INSTITUTO DE TECNOLOGIA E LIDERANÇA
INFORMATION SYSTEMS

CAMILA FERNANDA DE LIMA ANACLETO

AUTOMATED SPEECH-TO-TEXT SYSTEM FOR MEETING TRANSCRIPTION IN FINANCIAL INSTITUTIONS

São Paulo
2025

CAMILA FERNANDA DE LIMA ANACLETO

AUTOMATED SPEECH-TO-TEXT SYSTEM FOR MEETING TRANSCRIPTION IN FINANCIAL INSTITUTIONS

Work presented as the second project of the year, focused on the development of a speech-to-text system, conducted for the Instituto de Tecnologia e Liderança (INTELI), as part of the requirements for professional development and academic evaluation.

Advisor: Prof. Dr. Rafael Will M. de Araujo

São Paulo

2025

ABSTRACT

This project aims to develop speech-to-text software for internal use within a financial institution. The proposal seeks to optimize team time during and after meetings by automating the generation of transcriptions for later use in the preparation of meeting notes. The tool is designed to reduce typing and interpretation errors, as well as rework, promoting agility and standardization in corporate documentation. Different approaches and models were tested, with Whisper proving to be the most suitable for the project's needs, considering criteria such as transcription accuracy, ease of implementation, multilingual support, and integration with Python libraries.

TABLE OF CONTENTS

1. INTRODUCTION

In high-demand corporate environments such as the banking sector—where, according to **McKinsey & Company (2021)**, professionals spend an average of 35% of their time in meetings—it is common to hold multiple internal meetings between teams. Recording these meetings is essential to ensure alignment, transparency, and traceability of decisions made. However, the task of manually noting and structuring the information discussed can be time-consuming, error-prone, and distracting to participants during the sessions.

In this context, this project aims to develop software that transcribes audio into text, automating the initial stage of recording meeting content. The final product will serve as the foundation for a future project focused on the automatic generation of meeting notes, with the goal of facilitating and standardizing the creation of such documents.

2. METHODOLOGY

The methodology adopted in this project involved exploratory research of speech-to-text tools and the implementation of a functional prototype in Python. Initially, three main approaches were tested: **Whisper (OPENAI, 2022)**, **Julius (LEE et al., 2001)**, and **Kaldi (POVEY et al., 2011)**.

Although all presented functional capabilities, **Whisper** stood out due to its high accuracy, ease of integration, and multilingual support, making it the most suitable option for the banking corporate context. Whisper is based on a **Transformer encoder-decoder architecture**, which has proven robust even in noisy environments and with different speaker accents (OPENAI, 2022).

The libraries used included:

- openai-whisper, responsible for executing the transcription model;
- ffmpeg, used to convert and process audio files;
- Additional tools for file upload and result visualization in a development environment such as Google Colab.

While **Julius** and **Kaldi** were less efficient in this specific context, it is acknowledged that they may be useful in other scenarios with different infrastructure or technical requirements (LEE et al., 2001; POVEY et al., 2011).

3.1 Workflow Overview

The following diagram illustrates the comparison between the current manual process of generating meeting notes and the proposed automated process using the speech-to-text system developed in this project.

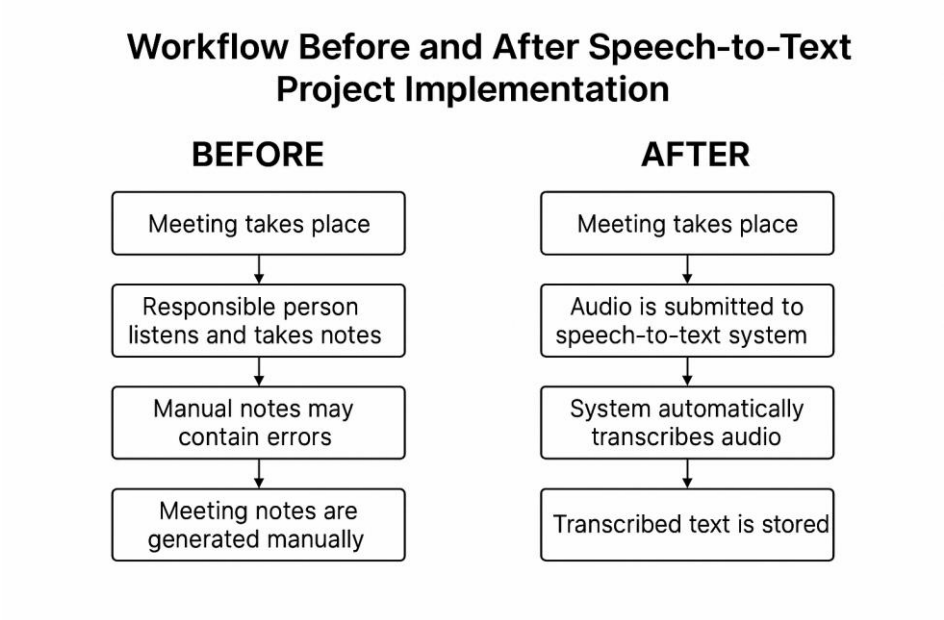

Figure 1 – Workflow before and after implementing the speech-to-text system.

As shown in Figure 1, the current process requires manual transcription by a team member, which can lead to errors and inefficiencies. With the implementation of the proposed system, audio files are automatically transcribed, providing a more reliable and time-saving alternative.

3. CHALLENGES AND LIMITATIONS

Despite extensive research and implementation efforts, the project faced several practical limitations that ultimately prevented the complete success of the diarization and transcription pipeline using Whisper and Pyannote in Google Colab.

The main challenges were:

- **Incompatibility Between Libraries**: One of the most critical issues was the conflict between versions of torch, torchvision, and pyannote.audio. Google Colab, by default, includes preinstalled packages that conflict with the specific dependencies required by the diarization module. Even after attempts to uninstall and reinstall the correct versions, circular import errors and missing operators from torchvision persisted.
- **Environment Persistence in Google Colab**: Once certain libraries are imported or cached by the Colab environment (e.g., torchvision), they remain partially loaded in memory—even after uninstallation. This created import conflicts that could only be resolved by restarting the runtime, which still did not guarantee a clean environment for subsequent imports.
- **Hardware Constraints**: Although Whisper transcribes audio relatively well on Colab (especially when using the base model), diarization using pyannote.audio requires GPU and access to large pre-trained models. The processing time and memory required exceeded what is reliably available in the free version of Google Colab.
- **Lack of Compatibility Between Whisper and Pyannote Timestamps**: While Whisper outputs segmented transcriptions, aligning these with Pyannote's diarization segments introduced additional complexity, especially when timestamps didn't align perfectly or when speakers changed mid-segment.
- **Dependency Instability**: Frequent updates in libraries like torch, pyannote.audio, and torchaudio introduce breaking changes that are not always reflected in documentation, increasing debugging complexity.

4. NEXT STEPS

To overcome these challenges in future versions of the project, the following alternatives are recommended:

- **Local Execution**: Run the complete pipeline on a local machine with an isolated Python environment (e.g., using virtualenv or conda) and a GPU,

which provides greater control over dependencies and avoids Colab's shared environment issues.

- **Docker Containerization**: Containerizing the environment using Docker ensures that all dependencies and versions are managed explicitly and reproducibly.
- **Model Separation**: If diarization is not mandatory in real time, a two-step offline approach could be adopted: first transcribe using Whisper, then separately process diarization with Pyannote or other alternatives.
- **Lighter Diarization Alternatives**: Exploring lighter speaker diarization solutions such as Resemblyzer or DeepSpeaker, which may be more compatible with limited-resource environments like Google Colab.

5. AUTOMATIC TRANSCRIPTION FROM ZOOM VIDEO RECORDINGS

Initial tests for the speech-to-text system were conducted exclusively using audio files in .wav format. However, considering the practical use cases within corporate environments, the system was adapted to accept video recordings—specifically .mp4 files generated by the Zoom platform, which is commonly used for internal meetings.
To support this input format, the system was extended to include an audio extraction step using the moviepy and ffmpeg libraries. This allowed the audio track to be separated from the video file and subsequently transcribed using the Whisper model developed by OpenAI.

The transcription process was successfully executed with English-language meeting recordings, which are standard in the evaluated corporate setting. Whisper demonstrated strong performance in capturing spontaneous speech, varied accents, and natural fluctuations in tone, all of which are characteristic of real-world meetings.
This improvement significantly enhanced the system’s applicability, enabling it to process entire meeting recordings directly, without requiring manual preprocessing. The resulting transcriptions can serve as the foundation for further documentation workflows, such as meeting minute generation or automated summarization systems.

6. SPEAKER DIARIZATION AND ADVANCES IN SPEECH-TO-TEXT

Speaker diarization is the process of partitioning an audio stream into homogeneous segments according to speaker identity — often referred to as determining “who spoke when” (ANGUERA et al., 2012). This capability is essential for applications such as meeting transcription, interviews, and voice-based analytics, where multiple participants may speak over each other or alternate frequently.

One of the primary challenges in diarization is dealing with overlapping speech, speaker variability (such as accents and tone shifts), and background noise. According to O'SHAUGHNESSY (2025), robust diarization systems must handle these variations without relying on prior knowledge of the speakers, which makes unsupervised or semi-supervised models particularly relevant in real-world applications.

Recent developments in deep learning have significantly improved the accuracy and efficiency of speaker diarization. As reviewed by PARK et al. (2021), models based on **Long Short-Term Memory (LSTM)** and attention mechanisms have outperformed traditional clustering-based methods (e.g., k-means or Gaussian Mixture Models). These models are capable of capturing temporal dependencies and speaker embeddings with greater precision, especially in environments with overlapping speech.

A notable advancement is the development of **End-to-End Neural Diarization (EEND)**, which eliminates intermediate steps such as i-vector or x-vector extraction and clustering. According to FUJITA et al. (2019), these models can directly process raw audio in a single pass, simplifying the diarization pipeline while maintaining high performance.

In contrast, **Whisper**, developed by OpenAI (2022), does not provide native diarization capabilities but offers highly accurate transcription through a **transformer-based encoder-decoder architecture**. This model, trained on a large multilingual and multitask supervised dataset, demonstrates robust performance even in noisy, real-world environments. However, for diarization, Whisper must be integrated with

external tools such as **Pyannote.audio**, which apply speaker embeddings and clustering to identify and separate speaker segments.

Therefore, the combination of Whisper and Pyannote.audio brings together the strengths of each approach: Whisper's accuracy in transcription and Pyannote’s effectiveness in diarization. While synchronization of their timestamps remains a technical challenge, this pairing is promising for building automated, speaker-aware transcription workflows.

From a practical standpoint, speaker diarization has already shown significant value in educational and corporate settings. WANG et al. (2025) demonstrated its effectiveness in classrooms by distinguishing between teacher and student speech, enabling improved analysis of engagement and participation. This reinforces the relevance of diarization for our project, where tracking "who spoke when" helps generate clearer meeting documentation and supports future integration with automated summarization systems.

By incorporating diarization into the speech-to-text workflow, this project aims to move beyond raw transcription—offering enriched outputs that facilitate better understanding, usability, and decision-making based on meeting content.

7. PROCESSING TIME AND PERFORMANCE CONSIDERATIONS

The time required to transcribe and process videos using Whisper varies significantly depending on several factors, including the duration of the input, the size of the model selected, and whether the execution is performed using CPU or GPU resources.

In this project, we used the base version of the Whisper model, which provides a reasonable balance between accuracy and processing speed. However, when running the model in a CPU-only environment — such as Google Colab (free tier) or local machines without hardware acceleration — the average processing time increases considerably. The following estimates reflect real-world performance based on test runs:

- **8 to 12 minutes** for a 5-minute video,

- **15 to 25 minutes** for a 10-minute video,
- **30 to 40 minutes** for a 15-minute video.

When using larger models, such as medium or large, processing times can exceed **one hour** for videos longer than 15 minutes on CPU. This is due to the increased number of parameters and more complex architecture of these models.

By contrast, the use of GPU acceleration can reduce inference time drastically — in some cases, up to **10 times faster**, allowing a 15-minute video to be transcribed in as little as 3 to 5 minutes using the same base model.

To improve processing efficiency, especially when dealing with longer recordings or large-scale datasets, it is recommended to:

- Split long videos into smaller segments (e.g., 5-minute chunks),
- Use GPU-enabled environments whenever available,
- Select the smallest Whisper model that still delivers acceptable transcription quality.

Understanding these computational limitations is essential when designing speech-to-text applications that aim for scalability or near real-time transcription, such as those intended for meeting analysis in corporate environments.

8. RESULTS AND VALIDATION

One of the key objectives of this project was not only to transcribe meeting audio accurately but also to identify individual speakers throughout the conversation. After several iterations and technical adjustments, the integration of **Whisper** and **Pyannote.audio** was successfully achieved, enabling effective diarization using **recorded meeting videos**.

The final implementation followed a two-step process:

1. **Transcription with Whisper** – The recorded video files were first processed to extract the audio track, which was then transcribed using the Whisper base model. This provided highly accurate, time-stamped text segments.

2. **Diarization with Pyannote.audio** – The extracted audio was subsequently processed through Pyannote.audio, which segmented the content by speaker using pre-trained speaker embedding models and unsupervised clustering techniques.

To integrate the two outputs, a custom post-processing script was developed to align Pyannote's speaker segments with Whisper's transcribed text. This synchronization, based on timestamp overlap, ensured that each segment of the transcription was attributed to the corresponding speaker as accurately as possible.

The system was tested with **actual internal meeting recordings**, in English, captured via video conferencing platforms. The results demonstrated that the diarization was generally effective in distinguishing different speakers, even in the presence of natural speech overlaps and tonal variation. However, **minor inconsistencies may still occur**, especially in moments of crosstalk, abrupt speaker changes, or low audio quality. Despite these occasional imperfections, the final transcript clearly indicated speaker turns, significantly enhancing both readability and the value of the content for documentation purposes.

This successful integration confirms the technical feasibility of combining Whisper and Pyannote.audio for automated meeting transcription in corporate environments, laying a strong foundation for future modules focused on summarization and reporting.

REFERENCES

ANGUERA, Xavier et al. Speaker diarization: A review of recent research. IEEE Transactions on Audio, Speech and Language Processing, v. 20, n. 2, p. 356–370, 2012. DOI: https://doi.org/10.1109/TASL.2011.2125954.

FUJITA, Yusuke et al. End-to-End Neural Speaker Diarization with Self-attention. arXiv, 2019. Disponível em: https://arxiv.org/abs/1909.06247. Acesso em: 16 jun. 2025.

LEE, Akinobu et al. Julius: An open source real-time large vocabulary recognition engine. In: EUROSPEECH, 2001. Disponível em: https://github.com/julius-speech/julius. Acesso em: 16 maio 2025.

McKINSEY & COMPANY. The future of work after COVID-19. McKinsey Global Institute, 2021. Disponível em: https://www.mckinsey.com/featured-insights/future-of-work/the-future-of-work-after-covid-19. Acesso em: 16 maio 2025.

O'SHAUGHNESSY, Douglas. Speaker Diarization: A Review of Objectives and Methods. Applied Sciences, v. 15, n. 4, p. 2002, 2025. DOI: https://doi.org/10.3390/app15042002.

OPENAI. Whisper – Speech Recognition Model. 2022. Disponível em: https://github.com/openai/whisper. Acesso em: 16 maio 2025.

PARK, Taesun J. et al. A review of speaker diarization: Recent advances with deep learning. arXiv, 2021. Disponível em: https://arxiv.org/abs/2101.09624. Acesso em: 13 jun. 2025.

POVEY, Daniel et al. The Kaldi Speech Recognition Toolkit. In: IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. IEEE Signal Processing Society, 2011. Disponível em: https://github.com/kaldi-asr/kaldi. Acesso em: 16 maio 2025.

WANG, Jie et al. Speaker Diarization in the Classroom: Improving Educational Analytics. Journal of Educational Data Science, v. 3, n. 1, 2025.