encoding (BPE) units or probabilities. The output from the AM is then processed by the language model, which predicts the probability of word sequences using statistical or neural methods like n-grams or neural networks (e.g., Long Short-Term Memory (LSTM), Transformers). This helps in predicting the most likely words based on the phonetic input.

The decoder combines the outputs from the acoustic and LM to determine the most probable sequence of words. It uses algorithms like the Viterbi algorithm or beam search to optimize this process. Finally, the system produces the output words, providing the transcription of the original speech signal. This integrated approach ensures accurate conversion of speech to text by leveraging the strengths of both acoustic and LM along with efficient decoding techniques.

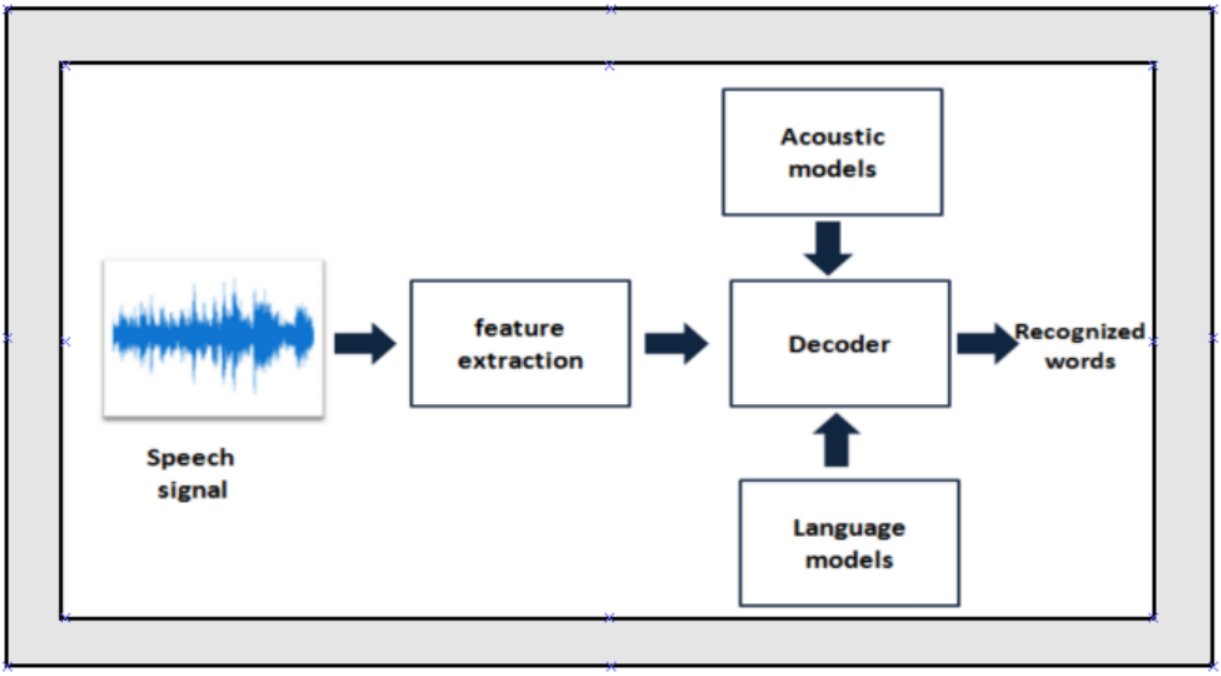


Figure 3: Architecture Diagram of ASR [5]

4.4. Component Design of ASR Module

4.4.1. Feature Extraction

We extracted MFCC features. The window size was set to 0.02 seconds, and the window stride was 0.01 seconds, which means there was a small overlap between windows. We used a Hann window to minimize edge effects. We employed 64 Mel filters to convert the audio signal into the Mel scale, and an Fast Fourier Transform (FFT) size of 512 to determine the number of frequency bins. No frame splicing was applied, and a very small dither value of 0.00001 was added to prevent quantization noise. We did not apply Short-Time Fourier Transform (STFT) convolution in our preprocessing. These settings were chosen to efficiently extract high-quality features for our ASR system.

4.4.2. Tokenization/ Dictionary Creation

We tokenize the text using a unigram SentencePiece tokenizer. This tokenizer operates on the principle of unigrams, meaning it treats each token as an independent unit without considering adjacent tokens. We set the vocabulary size to 128, which means the tokenizer has 128 unique tokens to represent the entire text. This helps in efficiently

handling the text by breaking it down into manageable and meaningful units, which is crucial for the subsequent stages of processing in our system.

4.4.3. Acoustic Model Training

We used Conformer-Connectionist Temporal Classification (CTC) architecture for AM training. Conformer-CTC is a variant of the Conformer model that employs a CTC loss instead of the traditional Recurrent Neural Network Transducer (Rnn-T)/ Transducer loss, making it a non-autoregressive model. The model retains a similar encoder architecture to the original Conformer but replaces the LSTM decoder with a linear decoder atop the encoder.

Key components of the Conformer-CTC architecture include:

1. Encoder: Similar to the original Conformer, the encoder combines self-attention and convolution modules. The self-attention layers are designed to capture global interactions across the input sequence, while the convolution layers focus on learning local correlations, providing a comprehensive representation of the input data.

2. Self-Attention Mechanisms: The self-attention modules in Conformer-CTC support two types of positional encodings:
    a. Absolute Positional Encoding: Traditional self-attention mechanism where position information is added directly to the input embeddings.
    b. Relative Positional Encoding: Inspired by Transformer-XL, this method allows the model to better capture the relative positions of tokens, enhancing its ability to understand contextual relationships within the sequence.

3. Linear Decoder: Conformer-CTC employs a linear decoder. This choice aligns with the CTC loss function, simplifying the model and making it non-autoregressive.

4. Encoding Levels: Conformer-CTC supports both sub-word and character-level encodings.

We trained a large Conformer-CTC model (~121M parameters) using NVIDIA NeMo for ASR. The encoder configuration includes 17 layers with a dimension of the input embedding of 512, 8 attention heads, a sub-sampling factor of 4, and a convolution kernel size of 31. Regularization is managed with a dropout rate of 0.1. We used a batch size of 24 for training with a maximum audio duration of 30 seconds. Shuffling is enabled for the training data. Optimization is performed using the AdamW optimizer with

a learning rate of 0.5 and a weight decay of 1e-3, with the NoamAnnealing scheduler and 10000 warmup steps. Training is conducted on 30 epochs. Models are optimized using a CTC loss.

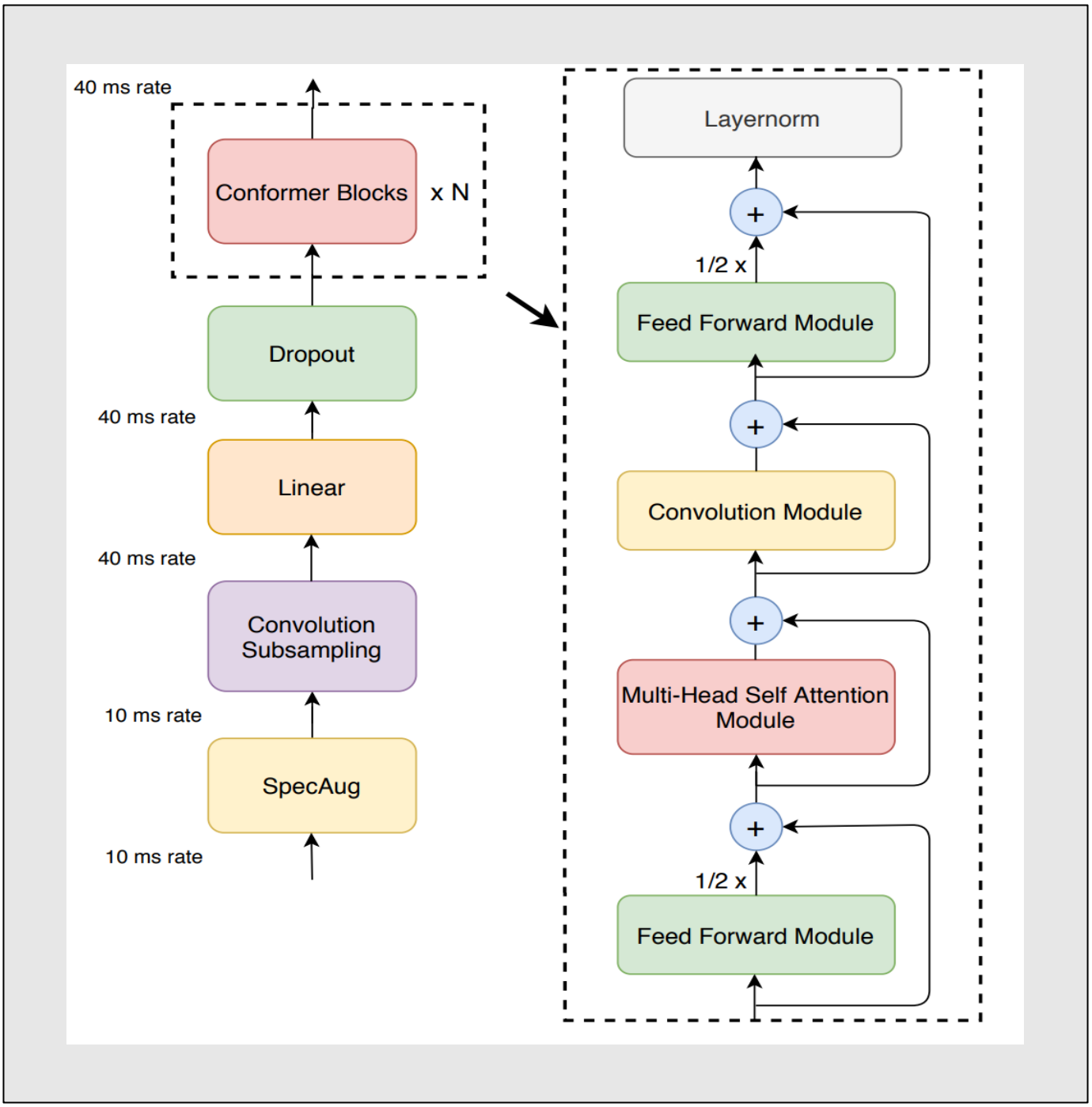


Figure 4: Architecture Diagram of Conformer-CTC [6]

4.4.4. Language Model

The ASR model's output is integrated into a statistical KenLM-based language model, where a beam search algorithm is employed to identify suitable words and correct the spellings of frequently used terms. For our models, we utilize IndicCorp along with our training text data to construct 5-gram KenLM-based LM. During this process, we set a word insertion penalty of -1 and an LM weight of 2, which balances the influence between the language model and the initial ASR output. However, the application of the language model can sometimes lead to an increase in the WER, particularly when the test set text significantly diverges from the training text of the language model. Our experiments also indicate that for developing a domain-specific ASR system, it is crucial to train the language model on text specific to that domain.

4.5. User Interface Design

A web-based dashboard will be employed to present speaker segments, transcriptions, and derived insights. This interface will facilitate efficient interaction with and analysis of call data, providing users with an intuitive platform to explore and interpret the information effectively.

4.6. Challenges Faced & Solutions in Building the ASR

4.6.1. Data Collection and Download

- Challenge: Collecting a diverse and representative dataset of Hindi was challenging due to variability in speech patterns, accents, and recording conditions. Downloading and aggregating large volumes of data also posed logistical and technical issues.
- Solution: A systematic approach was implemented to gather data from multiple public sources. Automated scripts and data management tools facilitated efficient downloading and organization of data.

4.6.2. Audio Processing

- Challenge: The audio data required extensive preprocessing to ensure consistency in format and quality. Issues included varying audio formats, sample rates, and channel configurations.
- Solution: Audio files were standardized to 16,000 Hz, mono channel, 16-bit .wav format. Tools like Pydub and Soxi were used for format conversion and trimming to maintain uniformity across the dataset.

4.6.3. Corrupted Audio

- Challenge: Some audio files were found to be corrupted or incomplete, affecting the overall quality of the training data.

- Solution: Corrupted files were identified through integrity checks and excluded from the training dataset. Robust error-handling mechanisms were implemented to mitigate the impact of corrupted files on the overall system performance.

4.6.4. Text Issues

- Challenge: The text data contained various issues, including extraneous punctuations, special characters, numbers, and English words, which complicated the transcription process.
- Solution: Regular expressions were employed to clean the text data, removing unwanted characters and normalizing the text. Specific scripts were used to handle and convert numbers to word forms to ensure consistency.

4.6.5. GPU Constraints

- Challenge: Training the ASR model required substantial GPU resources, which posed constraints in terms of memory and processing power.
- Solution: Optimization strategies such as batch processing, and distributed training were employed to manage GPU memory usage and enhance processing efficiency.

4.6.6. Accuracy Issues

- Challenge: Achieving high transcription accuracy was challenging due to the variability in accents, speech clarity, and background noise.
- Solution: The model was trained on a diverse dataset to cover various accents and speech conditions. Data & spec augmentation were used to improve accuracy.

4.6.7. Parameter Tuning

- Challenge: Selecting optimal parameters for the ASR model was complex and required extensive experimentation.
- Solution: A systematic approach to hyperparameter tuning was adopted, to identify the most effective parameter settings.

4.6.8. Data Size Issues

- Challenge: Handling large volumes of audio and text data posed challenges in terms of storage and processing.
- Solution: Efficient data management practices were implemented, including data compression and partitioning, to manage large datasets effectively.

4.6.9. Tokenizer Selection

- Challenge: Choosing the appropriate tokenizer for processing Hindi text, including handling matras and special characters, was essential for accurate transcription.
- Solution: After reviewing numerous research papers, we selected BPE as the tokenizer for our system. This decision was based on BPE's proven effectiveness in handling subword units and its ability to improve the accuracy and efficiency of text processing in complex LM.

4.6.10. English and Numerical Data

- Challenge: The presence of English words and numerical data in Hindi text posed additional challenges for the ASR system.
- Solution: English words and numerical data were removed.

By addressing these challenges with targeted solutions, the development of the ASR system was optimized for accuracy, efficiency, and robustness, providing a reliable tool for ASR and analysis in diverse and complex environments.

References

The following are referred journals from the preliminary literature review.

[1]: Scheidt, S., & Chung, Q. B. (2019). Making a case for speech analytics to improve customer service quality: Vision, implementation, and evaluation. Library and Information Sciences, Computer Networks and Communications, Information Systems, 1–12. DOI: https://doi.org/10.1016/j.ijinfomgt.2018.01.002

[2]: Rafaeli, A., Ziklik, L., & Doucet, L. (2008). The Impact of Call Center Employees' Customer Orientation Behaviors on Service Quality. Journal of Service Research, 10(3), 239-255. https://doi.org/10.1177/1094670507306685

[3]: Ahmed, I., Zhou, Y., Sharma, N., Hosier, J. (2024). Text Summarization for Call Center Transcripts. In: Arai, K. (eds) Intelligent Systems and Applications. IntelliSys 2023. Lecture Notes in Networks and Systems, vol 822. Springer, Cham. https://doi.org/10.1007/978-3-031-47721-8_36

[4]: https://hdl.handle.net/10589/218053
[5]: https://www.researchgate.net/figure/ASR-system-architecture_fig7_320466761
[6]:https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html

Appendices

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Work Integrated Learning Programmes Division
II SEMESTER 23-24

DSECLZG628T / **AIMLCZG628T DISSERTATION**

(EC-2 Mid-Semester Progress Evaluation Sheet)

Scheduled Month : *July*

Name of the Student: TANMAY JAIN

ID No. **:** 2022AA05306

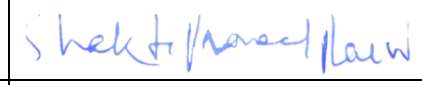
Email Address **:** 2022aa05306@wilp.bits-pilani.ac.in

Name of Supervisor : Shakti P. Rath

Project Title **:** Customer Care Call Analytics Powered by ASR & LLM for Hindi Language

EVALUATION DETAILS

EC No.	Component	Weightage	Comments (Technical Quality, Originality, Approach, Progress, Business value)	Marks Awarded
1	Dissertation Outline	10%	Well-crafted abstract that effectively communicates the project's objectives, methodology, and potential impact.	10%
2.	Mid-Sem Progress Seminar Viva Work Progress	 10% 5% 15%	The ASR system for Hindi has been successfully implemented with a conformer-based architecture, showing promising initial results. Integration with Speaker Diarization and LLM-based analytics is in progress.Results from these implementations should be included in the report to demonstrate effectiveness and progress.	 10% 4% 15%

	Organizational Mentor	
Name	Shakti P. Rath	
Qualification	PhD	
Designation & Address	Principal Architect & Bangalore	
Email Address	shaktiprasadrath9@gmail.com	
Signature		
Date	27 July, 2024	