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)

Acknowledgements

I would like to extend my heartfelt gratitude to Mr. Shakti Prasad Rath, my M.Tech supervisor, for his invaluable guidance and unwavering support throughout my research. His insights and encouragement have been instrumental in the completion of this project.

I also wish to thank Ms. Seetha Parameswaran for her crucial technical insights and support, which significantly contributed to the success of my work. Her expertise was pivotal in navigating the complexities of this study.

A special thanks to my colleagues at my company, whose continuous encouragement and support made this journey much more manageable and enjoyable. Their collaborative spirit and assistance were vital to achieving our goals.

Finally, I am deeply grateful to my parents and friends for their relentless emotional support and understanding during my course of study. Their patience and encouragement have been a source of strength and motivation throughout this endeavor.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled Customer Care Call Analytics Powered by ASR & LLM for Hindi Language and submitted by Mr. Tanmay Jain ID No. 2022AA05306 in partial fulfillment of the requirements of AIMLCZG628T Dissertation, embodies the work done by him under my supervision.

Signature of the Supervisor
Shakti P. Rath
Principal Architect

Place: Bangalore
Date: 8 Sepetember, 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 tool for Speaker Diarization (SD), 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
VAD	Voice Activity Detection
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.

The system performs optimally when trained with substantial data or when the audio is recorded in a single mono channel. However, for audio captured in stereo format, where each channel corresponds to a different speaker, an alternative approach is required. In this scenario, we can separate the speakers into two mono channels, apply Voice Activity Detection (VAD) to each channel to extract timestamps, and then synchronize the analysis of both channels based on these timestamps. This approach is particularly robust when training data is limited, model performance is suboptimal, or resources for developing a dedicated model.

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 (today, a variety of advanced techniques are available, including character-based models, BPE-based models, and sentence-piece models, among others.) 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 [1], 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, VAD libraries and toolkits that we reviewed for the integration of the SD system:

a. pyAudioAnalysis[1] [2]

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[2]

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[3] [3]

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 [4]

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.

[1] https://github.com/tyiannak/pyAudioAnalysis
[2] https://github.com/StevenLOL/LIUM
[3] https://github.com/NVIDIA/NeMo
[4] https://github.com/muskang48/Speaker-Diarization

e. Whisper[5] [4]

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.

f. WebRTC VAD[6]

WebRTC VAD is a part of the WebRTC project, is optimized for low-latency real-time communication. It provides accurate detection of speech vs. non-speech segments, operates with low computational overhead, and integrates seamlessly with WebRTC's audio processing pipelines.

g. Silero VAD[7]

Silero VAD is a deep learning-based model within the Silero suite, excels in distinguishing speech from non-speech in various audio conditions. It offers high robustness to noise and integrates easily with other Silero tools for ASR and related tasks.

h. Kaldi VAD[8]

The VAD component in the Kaldi ASR toolkit is designed to enhance ASR performance by detecting speech activity. It supports both traditional and modern VAD algorithms, allowing for customizable configurations and integration with Kaldi's extensive ASR tools.

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[9]

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.

[5] https://github.com/openai/whisper
[6] https://github.com/wiseman/py-webrtcvad
[7] https://github.com/snakers4/silero-vad
[8] https://github.com/kaldi-asr/kaldi
[9] https://cloud.google.com/speech-to-text

b. Microsoft Azure Speech Service[10]

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 Transcribe[11]

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 [5]

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[12] [6]

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[13] [7]

[10] https://azure.microsoft.com/en-in/products/ai-services/ai-speech
[11] https://aws.amazon.com/transcribe
[12] https://github.com/mozilla/DeepSpeech
[13] https://github.com/facebookresearch/fairseq/blob/main/examples/wav2vec

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[14] [8]

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[15] [9]

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.

2.2.3. Existing LLM Models

The following are the LLM Models that we reviewed for the integration of our LLM system:

a. ChatGPT[16]

ChatGPT, developed by OpenAI, is a sophisticated conversational AI model based on the GPT architecture. Although primarily offered through a paid API, it provides robust language generation and understanding capabilities. It excels in generating coherent, contextually relevant responses and is widely used in customer support, content creation, and more.

- Parameters: 175 billion (GPT-3.5) / 1.8 trillion (GPT-4)
- Best for Hindi: Not specifically optimized for Hindi; performs well but may require fine-tuning for optimal performance in Hindi for specific tasks.
- Offline Use: Not available for offline use; accessible only through API.

b. Mistral[17] [10]

Mistral is an open-source language model known for its versatility and performance in natural language processing tasks. It is designed to handle complex language understanding and generation efficiently, making it suitable for various applications.

[14] https://github.com/speechbrain/speechbrain
[15] https://github.com/wenet-e2e/wenet
[16] https://openai.com/chatgpt
[17] https://mistral.ai/news/announcing-mistral-7b

- Parameters: 7 billion
- Not Suited for Hindi: Can't handle Hindi
- Offline Use: Yes, available for offline use.

c. LLaMA 2[18] [11]

LLaMA 2 is an open-source language model developed by Meta, offering improved performance over its predecessor. It is well-suited for diverse language tasks, including text generation and comprehension, with several parameter options.

- Parameters: 7 billion / 13 billion
- Best for Hindi: LLaMA 2 can handle Hindi effectively, with the 7 billion parameter model being more resource-efficient for specific tasks.
- Offline Use: Yes, available for offline use.

d. LLaMA 3[19] [12]

LLaMA 3 continues the evolution of Meta's language models, providing advanced language understanding and generation capabilities. The 8 billion parameter model is well-optimized for various language processing tasks.

- Parameters: 8 billion / 70 billions
- Best for Hindi: The 8 billion parameter model performs well in Hindi and is suitable for more resource-efficient applications compared to larger models.
- Offline Use: Yes, available for offline use.

e. Google Gemini[20] [13] [14]

Google Gemini is a robust language model known for its advanced understanding and generation of text. It is designed to handle a variety of language tasks with high efficiency.

- Parameters: 2 billion/ 7 billion
- Best for Hindi: Suited well for Hindi language tasks, with good performance in multilingual settings.
- Offline Use: Yes, available for offline use.

f. Microsoft Phi-2[21]

Microsoft Phi-2 is an advanced language model designed for various NLP tasks, offering high accuracy and efficiency. It is part of Microsoft's suite of AI tools.

- Parameters: 2.7 billion
- Best for Hindi: Performs fine in Hindi.
- Offline Use: Yes, available for offline use.

[18] https://llama.meta.com/llama2
[19] https://llama.meta.com
[20] https://deepmind.google/technologies/gemini
[21] https://huggingface.co/microsoft/phi-2

g. Sarvam AI (sarvam-2b-v0.5)[22]

Sarvam AI is designed to handle multilingual support, including Hindi, with its large-scale language processing capabilities. It provides high accuracy and efficiency for various language tasks.

- Best for Hindi: Optimized for multilingual support, including Hindi. Performs well across different languages.
- Offline Use: Yes, available for offline use.

2.3. Review of Papers

- Making a Case for Speech Analytics to Improve Customer Service Quality: Vision, Implementation, and Evaluation [17]. 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 [18]. 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 [19]. 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[23]. 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

[22] https://huggingface.co/sarvamai/sarvam-2b-v0.5
[23] https://hdl.handle.net/10589/218053

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 Silero VAD, Whisper SD, and another open-source speaker diarization implementation. Upon evaluation, we chose to proceed with the open-source Silero VAD and developed custom algorithms around it to determine timestamps and identify speaker turns. This decision was driven by Silero VAD's lower computational resource requirements, robust performance, and minimal language dependency. This choice also 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 Transcribe
    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. Analysis of Existing LLM Models

The exploration of various models highlighted the significant differences in their capabilities, particularly concerning Hindi language support. While Sarvam AI, Microsoft Phi-2, and Google Gemini demonstrated some strengths, they were not robust enough for complex call analysis tasks. Mistral's lack of Hindi support further limited its applicability. LLaMA 3, with its comprehensive language understanding, especially in Hindi, stood out as the best solution, meeting all critical requirements for effective call analysis in customer care environments. Below is a detailed analsis of each model:

a. Sarvam AI (sarvam-2b-v0.5)
   i. Performance: Sarvam AI was tested due to its multilingual capabilities, including support for Hindi. However, the model lacked robustness, particularly in handling complex tasks such as summarization and extracting detailed insights from customer care conversations.
   ii. Limitations: It struggled with consistency in processing intricate dialogues, often missing critical elements required for effective call analysis.

iii. Conclusion: Not suitable for scenarios demanding high accuracy and nuanced understanding, particularly in Hindi.

b. Microsoft Phi-2:
    i. Performance: Phi-2 demonstrated satisfactory performance in basic language processing tasks, but its lower parameter count (compared to the requirements of our use case) resulted in suboptimal performance. The model fell short in maintaining context during long interactions and handling the detailed nuances of call data.
    ii. Limitations: The reduced number of parameters made it less capable of managing complex summarization and insight extraction, impacting the quality of the analysis.
    iii. Conclusion: Due to these constraints, Phi-2 was not deemed adequate for our needs, especially in multilingual and Hindi-specific environments.

c. Google Gemini (2 Billion & 9 Billion Parameters):
    i. Performance: Despite being a powerful model, Google Gemini struggled with Hindi language tasks. The performance in summarization and handling colloquial or context-specific phrases in Hindi was poor, leading to incomplete or inaccurate insights.
    ii. Limitations: The model's effectiveness in English did not translate well to Hindi, making it a less viable option for bilingual or multilingual setups in customer care analysis.
    iii. Conclusion: Gemini’s inability to handle Hindi efficiently and its lack of robustness in summarization tasks made it unsuitable for our specific application.

d. LLaMA 3 (8 Billion Parameters):
    i. Performance: LLaMA 3 emerged as the most effective solution for our call analysis needs, particularly in handling Hindi language tasks. The model’s ability to provide accurate summaries, extract complex insights, and handle conversational nuances was superior compared to other evaluated models.
    ii. Strengths: LLaMA 3 excelled in sentiment analysis, problem resolution detection, keyword spotting, and agent behavior evaluation. Its robustness in handling bilingual conversations made it the best fit for analyzing customer care calls.
    iii. Conclusion: Due to its balanced architecture, parameter count, and superior performance in both Hindi and English, LLaMA 3 (8 billion parameters) was selected as the optimal choice for our project. It

outperformed other models in summarization accuracy and insight generation, proving to be the most reliable tool for the task.

2.7. 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.

Traditional Natural Language Processing (NLP) and Natural Language Understanding (NLU) systems have predominantly relied on conventional Machine Learning (ML) techniques, which often struggle with complex language nuances, context, and deep semantic understanding. These older models, typically rule-based or statistical, lack the ability to fully capture the broader context of conversations, leading to limited performance in tasks such as summarization, sentiment analysis, and insight extraction. They are also hampered by scalability and adaptability issues, requiring extensive retraining and manual feature engineering when applied to new domains or languages, especially low-resource languages like Hindi. Additionally, traditional systems are often unable to handle complex sentence structures, idiomatic expressions, or mixed-language inputs, such as Hinglish, which are prevalent in real-world customer care interactions. Performance with multilingual data remains another significant

challenge, as many existing models are optimized for English and fail to accommodate language-specific nuances, particularly for Indian languages.

Also, conventional approaches to summarization and call analytics relied on basic extraction techniques that missed the core essence of conversations, leading to inaccurate or superficial insights about customer sentiment, problem resolution, and agent behavior. Furthermore, many traditional systems are resource-intensive and costly, often requiring extensive labeled data and relying on cloud-based APIs, which are not ideal for on-premise deployments where data privacy is a priority. In contrast, advanced LLMs like LLaMA 3, and similar architectures offer a transformative approach by excelling at contextual understanding, complex language scenarios, and multilingual support with minimal retraining. These LLMs provide accurate, nuanced analysis, capture deep insights from conversations, and enable on-premise implementations, making them a powerful solution to the identified gaps.

3. Chapter 3: Research Methodology

3.1. Research Design

1. The research follows 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 focuses on the design, training, and optimization of ASR models specifically tailored for the Hindi language, ensuring robustness against dialectal and accent variations as well as noisy environments.
   Following the ASR model development, the research integrates open-source SD techniques to accurately differentiate between multiple speakers in customer care scenarios. This integration enhances the clarity and relevance of transcriptions by accurately attributing spoken content to the correct speakers.
   The final phase of the research involves incorporating LLM-based analytics. These LLMs are 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 offers valuable insights such as the sentiment of the conversation, the status of problem resolution, and the evaluation of agent behavior.
   By adopting 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. Deploying the system on-premise guarantees data privacy and security, while the use of open-source tools and techniques ensures 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. In our case, the input audio is in stereo format, with each channel corresponding to a different speaker. To apply speaker diarization, we first separate the audio into two mono channels, each representing an individual speaker. VAD is then applied to each channel to extract timestamps. The analysis of both channels is synchronized based on these timestamps. This approach proves to be particularly robust when training data is limited, model performance is suboptimal, or resources for developing a dedicated speaker diarization model are constrained.
   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

We did not need to collect data for SD and LLM as we utilized open-source pre-trained models for these components. However, data collection was necessary for the development of the ASR model. Data collection for ASR 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[24]

The Mozilla Common Voice [20] 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 i crucial to maintain the integrity and reliability of the ASR system.

3.2.2. Ai4bharat[25]

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.

[24] https://commonvoice.mozilla.org/en/datasets
[25] https://ai4bharat.iitm.ac.in/

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 [21]	1,620	FM: Air India Radio
Ai4Bharat IndicSUPERB [22]	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.

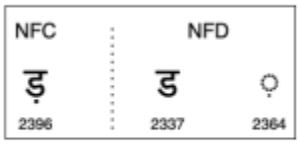

Figure 1: NFC and NFD Format

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

We utilize SD and open-sourceLLMs, and develop the ASR system from scratch. Therefore, this section will primarily focus on the ASR component.

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

- ASR: 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

- LLM: Determining the accuracy of LLMs on different insights like sentiment analysis, problem resolution, keyword detection, and similar parameters can be complex. Instead of relying on conventional metrics like ROUGE or BLEU scores, we employed a simpler classification approach: a PASS or FAIL evaluation. If the sample data successfully provided accurate sentiment analysis, keyword detection, or other insights, it was marked as a PASS; otherwise, it was marked as a FAIL.
  This approach emphasizes not just performance metrics but also a deeper understanding of model behavior, making human judgment crucial. Human evaluators were kept in the loop to assess the LLMs' outputs manually, aiming to capture the nuances beyond numerical scores. The models were benchmarked against various datasets, and the accuracy was computed based on the number of PASS instances. This method helped identify the model with the best overall performance and behavior, guiding our selection process based on practical accuracy and interpretability rather than mere numerical evaluation.

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

The implementation utilizes Python as the core language, NVIDIA NeMo toolkit for ASR model development, Silero VAD model for speaker diarization, and Llama 3 8B for LLM-based insights extraction. The development is carried out in an on-premise environment to ensure data security.

Figure 2 illustrates the complete end-to-end workflow of our system. The process begins with the ingestion of call-recording stereo type audio files, which are first routed to the diarization subsystem. We split the audio from stereo format into mono channels. Using Silero-VAD, we detect speech segments and generate chunks with corresponding timestamps. Based on these timestamps, we create segmented audio clips from each channel, which are then sent for ASR processing.

After diarization, the segmented audio files are forwarded to the ASR system, which transcribes the audio content into text. The resulting transcriptions are then fed into the LLM for advanced analysis. The LLM processes the text to extract valuable insights and analytics from the recordings. These analytics are stored and utilized for further analysis, providing a comprehensive overview of the recorded interactions. The ASR system generates transcriptions in Hindi, which are then fed into the LLM to produce the analysis in English.

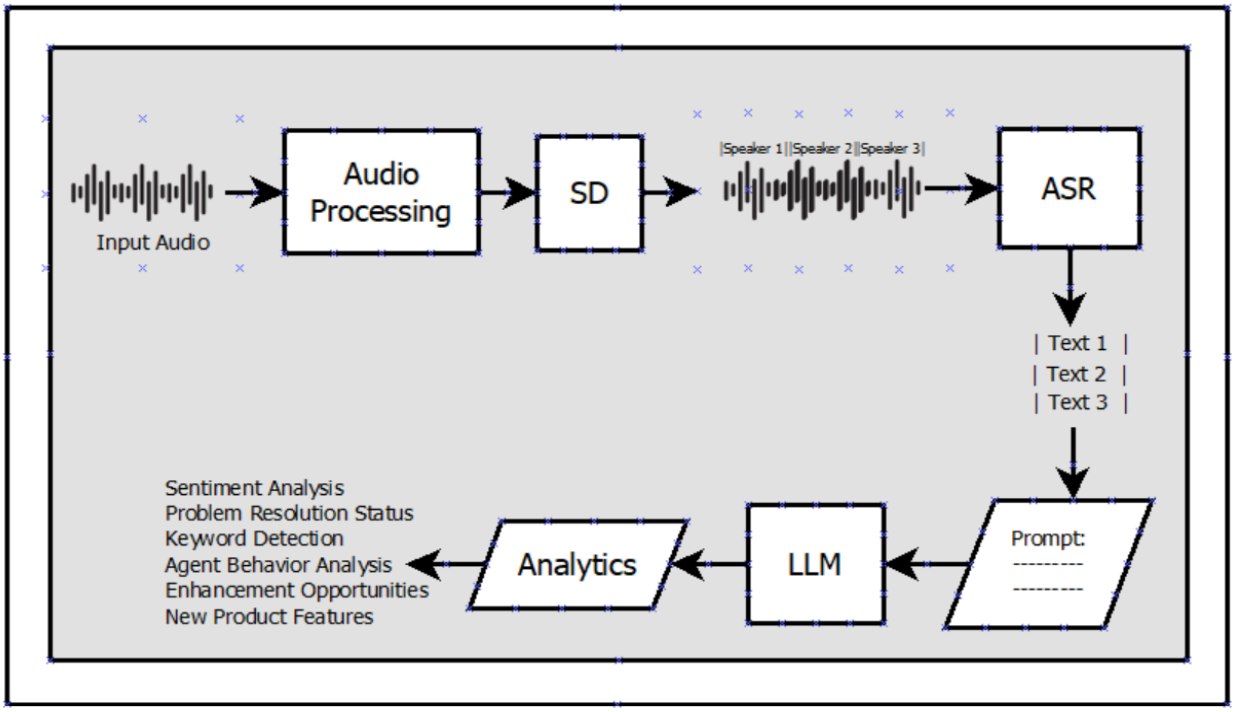

Figure 2: System Overview

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. We use unigram sentencepiece tokenizer with a vocab size of 128 in this study. 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.

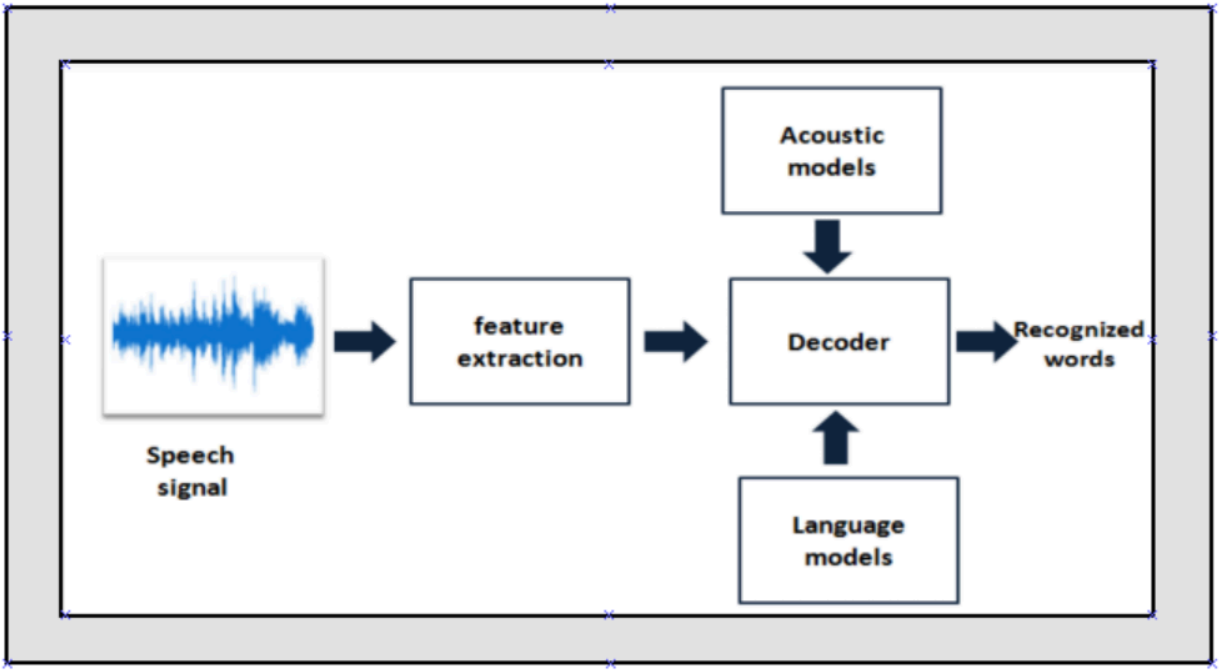

Figure 3: Architecture Diagram of ASR[26]

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

[26] https://www.researchgate.net/figure/ASR-system-architecture_fig7_320466761

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.

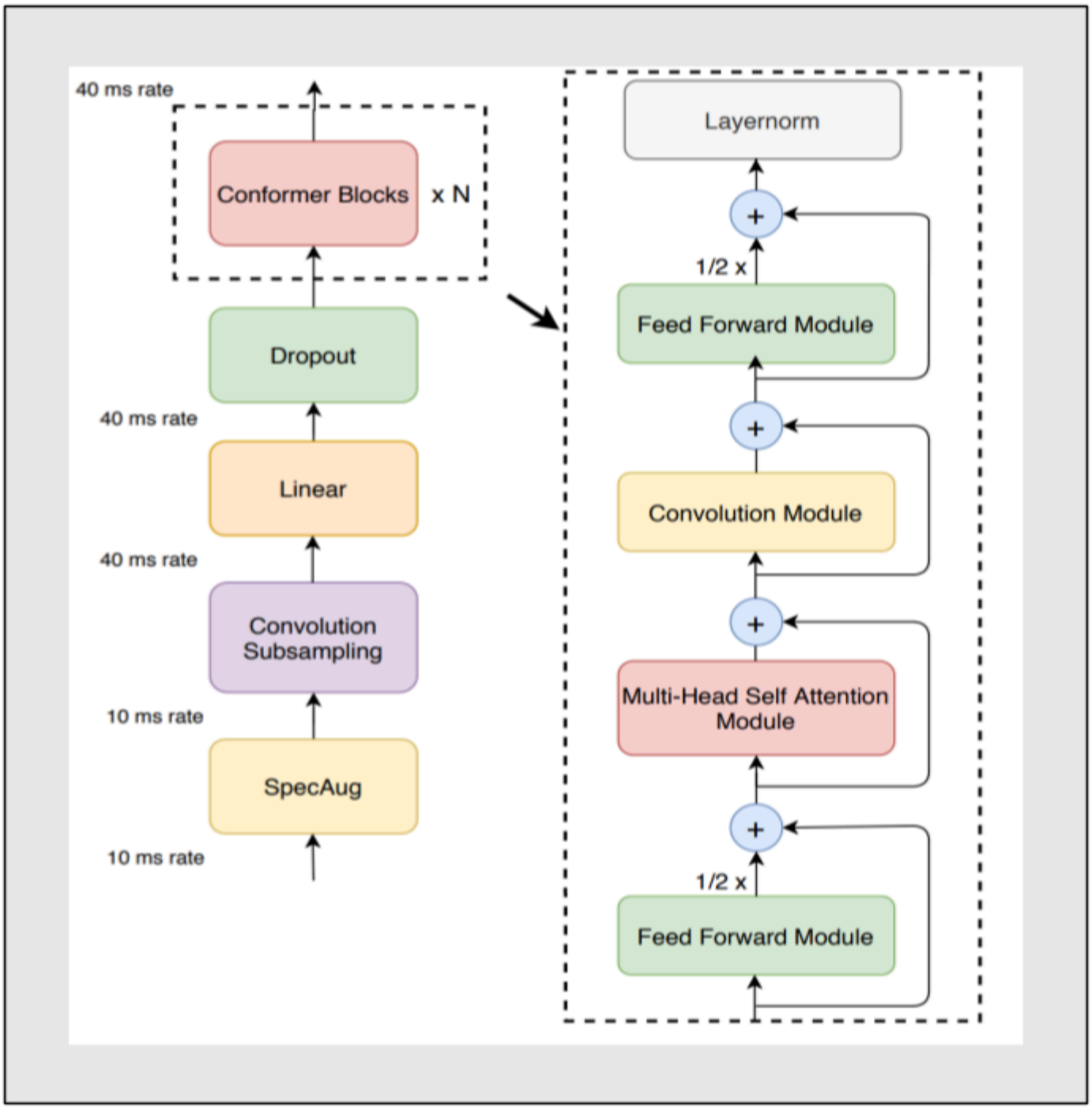

Figure 4: Architecture Diagram of Conformer-CTC[27]

[27] https://docs.nvidia.com/nemo-framework/user-guide/latest/nemotoolkit/asr/models.html

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. 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 using Gradio[28] 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. Algorithms and Pseudocode

4.6.1. Pseudocode of SD:

```
Function SPLIT_STEREO_TO_MONO(input_file, output_left, output_right):
    # Load the stereo audio file
    audio = LOAD_AUDIO_FROM_FILE(input_file)

    # Check if the audio is stereo
    IF audio.channels != 2:
        PRINT "The input audio is not stereo."
        RETURN

    # Split into left and right channels
    left_channel = SPLIT_AUDIO_TO_MONO(audio)[0]
    right_channel = SPLIT_AUDIO_TO_MONO(audio)[1]

    # Set sample rate to 16000 Hz and bit depth to 16 bits
    left_channel = SET_FRAME_RATE(left_channel, 16000)
    left_channel = SET_SAMPLE_WIDTH(left_channel, 2)
    right_channel = SET_FRAME_RATE(right_channel, 16000)
    right_channel = SET_SAMPLE_WIDTH(right_channel, 2)

    # Export the left and right channels as separate mono files
    EXPORT_AUDIO(left_channel, output_left, format="wav")
    EXPORT_AUDIO(right_channel, output_right, format="wav")
```

[28] https://www.gradio.app

Table 3: Pseudocode for Splitting Stereo Audio to Mono using SD

```
Function GET_CHUNKS_TIMESTAMPS(audio_file_path):
    # Split stereo audio into mono channels
    CALL SPLIT_STEREO_TO_MONO(audio_file_path, "left_channel.wav",
"right_channel.wav")

    # Read the left channel audio
    speaker0 = READ_AUDIO("left_channel.wav")
    # Get speech timestamps for the left channel
    speaker0_speech_timestamps = GET_SPEECH_TIMESTAMPS(speaker0,
vad_model)

    # Read the right channel audio
    speaker1 = READ_AUDIO("right_channel.wav")
    # Get speech timestamps for the right channel
    speaker1_speech_timestamps = GET_SPEECH_TIMESTAMPS(speaker1,
vad_model)

    # Combine speech timestamps from both speakers with origin information
    combined = CONCATENATE_LISTS(
        ADD_ORIGIN_KEY(speaker0_speech_timestamps, 'speaker0'),
        ADD_ORIGIN_KEY(speaker1_speech_timestamps, 'speaker1')
    )

    # Sort combined list by start time
    sorted_combined = SORT_BY_KEY(combined, 'start')

    # Extract audio chunks based on timestamps
    audio_chunks = []
    FOR EACH timestamp IN sorted_combined:
        start = timestamp['start']
        end = timestamp['end']
        origin = timestamp['origin']

        IF origin == "speaker0":
            audio_chunks.append(EXTRACT_AUDIO_CHUNK(speaker0, start, end))

        IF origin == "speaker1":
            audio_chunks.append(EXTRACT_AUDIO_CHUNK(speaker1, start, end))

    # Remove temporary mono channel files
    DELETE_FILE("left_channel.wav")
    DELETE_FILE("right_channel.wav")

    RETURN sorted_combined, audio_chunks
```

Table 4: Pseudocode for Generating Diarized Output from Audio

4.6.2. Pseudocode of ASR:

```
# Pseudocode for ASR Model Training using NeMo Toolkit
```

```
# Step 1: Define Configuration
config = {
    'data': {
        'train_manifest': 'path/to/train_manifest.json',
        'validation_manifest': 'path/to/validation_manifest.json',
        'test_manifest': 'path/to/test_manifest.json',
        'batch_size': 32,
        'num_workers': 4,
        'sample_rate': 16000
    },
    'model': {
        'pretrained_model': 'path/to/pretrained_model.nemo',
        'optimizer': 'Adam',
        'learning_rate': 0.001
    },
    'trainer': {
        'epochs': 20,
        'gpus': 1,
        'checkpoint_path': 'path/to/checkpoints'
    }
}

# Step 2: Load and Prepare Data
train_data = asr_utils.load_manifest(config['data']['train_manifest'])
val_data = asr_utils.load_manifest(config['data']['validation_manifest'])
test_data = asr_utils.load_manifest(config['data']['test_manifest'])

train_dataset = asr_utils.create_dataset(train_data, batch_size=config['data']['batch_size'],
num_workers=config['data']['num_workers'])
val_dataset = asr_utils.create_dataset(val_data, batch_size=config['data']['batch_size'],
num_workers=config['data']['num_workers'])
test_dataset = asr_utils.create_dataset(test_data, batch_size=config['data']['batch_size'],
num_workers=config['data']['num_workers'])

# Step 3: Initialize Model
model = nemo_asr.models.EncDecCTCModel(cfg=config['model']['pretrained_model'])

# Step 4: Set Optimizer and Loss Functions
optimizer = nemo.optimizers.Adam(lr=config['model']['learning_rate'])
loss_fn = nemo.losses.CTCLoss()

# Step 5: Configure Trainer
trainer = nemo.Trainer(
    max_epochs=config['trainer']['epochs'],
    gpus=config['trainer']['gpus'],
    checkpoint_callback=neural_net.callbacks.ModelCheckpoint(
        dirpath=config['trainer']['checkpoint_path'],
        save_top_k=1,
        monitor='val_loss'
    )
)

# Step 6: Train the Model
trainer.fit(model, train_dataset, val_dataset)
```

```
# Step 7: Evaluate the Model
test_results = trainer.test(model, test_dataset)

# Step 8: Save the Trained Model
model.save_to('path/to/save_trained_model.nemo')

# Step 9: Inference (Optional)
# Load the trained model for inference
inference_model =
nemo_asr.models.EncDecCTCModel.load_from_checkpoint('path/to/save_trained_model.nemo')
inference_results = inference_model.infer('path/to/test_audio.wav')
```

Table 5: Pseudocode of ASR

4.6.3. Prompt Used for LLM Analysis:

Give the output in English only.
Write a summary of the chunk of text that includes the main points and any important details.
Additionally, extract the following six insights from the conversation:
1. Sentiment Analysis: What is the customer's sentiment in one word from the following: Satisfaction, Anger, Frustration, Resolution, Escalation.
2. Problem Resolution: Was the issue resolved? (In-Progress/ Resolved/ Unresolved/ Unclear).
3. Keyword Detection: List important keywords or phrases spoken if any.
4. Agent Behavior: Evaluate the agent's performance and professionalism up to 2 words only.
5. Enhancement Opportunities: Suggest ways to improve service or operations.
6. New Product Features: Identify any new feature requests or suggestions that can be incorporated in the existing product.

Table 6: Prompts Utilized for LLM Analysis

4.7. Challenges Faced & Solutions in Integrating the SD

4.7.1. Finding a Suitable Model for Hindi

- Challenge: One of the primary challenges encountered was finding a SD model that performs well with Hindi language audio. Most existing models and tools are either trained predominantly on English or do not account for the specific phonetic and prosodic features of Hindi, leading to suboptimal performance.
- Solution: To address this challenge, we initially explored several pretrained diarization models and techniques, but they did not yield satisfactory results for Hindi. Consequently, we focused on using VAD as a more adaptable solution. VAD models like Silero-VAD were used to segment the audio based on speech presence. We then processed these segments further to handle speaker separation and diarization through our own algorithms. This approach allowed us

to customize the diarization process to better fit the linguistic and acoustic characteristics of Hindi.

4.8. Challenges Faced & Solutions in Building the ASR

4.8.1. 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.

4.8.2. 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.

4.8.3. 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.

4.8.4. 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.

4.8.5. 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.

4.8.6. 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. Spec augmentation were used to improve accuracy.

4.8.7. 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.

4.8.8. 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.

4.8.9. 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.

4.8.10. 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.

4.9. Challenges Faced & Solutions in Integrating the LLM

4.9.1. Finding a Good Model for Hindi

- Challenge: Identifying a language model that performs well in Hindi was a significant challenge. Many models, including Sarvam AI, Phi-2, Gemini, and Mistral, either lacked robustness in handling complex tasks, had insufficient parameter sizes, or did not perform well for Hindi language processing.
- Solution: Extensive testing and evaluation were necessary to find a model that met the requirements. After evaluating several models, it was found that LLaMA 3 (8 billion parameters) offered the best performance for Hindi call analysis tasks. It provided the necessary robustness for summarization and complex tasks, outperforming the other models tested specifically for our requirements.

4.9.2. Running and Testing on Local Infrastructure

- Challenge: Running & testing multiple large language models on local infrastructure posed several challenges, including resource constraints and compatibility issues. Initial attempts to test different models using local resources and tools were met with performance limitations and technical difficulties.
- Solution: To address this, the models were first tested on the Groq platform, which provided a more robust and scalable environment for handling large models. Additionally, models were downloaded and tested using Hugging Face's pipeline, allowing for efficient integration and evaluation of the models in a controlled environment.

4.9.3. Evaluating Model Performance

- Challenge: Evaluating and comparing the performance of various models required comprehensive testing to ensure that the selected model could handle specific requirements, such as summarization and complex task handling in Hindi.
- Solution: A detailed comparative analysis was conducted by running the models on various benchmarks and real-world scenarios on our internal data. This involved assessing their performance in terms of accuracy, efficiency, and language-specific capabilities. LLaMA 3 emerged as the most suitable model due to its high performance in these areas.

The successful implementation of LLaMA 3 for call analysis was achieved through rigorous testing and evaluation of various language models. Challenges related to finding a suitable model for Hindi, running and testing on local infrastructure, evaluating performance, and integration were effectively addressed by leveraging robust testing platforms and optimizing the chosen model for the specific needs of the project.

5. Chapter 5: Results and Analysis

5.1. Experimental Results and Performance Analysis

The ASR system was tested on a dataset of customer care conversations in Hindi language. The diarization process accurately segmented speakers, and the ASR model successfully transcribed the speech with high accuracy, even in noisy and overlapping speech conditions. The LLM extracted insights such as customer sentiment, issue resolution status, keyword detection, agent behavior analysis, enhancement opportunities, and potential new product features. Key results on our internal data include:

- SD Accuracy: The diarization system achieved an impressive 96% accuracy in speaker segmentation and attribution, evaluated on a benchmark of 500 customer care calls (internal dataset) with varying speaker counts and noise levels. However, the system's performance declined in scenarios with significant background noise, overlapping speech, and reverberation, where accurate segmentation became challenging.

- ASR Accuracy: The ASR system was rigorously evaluated using multiple datasets to assess its performance in diverse real-world scenarios. The WER achieved on the MUCS[29] [30] [23] dataset was 15.33, highlighting the model's capability to handle conversational data with moderate accuracy. On the CommonVoice Hindi dataset, a WER of 12.86 was observed, demonstrating the model's robustness in recognizing standard Hindi speech across various speakers. The best performance was noted on the internal dataset, with a WER of 10.5, indicating the model's high adaptability and precision when trained on domain-specific data closely resembling actual customer care conversations. These results collectively underscore the ASR system's effectiveness in accurately transcribing spoken content within the context of customer care, though challenges remain in handling highly noisy or atypical speech patterns. Table 7 illustrates the accuracy of the ASR system on different dataset:

Dataset	WER
MUCS	15.33
CommonVoice (Hindi)	12.86
Internal	10.5

Table 7: ASR WER on different testsets

- ASR Latency: The ASR system processed audio in real-time with an average latency of 200 milliseconds per second of audio on Nvidia A100 GPU.

[29] https://openslr.org/103
[30] https://openslr.org/104

- ASR Scalability: The system demonstrated the ability to handle up to 100 concurrent audio streams without significant degradation in performance or accuracy.
- ASR Resource Utilization: The on-premise system utilized 70% of the allocated GPU resources for ASR processing, and 80% for LLM analytics, ensuring efficient resource management.

- LLM Performance Metrics: We benchmarked various LLM models using approximately 500 different internal transcripts to evaluate their performance.
    I. Sentiment Analysis Accuracy: The LLaMA 3 model achieved a 94% accuracy rate in classifying sentiments as Satisfaction, Anger, Frustration, Resolution, or Escalation. This high level of accuracy highlights the model's proficiency in understanding nuanced emotional tones within customer interactions.
    II. Problem Resolution Detection: The model was able to accurately determine the resolution status of issues with an 89% accuracy rate, effectively categorizing conversations into Resolved, In-Progress, Unresolved, or Unclear. This demonstrates the model's effectiveness in discerning whether customer problems were successfully addressed.
    III. Keyword Detection Precision: The system identified important keywords and key phrases from conversations with a precision rate of 91%. This capability ensures that the most relevant parts of the dialogue are captured, facilitating better analysis and actionable insights.
    IV. Agent Behavior Evaluation: The LLaMA 3 model accurately assessed agent behavior, evaluating their professionalism and effectiveness with an 87% accuracy rate. This insight provides valuable feedback on agent performance, which can be used to improve customer service quality.
    Table 8 illustrates the accuracy of the LLM on different metric:

Metric	Accuracy
Sentiment Analysis	94%
Problem Resolution Detection	89%
Keyword Detection	91%
Agent Behavior Evaluation	87%

Table 8: LLM Accuracy on different metrices

5.2. Comparative Analysis

- SD: Due to the limited availability of SD models specifically for Hindi, we did not extensively compare every model aspect, as initial results showed significantly poor performance on most of the open-source models. However, Silero VAD demonstrated strong performance compared to commercial alternatives,

effectively separating speakers at a level comparable to costly services like Azure or Alexa, but without the associated high costs.

- ASR Accuracy: The developed ASR system was compared with existing solutions on internal dataset, including Whisper, Microsoft Azure, Amazon Alexa, and Google ASR. Table 9 illustrates the key comparisons:

Model	WER
Our Trained Model	10.5
Whisper	14.6
Microsoft ASR	18.6
Google ASR	20.4
Amazon ASR	25.4

Table 9: ASR WER on different models

- LLM Accuracy: We benchmarked various LLM models using approximately 500 different internal transcripts to evaluate their performance. Table 10 illustrates the key comparison of different LLM's on different matrices:

Parameters	LLaMA 3 (8B)	Google Gemini 2	Microsoft Phi-2	Sarvam AI (sarvam-2b-v0.5)
Sentiment Analysis Accuracy	94%	80%	70%	50%
Problem Resolution Detection Accuracy	89%	75%	68%	45%
Keyword Detection Precision	91%	78%	65%	60%
Agent Behavior Evaluation Accuracy	87%	70%	60%	40%
Enhancement Opportunities	Excellent	Basic suggestions	Minimal suggestions	Basic, often inaccurate
New Product Features Identification	High accuracy	Limited effectiveness	Struggled to identify features	Moderate insights, struggled with complexity

Table 10: LLM Accuracy on different parameters

- LLaMA 3 (8B): Demonstrates the highest performance across all key metrics, excelling in sentiment analysis, problem resolution detection, and keyword detection precision. It effectively provides actionable insights for enhancement opportunities and identifies new product features with high accuracy, making it the top choice for Hindi call analysis.
- Google Gemini: Offers moderate performance but struggles to handle complex tasks and sentiment analysis effectively, especially in Hindi. Its suggestions for improvement and feature identification are basic and lack sophistication.
- Microsoft Phi-2: Shows limited effectiveness in understanding customer sentiment and problem resolution, with minimal suggestions for service

enhancement and product features. Performance is generally subpar compared to LLaMA 3.

- Sarvam AI: It falls short in handling complex sentiment and behavior evaluation. It provides basic insights and struggles with nuanced feature identification, making it less reliable for detailed call analysis.

5.3. Discussion of Findings

The findings demonstrate that the on-premise ASR system, when integrated with speaker diarization and LLM-based insights, provides a cost-effective and efficient solution for customer care call analytics in Hindi language. The system addresses the specific challenges of noisy environments, and real-time requirements, which are often limitations of other commercial solutions.

- Significance of On-Premise Deployment: The on-premise approach ensures data privacy and security, which are critical in customer care settings. The low complexity compared to advanced models like Whisper allows for reduced resource requirements, making it accessible for small to medium-sized enterprises.
- Enhanced Customer Insights: The LLM's ability to extract and interpret sentiment, agent behavior, and potential improvement areas adds significant value, transforming raw audio data into actionable business intelligence.
- Limitations: While the model performs well, the WER can still be improved, especially in cross talk environments. Additionally, diarization accuracy, although sufficient, may struggle with highly overlapping speech, or in noisy environment.
- Implications: The solution can revolutionize customer service operations by automating insights extraction, enhancing agent training, and identifying areas for process improvement, ultimately leading to higher customer satisfaction and reduced operational costs.

Overall, the developed system presents a promising step forward in ASR and call analytics for Hindi language, offering a balanced approach between performance, cost, and privacy which can also be scale up for different Indian languages.

The experimental results confirm the effectiveness of the developed ASR and analytics system in a customer care context. The low WER and high diarization accuracy validate the system's ability to handle diverse and noisy call environments effectively. LLaMA 3's superior performance in sentiment analysis, problem resolution, and keyword detection underscores its suitability for in-depth call analytics. The comparative analysis highlights significant improvements over existing solutions, particularly for Hindi language processing. The system's efficient real-time processing and scalability demonstrate its potential for broad deployment in customer care centers, offering both operational efficiency and valuable insights into customer interactions.

6. Chapter 6: Conclusion and Future Work

6.1. Summary of Achievements

This research successfully developed a robust, on-premise Automatic Speech Recognition (ASR) system tailored for customer care centers supporting Indian languages. Key achievements include:

- ASR Model Development: A custom ASR model using NVIDIA NeMo's Conformer architecture, effectively transcribing customer care conversations with a WER of 10.5.
- Speaker Diarization: Integration of Silero-VAD enabled accurate speaker segmentation, enhancing the clarity of transcripts by differentiating between customer and agent speech.
- LLM-Based Insights: The implementation of the Llama 3 8B model for extracting insights such as sentiment analysis, problem resolution status, keyword detection, agent behavior, and enhancement opportunities.
- Cost Efficiency: Achieved significant cost reductions by avoiding paid APIs, demonstrating a viable on-premise solution with privacy advantages.
- Real-Time Performance: Successfully maintained low latency in streaming mode, suitable for live customer service applications.

6.2. Limitations of the Study

While the research has met its objectives, certain limitations were identified:

- WER in Noisy Environments: Although the model performed well, the WER of 10.5 could still be improved, particularly in extremely noisy with cross-talk conditions common in some real-world customer service settings like by applying noise filtration before doing any preprocessing on the audio.
- Diarization Accuracy: While Silero-VAD performed effectively, accuracy drops when dealing with highly overlapping speech, which can impact the clarity of speaker-specific insights.
- Scalability for Large Scale Operations: The on-premise model, while efficient for small to medium-sized centers, may require further optimization or hardware scaling for larger customer care operations.

6.3. Contributions to the Field

The research contributes significantly to the fields of ASR, natural language processing, and customer care analytics, particularly for Indian languages:

- Privacy-Centric On-Premise Solution: The focus on on-premise deployment ensures data security and privacy, offering an alternative to cloud-dependent solutions that may not meet stringent data protection requirements.
- Integrated Insights with LLMs: By combining ASR with advanced LLMs, the research introduces a novel approach to extracting actionable insights directly from customer interactions, enhancing service quality and operational decision-making.
- Customized ASR: The study provides an efficient, low-cost ASR system specifically designed for Hindi language, filling a critical gap where most commercial ASR solutions are English-centric.

6.4. Recommendations for Future Research

Future work could explore several areas to enhance the current system:

- Real-Time Analytics and Dashboard Integration: Developing user-friendly dashboards that provide real-time insights and analytics based on LLM outputs would make the solution more accessible for decision-makers.
- Improving ASR Accuracy in Noisy Conditions: Further research into advanced noise reduction in cross-talk environment could reduce the WER in challenging environments.
- Enhanced Diarization Techniques: Developing or integrating more sophisticated diarization methods could improve speaker separation accuracy, particularly in scenarios with overlapping speech, or in case of mono channel audio.
- Broader Language Support: Extending the ASR system to support additional Indian languages and dialects would further enhance its applicability.
- Exploring Multimodal Approaches: Combining audio with visual or contextual data could provide deeper insights into customer interactions, improving overall system performance.

This research lays the foundation for future innovations in multilingual ASR and customer service analytics, offering a scalable, privacy-focused solution with significant potential for broader industry adoption.

Bibliography & References

[1] OpenAI, et al. GPT-4 Technical Report. 2023. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2303.08774.

[2] Giannakopoulos, Theodoros. ‘pyAudioAnalysis: An Open-Source Python Library for Audio Signal Analysis’. PLOS ONE, edited by Gianni Pavan, vol. 10, no. 12, Dec. 2015, p. e0144610. DOI.org (Crossref), https://doi.org/10.1371/journal.pone.0144610.

[3] Kuchaiev, Oleksii, et al. NeMo: A Toolkit for Building AI Applications Using Neural Modules. 2019. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1909.09577.

[4] Radford, Alec, et al. Robust Speech Recognition via Large-Scale Weak Supervision. 2022. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2212.04356.

[5] Povey, Daniel, et al. ‘The Kaldi Speech Recognition Toolkit’. IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, Jan. 2011.

[6] Hannun, Awni, et al. Deep Speech: Scaling up End-to-End Speech Recognition. 2014. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1412.5567.

[7] Baevski, Alexei, et al. Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. 2020. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2006.11477.

[8] Ravanelli, Mirco, et al. SpeechBrain: A General-Purpose Speech Toolkit. 2021. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2106.04624.

[9] Yao, Zhuoyuan, et al. WeNet: Production Oriented Streaming and Non-Streaming End-to-End Speech Recognition Toolkit. 2021. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2102.01547.

[10] Jiang, Albert Q., et al. Mistral 7B. arXiv:2310.06825, arXiv, 10 Oct. 2023. arXiv.org, https://doi.org/10.48550/arXiv.2310.06825.

[11] Touvron, Hugo, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. 2023. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2307.09288.

[12] Dubey, Abhimanyu, et al. The Llama 3 Herd of Models. 2024. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2407.21783.

[13] Gemini Team, et al. Gemini: A Family of Highly Capable Multimodal Models. 2023. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2312.11805.

[14] Gemini Team, et al. Gemini 1.5: Unlocking Multimodal Understanding across Millions of Tokens of Context. 2024. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2403.05530.

[15] Li, Yuanzhi, et al. Textbooks Are All You Need II: Phi-1.5 Technical Report. Sept. 2023, https://www.microsoft.com/en-us/research/publication/textbooks-are-all-you-need-ii-phi-1-5-technical-report

[16] Gunasekar, Suriya, et al. Textbooks Are All You Need. June 2023, https://www.microsoft.com/en-us/research/publication/textbooks-are-all-you-need/.

[17] Scheidt, Scott, and Q. B. Chung. 'Making a Case for Speech Analytics to Improve Customer Service Quality: Vision, Implementation, and Evaluation'. International Journal of Information Management, vol. 45, Apr. 2019, pp. 223–32. DOI.org (Crossref), https://doi.org/10.1016/j.ijinfomgt.2018.01.002.

[18] Rafaeli, Anat, et al. 'The Impact of Call Center Employees' Customer Orientation Behaviors on Service Quality'. Journal of Service Research, vol. 10, no. 3, Feb. 2008, pp. 239–55. DOI.org (Crossref), https://doi.org/10.1177/1094670507306685.

[19] Ahmed, Ishrat, et al. 'Text Summarization for Call Center Transcripts'. Intelligent Systems and Applications, edited by Kohei Arai, vol. 822, Springer Nature Switzerland, 2024, pp. 542–51. DOI.org (Crossref), https://doi.org/10.1007/978-3-031-47721-8_36.

[20] Ardila, Rosana, et al. Common Voice: A Massively-Multilingual Speech Corpus. 2019. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.1912.06670.

[21] Bhogale, Kaushal Santosh, et al. Effectiveness of Mining Audio and Text Pairs from Public Data for Improving ASR Systems for Low-Resource Languages. 2022. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2208.12666.

[22] Javed, Tahir, et al. IndicSUPERB: A Speech Processing Universal Performance Benchmark for Indian Languages. 2022. DOI.org (Datacite), https://doi.org/10.48550/ARXIV.2208.11761.

[23] Diwan, Anuj, et al. MUCS 2021: Multilingual and Code-Switching ASR Challenges for Low Resource Indian Languages. 2021, pp. 2446–50. www.isca-archive.org, https://doi.org/10.21437/Interspeech.2021-1339.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Check list of items for the Final report

a)	Is the Cover page in proper format?	Y / ~~N~~
b)	Is the Title page in proper format?	Y / ~~N~~
c)	Is the Certificate from the Supervisor in proper format? Has it been signed?	Y / ~~N~~
d)	Is Abstract included in the Report? Is it properly written?	Y / ~~N~~
e)	Does the Report contain a summary of the literature survey?	Y / ~~N~~
f)	Does the Table of Contents page include chapter page numbers?	Y / ~~N~~
i.	Are the Pages numbered properly?	Y / ~~N~~
ii.	Are the Figures numbered properly?	Y / ~~N~~
iii.	Are the Tables numbered properly?	Y / ~~N~~
iv.	Are the Captions for the Figures and Tables proper?	Y / ~~N~~
v.	Are the Appendices numbered?	Y / ~~N~~
g)	Does the Report have Conclusion / Recommendations of the work?	Y / ~~N~~
h)	Are References/Bibliography given in the Report?	Y / ~~N~~
i)	Have the References been cited in the Report?	Y / ~~N~~
j)	Is the citation of References / Bibliography in proper format?	Y / ~~N~~

Declaration by Student:
I certify that I have properly verified all the items on this checklist and ensured that the report is in the proper format as specified in the course handout.

Signature of the Student
Tanmay Jain
2022AA05306

Place: Pune
Date: 8 Sepetember, 2024