Harf Ba Harf

Urdu Transcription and Diarization

by

Talal Khan - 25253
Saad Lakhani - 24471
Nimra Humayun - 24450
Maham Ahmed - 24518

Advised by

Dr. Zain Uddin
Lecturer

Co-advised by

Mr. Adil Saleem
PhD scholar, research assistant and alumni

Department of Computer Science
School of Mathematics and Computer Science (SMCS)

Industry mentorship by **Softech Worldwide**
Spring Semester 2025
Institute of Business Administration (IBA) Karachi Pakistan

Contents

Abstract

Conversational speech is what we call normal, everyday, spoken speech. To represent this audio data in a text form we require the methods of transcription and speaker diarization to navigate conversations and gain valuable insight from them. While products in English and other high-resource languages are abundant, Urdu users face a lack of integrated systems that perform these tasks. This gap limits accessibility and productivity for millions of speakers of the language, especially professionals, students, and the hearing-impaired. Using publicly available, fine-tuned models, we evaluate on public and locally sourced data sets to develop a complete product that is an accurate, intuitive and user-friendly package to Pakistan's growing technology needs of Urdu transcription and speaker diarization.

Chapter 1

Introduction

Urdu, the 11th most spoken language globally (Eberhard et al. (2020)), has long been neglected in the domain of conversational ASR and diarization. While real-time transcription tools are prevalent for English and other high-resource languages, Urdu-speaking professionals, students, and researchers often lack reliable tools for capturing spoken content. This results in lost productivity, reduced accessibility for the hearing-impaired, and exclusion from global digital advancements. ASR converts spoken audio to written text, while speaker diarization identifies "who spoke when" (Anguera et al. (2012)). Together, these technologies are vital in contexts like corporate meetings, academic lectures, and interviews. Harf ba Harf aims to bridge this gap by developing a comprehensive solution that addresses both transcription and diarization for Urdu, along with features like dialect support, noise robustness, and transcription export.

Chapter 2

Proposed Approach

System Overview

The Harf ba Harf system is built on a client-server architecture designed for Urdu audio transcription and diarization tasks. Users interact through a Flutter application, which integrates Google Calendar for scheduling. The client app communicates with a FastAPI backend server, which acts as a modular pipeline connecting to advanced pretrained models specialized in speech recognition, speaker diarization, and summarization. The backend is exposed via ngrok, while Firebase manages data persistence and configuration. Additionally, the backend triggers a Zoom bot (implemented using FFMPEG) that joins Zoom calls, records audio, and forwards it to the AI models for processing.

2.1 Modular Architecture and API Layer

The backend architecture is structured using FastAPI, a high-performance web framework tailored for efficient machine learning model serving.

Dynamic public endpoint exposure is achieved via ngrok, which tunnels localhost endpoints to the public internet. Firebase Remote Config seamlessly manages dynamic ngrok URLs, ensuring the mobile app can reliably fetch backend URLs without app redeployment.

The backend connects to firebase directly, writing the transcription output directly to the relevant documents. The status of each meeting is recorded, and displayed in the app.

2.2 Google Calendar Integration and Zoom Meeting Bot

The system provides calendar management services as well calendar integration capabilities, enabling users to efficiently schedule, manage, and join Zoom meetings directly from the app.

Users can synchronize their meetings with popular calendar platforms such as Google, ensuring seamless scheduling and organization. A dedicated meeting bot, implemented suing FFMPEG, is deployed to automatically join meetings when prompted from the app, record audio, and subsequently send the recordings to the backend for processing.

2.3 Audio Upload Methods

Users can provide audio via:

- **Automated Bot:** Joins meetings, records audio, and uploads directly.
- **In-app Recording:** Manual initiation of audio recording.
- **Manual File Upload:** Uploading pre-recorded audio files.

Processed data is stored in Firebase and retrieved by the mobile app.

2.4 Model Pipeline: Diarization and Transcription

2.4.1 Speaker Diarization

The diarization pipeline utilizes pyannote.audio `pyannote/speaker-diarization-3.1` Plaquet and Bredin (2023). This toolkit systematically performs voice activity detection (VAD) using 5-second sliding windows. It then extracts speaker-specific embeddings with a ResNet-based neural network, generating fixed-dimensional vectors (x-vectors). It then applies agglomerative hierarchical clustering (AHC) to group embeddings into distinct speakers based on cosine similarity.

2.4.2 Audio Chunking and Preprocessing

In this stage, the speaker segments are concatenated by continuity and then exported as temporary WAV files via pydub. Each WAV file representt a segment from the diairzation output. The filesa are then normalized using torchaudio, and subsequently batched for transcription.

2.4.3 Transcription

Using `openai/whisper-large-v3-turbo` (Radford et al. (2022)) model via Hugging Face: Diarization is conducted first, followed by transcription, as performing these tasks separately and subsequently aligning their timestamps is highly resource-intensive and impractical. Audio segments identified from diarization are batch-processed through the model pipeline provided by Hugging Face, delivering accurate Urdu transcriptions.

2.5 Summarization Engine

The system incorporates a summarization pipeline powered by the `Mudasir692/bart-urdu-summarizer` (Mudasir692 (2024)) model specifically fine-tuned for Urdu language summarization.The model concatenates transcript segments then proceeds to utlizie the MBart50 tokenizer for input tokenization. Then it generates coherent summaries via Transformer decoder layers, optimized to produce contextually relevant meeting summaries in Urdu.

2.6 Firebase Integration and Schema

Firebase Firestore stores user data in: `users/{userID}/meetings/{meetingID}` This ensures that users are only able to access their own data.

Each meeting document includes:

- Title
- Date of meeting
- Duration (seconds),
- Summary,
- Status,
- Transcript entries (Speaker ID, text, timestamp).

Firebase authentication ensures secure data storage and retrieval.

2.7 ngrok Tunnel Management via Firebase

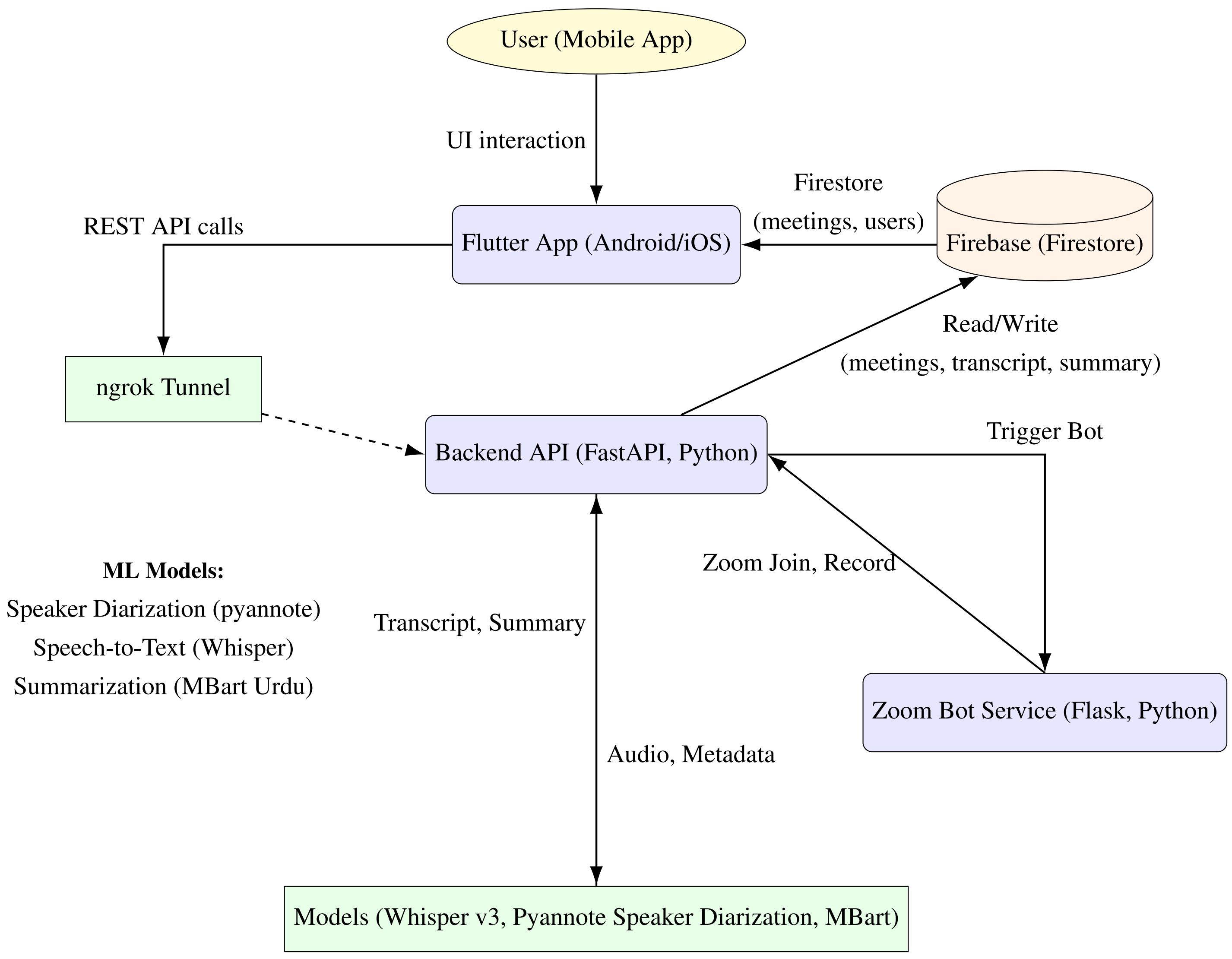
Due to ngrok's dynamic URL generation, backend endpoints are managed through Firebase Remote Config. This approach ensures real-time URL synchronization without requiring static IPs or domain registrations during the development phase, enhancing scalability for future deployments.

2.8 Subsystem and Hardware Deployment

Subsystem	**Hardware Deployment**

User Interface (Flutter App)	Client mobile device (Android/iOS); communicates via RESTful API
Backend API (FastAPI)	Ngrok; manages requests and processing
Speech Processing	GPU-enabled collab instance; handles diarization and transcription
Summarization Engine	Backend microservice; generates summaries
Database (Firebase)	Firebase NoSQL database

2.9 Architecture Diagram



User (Mobile App)
UI interaction
Flutter App (Android/iOS)
Firestore (meetings, users)
Firebase (Firestore)
REST API calls
ngrok Tunnel
Read/Write (meetings, transcript, summary)
Backend API (FastAPI, Python)
Trigger Bot
Zoom Join, Record
ML Models:
Speaker Diarization (pyannote)
Speech-to-Text (Whisper)
Summarization (MBart Urdu)
Transcript, Summary
Zoom Bot Service (Flask, Python)
Audio, Metadata
Models (Whisper v3, Pyannote Speaker Diarization, MBart)

Chapter 3

Experimental Settings

We evaluated the performance of our application by running the frontend and backend components on easily avaible hardware. We tested both functionality and performance in both using the application and for requests to the backend for transcription and diarization.

To evaluate the performance and generalizability of our Urdu transcription and diarization system, we used publicly available Urdu datasets to check how they perform against known metrics.

3.1 Evaluation Datasets

Transcription Dataset: We used the public dataset Urdu Audio Dataset by HowManyMore **?**howmannymore$_{u}rdu_{a}udiodataset), whichconsistsofover50,000manuallytranscribedUrduaudiosampl$
$truthtext, allowingprecisemeasurementofmodelerrorusingWER(WordErrorRate)andCER(Charac$

Diarization Dataset: For evaluating speaker diarization, we utilized the ”Conversations in the Wild” dataset proposed by Agha Ali Raza et al (Zaheer et al. (2025)). This dataset includes complex, real-world Urdu conversations annotated with:

- `.rttm` (Rich Transcription Time Marked files for speaker labels)
- `.lst` (List of audio files)
- `.uem` (Un-partitioned Evaluation Map specifying valid regions for scoring)

This dataset reflects noisy, real-world, multi-speaker conversational setups, providing a robust benchmark for evaluating diarization accuracy.

3.2 Hardware & Software Setup

Backend & Models: Core i3 13600k, Nvidia RTX 4070TI 12 GB VRAM, 32 GB DDR5 RAM and a Google Colab T4 Instance with 15GB of VRAM
Software: Python 3.9+, PyTorch, Hugging Face Transformers, pyannote.audio, Firebase Admin SDK.
Frontend: Flutter (Dart), tested on a mid-range android smarttphone running Android 9.
Tunnel: ngrok for exposing FastAPI endpoint dynamically, synced via Firebase Remote Config.

Chapter 4

Results and Discussion

Our evaluation focuses primarily on transcription accuracy and speaker diarization consistency under varied real-world Urdu audio conditions.

4.1 Transcription Evaluation

Using the 4,031 audio samples from the selected dataset, we computed Word Error Rate (WER) and Character Error Rate (CER). These metrics were aggregated into buckets (intervals of 0.1) to show distribution.

WER Results

WER Interval	Count	Percentage
0.0	674	16.72%
0.1	713	17.68%
0.2	754	18.70%
0.3	504	12.50%
0.4	322	7.99%
0.5	301	7.47%
0.6	207	5.13%
0.7	114	2.83%
0.8	109	2.70%
0.9	13	0.32%
1.0	320	7.94%

Table 4.1: WER Distribution Across 4,031 Samples

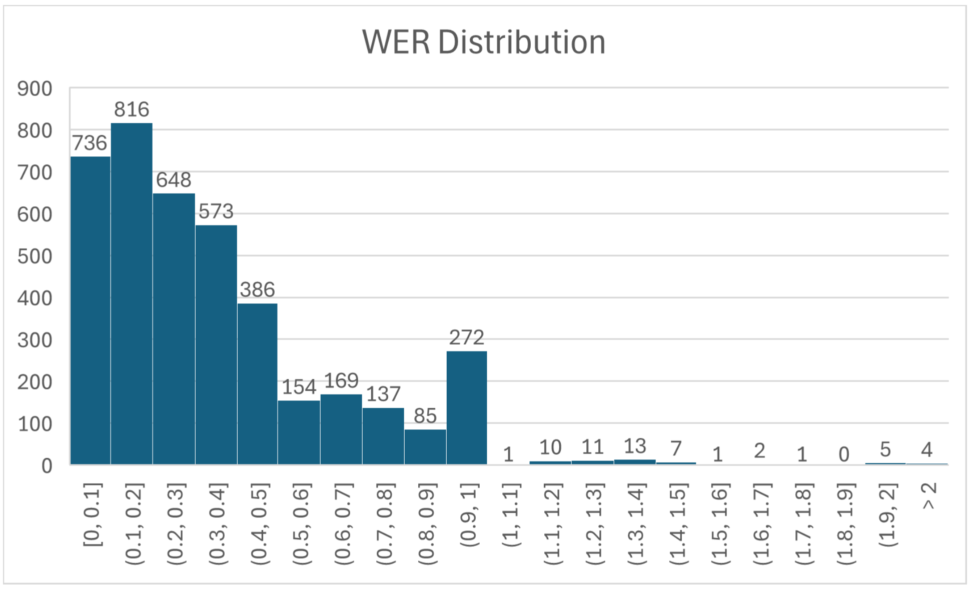


Figure 4.1: WER Distribution Across 4,031 Urdu Audio Samples

Insights

Over 52% of samples had WER ≤ 0.2, indicating high intelligibility while Only 7.94% of samples had complete failure (WER=1.0), often due to extremely noisy or short clips. The long tail beyond WER 0.5 possibly indicates challenges in accent diversity and overlapping speech.

CER Results

CER Interval	Count	Percentage
0.0	2,392	59.33%
0.1	854	21.18%
0.2	281	6.97%
0.3	141	3.50%
0.4	66	1.64%
0.5	63	1.56%
0.6	41	1.02%
0.7	22	0.55%
0.8	37	0.92%
0.9	48	1.19%
1.0	86	2.13%

Table 4.2: CER Distribution Across 4,031 Samples

Insights

The CER ≤ 0.1 group accounts for over 80% of samples which implies that Whisper performs very well at the character level, even when the WER is high.. This demonstrates Whisper's ability to phonetic complexity in Urdu script, though minor word-level hallucinations persist in overlapping speech regions.

4.2 Diarization Evaluation

Diarization evaluation was conducted by alighiing predicted speaker segments against RTTM references. UEM files restricted the scoring to valid regions (excluding silence and unknown).

DER Distribution

Diarization Error Rate (DER) is a standard metric used to evaluate the accuracy of speaker diarization systems (Anguera et al. (2012)). It measures the percentage of time that the diarization system's output differs from the reference (ground truth) speaker labels over the duration of an audio recording.

DER Interval	Percentage
0.0	61.29%
0.1	16.13%
0.2	9.68%
0.3	6.45%
0.6	3.23%
0.7	3.23%

Table 4.3: Diarization Error Rate (DER) Distribution on Conversations in the Wild Dataset

Insights

Majority of files have DER below 0.2, with 61.29% in the lowest bucket even though some files exhibit higher DER, indicating room for improvement in speaker segmentation. In the pipeline itself, Using diarization prior to transcription simplified timestamp alignment and improved processing efficiency.

Chapter 5

Limitations and Future Work

While *Harf Ba Harf* demonstrates a robust proof-of-concept with strong transcription and diarization capabilities tailored for Urdu, certain limitations currently restrict its usage as a commercial product.

5.1 Infrastructure & Hosting Limitations

At present, the backend system—including heavy model pipelines for transcription and diarization—is deployed using ngrok and local GPU machines or Google Collab. While this setup works for experimentation and early user testing, it is not suitable for a fully deployed commercial application due to the lack of stability provided by ngrok and the high costs of persistent GPU usage on the cloud.

5.2 Model Limitations and Fine-Tuning Needs

The model used for this project, namely Whisper, Pyannote, MBart are off-the-shelf models. While Whisper handles general Urdu well, it occasionally struggles with local dialects or fast paced speech. Similarly, Pyannote's speaker embeddings are based on English-centric data, which can occasionally misclassify speakers with similar pitch or articulation styles.

Not a lot of domain-specific fine-tuning has been done, primarily due to limited GPU access and lack of large-scale Urdu datasets. A future frameowrk utliizing Harf ba Harf data (with the relevant consent) to fine tune the models will greatly benefit the generalizability of the product.

Bibliography

Anguera, X., S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals (2012, 02). Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech Language Processing 20*, 356–370.

Eberhard, D., G. Simons, and C. Fennig (2020, 02). Ethnologue: Languages of the world, 23rd edition.

Mudasir692 (2024). bart-urdu-summarizer. `https://huggingface.co/Mudasir692/bart-urdu-summarizer`. Accessed: 2025-06-03.

Plaquet, A. and H. Bredin (2023). Powerset multi-class cross entropy loss for neural speaker diarization. In *Proc. INTERSPEECH 2023*.

Radford, A., J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever (2022). Robust speech recognition via large-scale weak supervision.

Zaheer, N., A. A. Raza, and M. Shabbir (2025). Conversations in the wild: Data collection, automatic generation and evaluation. *Computer Speech & Language 89*, 101699.

Chapter 6

Contributions

Contribution Matrix

Task	**Talal**	**Saad**	**Nimra**	**Maham**
Requirements Specification	25%	25%	25%	25%
Use Case Design	-	-	80%	20%
UI/UX	-	-	10%	90%
Data Handling	50%	-	50%	-
Backend Development	50%	50%	-	-
AI Model Training	80%	10%	10%	-
Bot and Automation Development	-	-	100%	-
App Design	-	-	10%	90%
App Development	-	100%	-	-
Testing	70%	30%	-	-
Documentation	25%	25%	25%	25%