UNIVERSITY OF GREENWICH

Undergraduate Final Year Project Report

An Investigation into Advanced NLP Techniques for Meeting Transcription and Summarization in a Web-Based System

LA GIA HUY

Bachelor of Science with Honors in Computing

001282717

Table of Contents

Table of Figures

1. Introduction

1.1. Project Subject Introduction

Audio-to-text and text summarization technologies have become important areas of natural language processing (NLP), supporting everything from transcribing meetings and conferences to providing accessibility for the hearing impaired and serving diverse needs such as analyzing audio data in research or education. According to a report from Verified Market Research, the global audio transcription services market was valued at around 20 billion dollars in 2022 and is expected to reach over 40 billion dollars by 2030, growing at a compound annual growth rate (CAGR) of around 10% (Verified Market Research, 2023). This growth is driven by the rise of online audio and video content, the need for regulatory compliance, and the proliferation of remote work models, especially in fields such as healthcare, legal, and media. However, converting audio to text and summarizing its content accurately faces many challenges, especially when dealing with long audio files, multiple speakers, or multiple languages. These challenges require advanced techniques in NLP, including speech recognition models, speaker segmentation, and automatic text summarization, to ensure accuracy and efficiency. This project focuses on researching, investigating, and applying audio-to-text conversion and text summarization techniques and models, to explore the performance of advanced methods in processing and presenting audio data accurately and in an accessible manner.

The main goal of the project is to research and evaluate advanced NLP models for audio-to-text conversion and text summarization, focusing on improving accuracy in speech recognition, speaker separation, and generating concise yet meaningful summaries, especially for practical applications such as meeting note-taking or audio content accessibility.

1.2. Project Objectives

The objective of this project is to develop an end-to-end web-based application that automates the process of audio-to-text transcription, speaker diarization, and text summarization, leveraging state-of-the-art NLP models. The system is designed for practical use cases such as meeting note-taking or interview conversation. To achieve this overarching goal, the project is divided into the following specific objectives:

1.2.1. Objective 1: Build a Secure and Scalable Web Application Framework

- Develop a web server using Flask to handle user interactions for uploading audio files and retrieving transcription results.
- Implement file management workflows to ensure secure storage and retrieval of uploaded audio and generated documents.

- Establish HTTPS security contexts using certifi and urllib library to safely interact with external services like Hugging Face.

1.2.2. Objective 2: Implement Audio Preprocessing Pipeline

- Ensure support for diverse audio formats by integrating FFmpeg for format conversion and audio normalization.
- Validate and manage audio file uploads, ensuring proper file handling and preprocessing readiness.
- Generate unique identifiers and maintain structured directories for input and output management.

1.2.3. Objective 3: Integrate Advanced Speech Recognition (ASR) using Whisper

- Load and run OpenAI's Whisper model to perform transcription on uploaded audio files.
- Support both English-only and multilingual transcription modes.
- Enable dynamic model scaling (tiny, base, medium, large) to balance between processing time and transcription accuracy.

1.2.4. Objective 4: Incorporate Speaker Diarization with WhisperX and PyAnnote

- Use WhisperX for aligning transcribed words with timestamps.
- Integrate PyAnnote for speaker diarization, assigning speaker labels (SPEAKER A, SPEAKER B) to segments.
- Manage edge cases such as overlapping speakers and ensure speaker consistency in outputs.

1.2.5. Objective 5: Design and Implement Text Summarization Workflow

- Apply a pre-trained BART-based summarization model (knkarthick/MEETING_SUMMARY) to create concise summaries.
- Develop custom chunking algorithms that intelligently split transcripts into overlapping segments to comply with model token limits while preserving semantic continuity.
- Generate both global summaries and chunk-wise summaries for comprehensive result presentation.

1.2.6. Objective 6: Automate Document Generation and Output Delivery

- Automate the creation of Word (.docx) reports using python-docx, embedding transcription results, speaker segments, summaries, and processing metadata.
- Provide structured and professionally formatted outputs aligned with document styling (Times New Roman, justified alignment).
- Ensure download capability through secure API endpoints.

1.2.7. Objective 7: Performance Metrics and Logging

- Implement logging mechanisms to capture token counts, processing times, and potential errors.
- Calculate and display performance metrics such as:
- Processing Time (formatted in HH:MM:SS).
- Total Tokens Processed.
- Number of Speakers Detected.
- Provide user-friendly feedback on processing progress and completion.

1.3. Project Plan

Timeline	Works to do	Note Details
Week 1	Project topic refinement and initial research	Define project scope, clarify objectives, and explore related work in the field.
Week 2	Literature review and domain understanding	Study techniques in ASR, speaker diarization, and summarization.
Week 3	Exploration of available tools and frameworks	Research potential models and libraries for transcription and audio processing.
Week 4	Dataset preparation and testing plan design	Collect audio samples and design evaluation strategies for tool comparison.

Week 5	Initial implementation and model experimentation	Begin testing different ASR and diarization models on sample data.
Week 6	Comparative evaluation of model outputs	Analyze model performance to determine the most suitable pipeline.
Week 7	Integration of transcription module	Integrate selected ASR solution into the system with basic output formatting.
Week 8	Speaker segmentation implementation	Develop speaker diarization and align it with transcription outputs.
Week 9	Output structuring with speaker labels	Format transcriptions into readable, labeled dialogue.
Week 10	Research and apply text summarization methods	Experiment with summarization models and generate concise summaries.
Week 11	Document generation module	Create exportable .docx files containing transcripts, summaries, and metadata.
Week 12	Web interface development begins	Set up basic user interface and connect initial backend components.
Week 13	Add upload, configuration, and download features	Allow users to upload audio, choose options, and download results.
Week 14	Prepare for deployment environment	Containerize the application and set up for consistent testing.
Week 15	System testing and error tracking	Conduct end-to-end tests and identify integration issues.

Week 16	Evaluation using quantitative and qualitative methods	Measure WER, inspect diarization and summarization quality.
Week 17	Debugging and performance optimization	Refine models and improve user experience based on testing outcomes.
Week 18	Final integration and interface polishing	Ensure consistency, usability, and clean output formatting.
Week 19	Report drafting and system documentation	Write detailed technical documentation and start final report.
Week 20	Final edits, submission, and preparation for defense	Review all deliverables, prepare presentation, and submit the complete project.

1.4. Project Outcomes

This project will produce a web-based application that integrates advanced NLP techniques for meeting transcription and summarization. The key outcomes are outlined across the following core components:

Transcribed Audio with High Accuracy

- Audio recordings will be transcribed into clean, readable text using OpenAI's Whisper model.
- Transcripts will preserve sentence boundaries and punctuation, enabling easy understanding and usability in professional settings.
- Users can choose transcription modes (English-only or multilingual) and select from multiple model sizes to balance speed and accuracy.

Speaker-Labeled Transcripts

- The system will apply speaker diarization using WhisperX and PyAnnote to differentiate between speakers in multi-party audio.
- Each speech segment will be labeled (SPEAKER A, SPEAKER B, …), providing a clear structure for conversations. This feature will be especially beneficial for meeting minutes, interviews, and collaborative work.

Summaries of Long Conversations

- The system will summarize full transcripts using a BART-based model (knkarthick/MEETING_SUMMARY), creating concise, informative summaries.
- Summaries will include both section-level (chunk-wise) and overall (combined) overviews. These outputs will help users quickly capture key discussion points and decisions from long recordings.

Downloadable .docx Reports

- The application will export results into professional Word documents containing: full transcriptions (with or without speaker labels), summaries, metadata such as processing time, number of detected speakers, and token count.
- All documents will use Times New Roman font, justified alignment, and consistent formatting for easy storage, presentation, or sharing.

1.5. Project Evaluation

The overall approach of this project is considered appropriate and executable within the given timeline, supported by accessible tools and reliable frameworks. By using pre-trained models such as Whisper for transcription, WhisperX and PyAnnote for speaker segmentation, and a BART-based model for summarization, the system builds upon well-established NLP methods. The implementation follows a modular structure, allowing each component to be developed and tested effectively. Evaluation is based on transcription accuracy using Word Error Rate (WER), combined with qualitative assessment of speaker labeling and summary quality. While section 6.4 provides a comprehensive evaluation, this section affirms that the design is valid, the techniques are applicable, and the results demonstrate both consistency and practical relevance in real-world audio processing scenarios.

2. Literature Review

2.1. Domain Knowledge: Automatic Speech Recognition, Speaker Diarization, and Text Summarization

2.1.1. Automatic Speech Recognition (ASR)

Automatic Speech Recognition (ASR) is a key field in Natural Language Processing (NLP) focused on converting spoken language into written text. Early ASR systems were based on statistical models like Hidden Markov Models (HMM) combined with Gaussian Mixture Models (GMM). However, these systems struggled with noisy environments and conversational speech. Recent advancements have shifted towards end-to-end deep learning architectures, notably Transformer-based models, which process raw audio into textual outputs more effectively. OpenAI’s Whisper is an example of a modern ASR system that uses a large multilingual dataset and an encoder-decoder architecture to achieve high accuracy across diverse languages and accents (Radford et al., 2022).

2.1.2. Speaker Diarization

Speaker Diarization is the process of segmenting an audio stream into parts that correspond to each individual speaker. This task is crucial for scenarios involving multiple speakers, such as meetings, interviews, or discussions. Traditional diarization pipelines relied on extracting speaker embeddings (x-vectors) and clustering them using algorithms like Agglomerative Hierarchical Clustering (AHC). However, recent methods such as PyAnnote use deep neural network embeddings and probabilistic clustering techniques, significantly improving diarization accuracy in real-world scenarios (Bredin et al., 2020).

2.1.3. Text Summarization

Text summarization is an NLP task that aims to generate concise versions of longer texts while retaining their core meaning. Summarization techniques are divided into extractive (selecting important sentences) and abstractive (generating paraphrased summaries). Transformer-based models, particularly BART and PEGASUS, have achieved remarkable performance in abstractive summarization. In the context of meeting transcripts, models fine-tuned on datasets like the AMI Meeting Corpus (knkarthick/MEETING_SUMMARY) are capable of producing meaningful summaries even for lengthy conversational transcripts (Lewis et al., 2020; Zhang et al., 2021).

2.1.4. Integrated Processing Pipelines for Speech-to-Summary Applications

An end-to-end pipeline for converting audio into summarized text involves several stages:

- Audio Preprocessing (format conversion, noise normalization using FFmpeg)
- Speech Recognition (ASR with models like Whisper)
- Speaker Diarization (via PyAnnote integrated with WhisperX)
- Text Chunking and Summarization (using BART-based models)

Frameworks like WhisperX orchestrate these steps, enabling the transformation of raw audio into speaker-attributed transcripts and eventually into concise summaries (Bain et al., 2023).

2.2. Literature Research

2.2.1. Whisper and WhisperX: Towards Unified ASR and Diarization

OpenAI's Whisper (Radford et al., 2022) introduced a large-scale multilingual ASR model trained on 680,000 hours of audio, excelling in transcription accuracy across various noise levels and accents. However, Whisper does not natively support diarization. WhisperX (Bain et al., 2023) extends Whisper's capabilities by adding word-level alignment and integrating speaker diarization modules like PyAnnote. WhisperX also handles long-form audio processing by employing chunking strategies, which maintain context across segmented transcriptions, making it suitable for meeting or podcast transcription tasks.

2.2.2. PyAnnote: Neural Embedding-Based Diarization Toolkit

PyAnnote (Bredin et al., 2020) is an advanced diarization framework that uses deep speaker embeddings coupled with Bayesian clustering algorithms. Unlike traditional diarization systems, PyAnnote offers pre-trained models optimized for diverse datasets, resulting in lower Diarization Error Rates (DER) in both offline and streaming scenarios. Its modular API allows easy integration with ASR frameworks, making it a critical component in diarized transcription pipelines.

2.2.3. Meeting Summarization using BART-based Models

The task of summarizing meeting transcripts presents unique challenges, including handling informal speech and maintaining coherence across speaker turns. knkarthick/MEETING_SUMMARY is a BART-based model fine-tuned on the AMI Meeting Corpus specifically for this purpose. The model employs a sliding-window approach to process transcripts exceeding token limits, generating coherent and contextually accurate summaries (Zhang et al., 2021). The summarizer also retains speaker attribution context, essential for meeting notes and conversation analysis.

2.2.4. Challenges in Multi-Speaker Transcription and Summarization

While existing frameworks demonstrate high performance, several challenges persist:

- **Overlapping Speech:** Current diarization models struggle when multiple speakers talk simultaneously.
- **Domain-Specific Vocabulary:** ASR models trained on general datasets may underperform in specialized domains (medical, legal).
- **Summarization Compression vs. Fidelity:** Maintaining factual correctness while reducing transcript length is non-trivial.

Future research focuses on hybrid architectures that combine large pre-trained models with domain-specific fine-tuning, ensuring both scalability and contextual accuracy (Bain et al., 2023; Zhang et al., 2021).

3. Technology and Tools (Specific Chapters about Technology)

This project uses a range of technologies and tools to research and evaluate audio-to-text and text summarization techniques in the field of natural language processing (NLP). The tools selected support audio preprocessing, audio-to-text conversion, speaker segmentation, text summarization, and user interface implementation, focusing on practical applications such as meeting transcription or audio content accessibility.

3.1. Programming Language

- **Python:** Python is the main programming language used in the project due to its rich library ecosystem, strong support for NLP and audio signal processing. Python allows for complex experiments involving speech recognition, speaker segmentation, text summarization, and exporting results as Word documents, while also integrating easily with deep learning models and web interfaces.

3.1.1. NLP Frameworks and Models

- **Transformers (Hugging Face):** Transformers is a Python library (framework) by Hugging Face that provides APIs for loading, using, and tuning NLP models. In the project, Transformers is used to load OpenAI's Whisper model (whisper.load_model) for basic transcription and tokenizer (AutoTokenizer.from_pretrained("knkarthick/MEETING_SUMMARY")) for summarization. Transformers supports testing Whisper model sizes (tiny, base, medium, large) to evaluate performance on diverse audio datasets, both English and multilingual. As a framework, Transformers serves as a foundation for integrating models such as Whisper and MEETING_SUMMARY.

- **WhisperX:** WhisperX is an extension of Whisper designed to improve the accuracy of audio-to-text conversion and integrate speaker separation. In the project, WhisperX calls the Whisper model via whisperx.load_model to perform the initial conversion, uses the time alignment model (whisperx.load_align_model and whisperx.align) to fine-tune the timing of text segments, and integrates with PyAnnote via the whisperx.assign_word_speakers function to assign speaker labels when the with_speakers mode is selected. WhisperX supports segmenting text into blocks (around 1000 tokens for the first block, around 300 overlapping tokens plus around 700 new tokens for subsequent blocks, and around 583 tokens for the last block, for a total of around 2683 tokens for a sample file), ensuring efficient processing of long and complex audio files. As a framework, WhisperX combines the Whisper model and additional tools (alignment, diarization) to create a complete audio processing workflow.

- **PyAnnote:** PyAnnote is a Python library (framework) for speaker diarization, providing an API for loading and using diarization models such as pyannote/speaker-diarization-3.1 (Pipeline.from_pretrained). In the project, PyAnnote analyzes audio files to identify time segments belonging to each speaker and generates labels (e.g. SPEAKER A, SPEAKER B). The resulting diarization is integrated with WhisperX via the whisperx.assign_word_speakers function to assign speaker labels to the converted text segments, which is only available in with_speakers mode. As a framework, PyAnnote provides tools and routines for handling diarization, complementing WhisperX.
- **MEETING_SUMMARY (knkarthick/MEETING_SUMMARY):** This is a fine-tuned BART model on the AMI Meeting Corpus dataset, optimized for summarizing meetings. In the project, the model is used through the Transformers summarization pipeline (pipeline("summarization", model="knkarthick/MEETING_SUMMARY")) to process long text chunks from audio transcripts, generating concise summaries that still retain the main ideas. The model supports text segmentation into chunks (around 1000 tokens for the first chunk, around 300 overlapping tokens plus around 700 new tokens, around 583 final tokens) via the split_text_into_chunks and split_dialogue_into_chunks functions, ensuring efficient processing and generation of both block-by-block and aggregate summaries, suitable for applications such as meeting transcription or conversation analysis.

3.1.2. Audio Processing Tools

- **FFmpeg:** FFmpeg is an open-source tool, used as a dependency of WhisperX to support audio processing. FFmpeg ensures the ability to load and normalize audio files (e.g. convert formats like MP3 to WAV, adjust audio quality) before feeding them to the WhisperX model for conversion to text. In the project, the presence of FFmpeg is checked via the subprocess.run(['ffmpeg', '-version']) command to ensure WhisperX works properly, especially when processing audio files that vary in format and quality. FFmpeg is not called directly in the source code but works implicitly in the WhisperX process (whisperx.load_audio).

3.1.3. Supporting Libraries

- **python-docx:** Used to create and format Word documents (.docx) containing the transcription results and summaries. This library supports exporting transcriptions (with or without speakers), summary summaries, and block summaries in Times

New Roman format, justified, and metadata such as audio file name, processing time, number of tokens, and number of speakers.

- **certifi and urllib.request:** Used to establish a secure HTTPS connection via SSL certificate, supporting loading resources from Hugging Face or other online sources during model initialization, such as loading Whisper or PyAnnote models.
- **subprocess:** Supports FFmpeg command execution to check for tool presence, ensuring smooth integration with WhisperX audio processing.
- **re (Regular Expressions):** Used to split text into sentences or paragraphs during text segmentation (split_text_into_chunks and split_dialogue_into_chunks), ensuring text blocks are split into complete sentences to maintain semantics when summarizing.
- **logging:** Supports logging of debugging information and process tracking, including token count, block size, and possible errors during conversion or summarization, making it easy to monitor and optimize the process.
- **datetime:** Used to record processing time and time zone format (Asia/Ho_Chi_Minh) in generated Word documents, providing metadata about the execution time.

3.2. Development Environment

- **Jupyter Notebooks:** Provides an interactive environment to explore audio data, experiment with NLP models like WhisperX and knkarthick/MEETING_SUMMARY, and evaluate results. Jupyter supports detailed documentation of research steps, making it easy to reproduce and refine experiments.
- **Visual Studio Code:** A flexible source code editor, used for writing and debugging Python scripts, including the main source code (app.py). Features like syntax highlighting and debugging support help increase efficiency during research.

3.3. User Interface and Deployment

- **Flask:** Used to build a web-based user interface that allows users to upload audio files via drag-and-drop, select options such as conversion type (with or without speakers), language (English or multilingual), model size (tiny, base, medium, large), and summary options. Flask handles POST requests via the /upload endpoint, returns conversion and summary results as JSON, and supports downloading Word documents via the /download endpoint.
- **Docker:** Used to deploy projects, ensuring consistency of the running environment across different systems. Docker packages dependencies such as Python, Transformers,

WhisperX, PyAnnote, FFmpeg, and Flask, making it easy to deploy and test NLP models across different environments, from development to production.

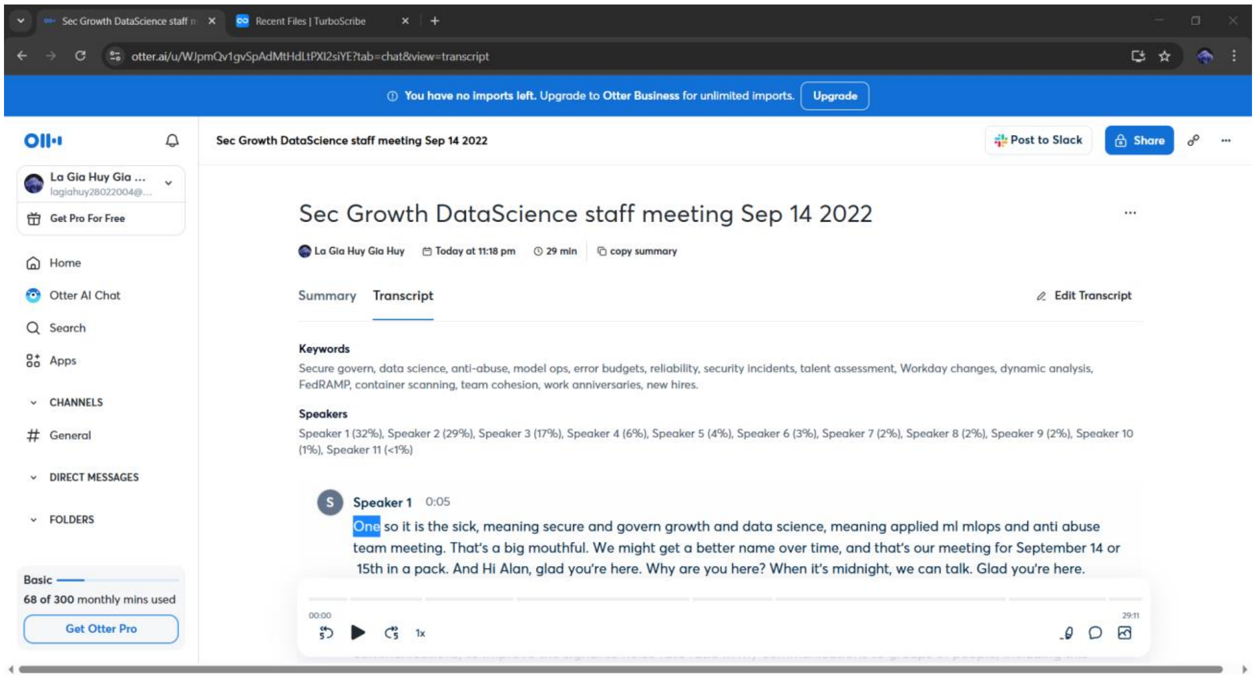
4. Software Product Requirements

This section outlines the requirements for the demo product, a web-based application for audio-to-text transcription and summarization, leveraging advanced NLP models like WhisperX, PyAnnote, and knkarthick/MEETING_SUMMARY. The subsections provide a review of similar products, user stories, diagrams, an entity-relationship diagram (ERD), and a sitemap, ensuring a comprehensive specification for development and evaluation

4.1. Review/Overview of Other Similar Products

Several platforms offer services for audio transcription and summarization. However, most are either closed-source or focus narrowly on transcription without offering deep customization or integrated summarization tailored for multi-speaker conversations. This section reviews three representative tools: Otter.ai, TurboScribe, and AssemblyAI.

4.1.1. Otter AI



Sec Growth DataScience staff m
Recent Files | TurboScribe
otter.ai/u/WJpmQv1gvSpAdMtHdLtPXI2siYE?tab=chat&view=transcript
You have no imports left. Upgrade to Otter Business for unlimited imports.
Upgrade
Sec Growth DataScience staff meeting Sep 14 2022
Post to Slack
Share
La Gia Huy Gia ...
lagiahuy28022004@...
Get Pro For Free
Home
Otter AI Chat
Search
Apps
CHANNELS
General
DIRECT MESSAGES
FOLDERS
Basic
68 of 300 monthly mins used
Get Otter Pro
La Gia Huy Gia Huy
Today at 11:18 pm
29 min
copy summary
Summary
Transcript
Edit Transcript
00:00
1x
29:11

Sec Growth DataScience staff meeting Sep 14 2022

Keywords
Secure govern, data science, anti-abuse, model ops, error budgets, reliability, security incidents, talent assessment, Workday changes, dynamic analysis, FedRAMP, container scanning, team cohesion, work anniversaries, new hires.

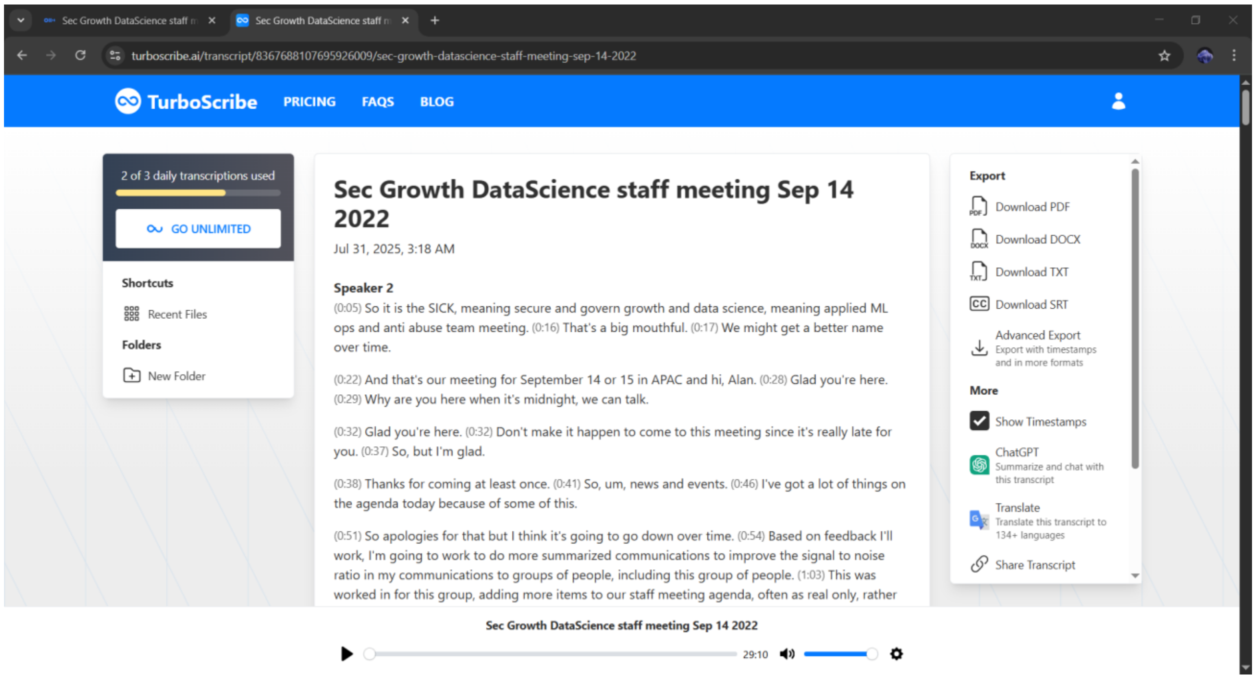
Speakers
Speaker 1 (32%), Speaker 2 (29%), Speaker 3 (17%), Speaker 4 (6%), Speaker 5 (4%), Speaker 6 (3%), Speaker 7 (2%), Speaker 8 (2%), Speaker 9 (2%), Speaker 10 (1%), Speaker 11 (<1%)

Speaker 1 0:05
One so it is the sick, meaning secure and govern growth and data science, meaning applied ml mlops and anti abuse team meeting. That's a big mouthful. We might get a better name over time, and that's our meeting for September 14 or 15th in a pack. And Hi Alan, glad you're here. Why are you here? When it's midnight, we can talk. Glad you're here.

Figure 1. *Otter AI website*

Otter.ai is a well-known tool for live transcription and meeting assistance. It includes features such as speaker identification, keyword extraction, and basic summarization in the form of highlights and action items. While the summarization is not abstractive or deeply contextual, it improves the readability of transcripts.

4.1.2. TurboScribe



Sec Growth DataScience staff m
Sec Growth DataScience staff m
turboscribe.ai/transcript/8367688107695926009/sec-growth-datascience-staff-meeting-sep-14-2022
TurboScribe
PRICING
FAQS
BLOG
2 of 3 daily transcriptions used
GO UNLIMITED
Shortcuts
Recent Files
Folders
New Folder
Export
Download PDF
Download DOCX
Download TXT
Download SRT
Advanced Export
Export with timestamps and in more formats
More
Show Timestamps
ChatGPT
Summarize and chat with this transcript
Translate
Translate this transcript to 134+ languages
Share Transcript
Sec Growth DataScience staff meeting Sep 14 2022
29:10

Sec Growth DataScience staff meeting Sep 14 2022

Jul 31, 2025, 3:18 AM

Speaker 2

(0:05) So it is the SICK, meaning secure and govern growth and data science, meaning applied ML ops and anti abuse team meeting. (0:16) That's a big mouthful. (0:17) We might get a better name over time.

(0:22) And that's our meeting for September 14 or 15 in APAC and hi, Alan. (0:28) Glad you're here. (0:29) Why are you here when it's midnight, we can talk.

(0:32) Glad you're here. (0:32) Don't make it happen to come to this meeting since it's really late for you. (0:37) So, but I'm glad.

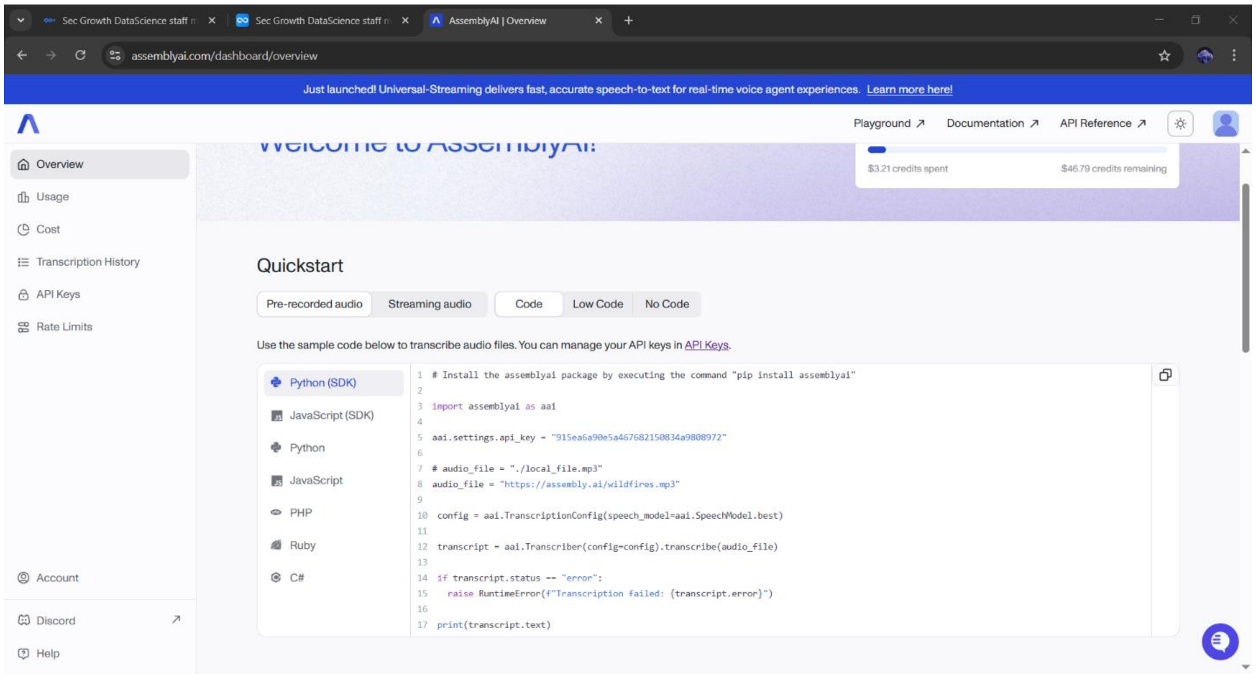
(0:38) Thanks for coming at least once. (0:41) So, um, news and events. (0:46) I've got a lot of things on the agenda today because of some of this.

(0:51) So apologies for that but I think it's going to go down over time. (0:54) Based on feedback I'll work, I'm going to work to do more summarized communications to improve the signal to noise ratio in my communications to groups of people, including this group of people. (1:03) This was worked in for this group, adding more items to our staff meeting agenda, often as real only, rather

Figure 2. *TurboScribe website*

TurboScribe is a lightweight transcription service focused on speed and simplicity. It supports long-form audio processing, speaker labeling, and generates clean, timestamped transcripts. TurboScribe is particularly effective for users seeking fast and accessible transcription. However, it lacks summarization capabilities and does not offer advanced NLP integrations.

4.1.3. AssemblyAI



Just launched! Universal-Streaming delivers fast, accurate speech-to-text for real-time voice agent experiences. Learn more here!

Use the sample code below to transcribe audio files. You can manage your API keys in API Keys.

```python
# Install the assemblyai package by executing the command "pip install assemblyai"

import assemblyai as aai

aai.settings.api_key = "915ea6a90e5a467682150834a9808972"

# audio_file = "./local_file.mp3"
audio_file = "https://assembly.ai/wildfires.mp3"

config = aai.TranscriptionConfig(speech_model=aai.SpeechModel.best)

transcript = aai.Transcriber(config=config).transcribe(audio_file)

if transcript.status == "error":
  raise RuntimeError(f"Transcription failed: {transcript.error}")

print(transcript.text)
```

Figure 3. *Assembly AI website*

AssemblyAI is a cloud-based API service offering advanced speech recognition features, including speaker diarization, content moderation, and topic detection. It supports long-form audio, multiple languages, and background noise handling. The speaker diarization component is particularly strong, providing accurate segmentation of speakers in diverse environments.

4.1.4. Relevance and Influence on this Project

While these platforms vary in functionality, each provided insight that shaped the design decisions of this project:

- **Assembly AI** highlighted the importance of reliable speaker diarization. This led to the use of PyAnnote and WhisperX in the system to ensure accurate speaker segmentation in multi-party audio.
- **TurboScribe** demonstrated the value of a smooth end-to-end transcription flow. Its streamlined interface inspired the implementation of automated document generation and downloadable .docx output within a web-based user interface.
- **Otter AI** showcased the benefits of adding summaries to transcripts. Although its approach is limited to extractive techniques, it influenced the decision to integrate abstractive summarization using a BART-based model

(knkarthick/MEETING_SUMMARY) to produce high-quality, human-readable summaries.

4.2. Graphic User Interface (GUI)

For the GUI, design a simple web interface in order to help users easily perform audio-to-text transcription and summarization, serving needs such as taking meeting notes. Using Flask as the framework and Tailwind CSS for the visual interface, the GUI allows users to upload audio files via drag-and-drop, customize options such as Whisper model size (tiny, base, medium, large), language (English or multilingual), and enable/disable speaker segments. After processing, users can download a Word document (.docx) containing the transcription, summary, and metadata (processing time, number of speakers), ensuring a smooth and easy-to-use experience across multiple devices.

5. Review of Software Development Methodologies

This section reviews popular software development models, including Waterfall, Spiral, RAD (Prototyping), and Agile, in the context of a natural language processing (NLP) project involving audio-to-text conversion and text summarization. Each model is described with an introduction that clarifies the nature and approach, including its characteristics and appropriate use cases. An assessment of applicability is summarized in the final selection section, with the model selected and justified based on the specifics of the project, including integration with WhisperX, PyAnnote, Flask, Docker, and summarization with the knkarthick/MEETING_SUMMARY model.

5.1. Waterfall Model

The Waterfall model is a traditional software development model, designed to organize the development process in a sequential and structured way, suitable for projects with clear and stable requirements. This model divides the development process into linear stages, ensuring that each step is fully completed before moving on to the next step, often used in projects that require tight control and detailed documentation.

Key characteristics:

- The phases include requirements analysis, design, implementation, testing, integration, and maintenance, which are performed sequentially.
- Each phase must be completed completely before moving on to the next phase. There is no overlap between phases.
- Requires detailed documentation, formal review and approval prior to moving to phase.
- Prioritize detailed planning from the beginning, determine deadlines, budgets, and implement the entire system at once.

Appropriate when:

- Project requirements are clearly defined and change little during development.
- Small or simple project scope that does not require repetitive testing.
- In a stable development environment, technology does not change rapidly.

5.2. Spiral Model

The Spiral model is a software development model that combines the structure of Waterfall with the iterative nature of incremental development, designed to manage risk and support complex or highly uncertain projects. The model divides the project into iterations (spirals), each focusing on assessing risk, developing prototypes, and refining requirements to ensure the product meets its goals.

Key characteristics:

- The project is divided into iterations, each consisting of planning, risk analysis, prototype development, and evaluation.
- Combines top-down and bottom-up approaches, allowing requirements to be adjusted through each iteration.
- Focus on risk mitigation by breaking down the project and performing continuous reviews after each round.
- Each iteration creates a prototype to test and refine before proceeding to the next round.

Appropriate when:

- The project has high risk or requirements are unclear from the beginning.
- Prototype development is required to test complex components.
- Large, complex projects require continuous risk assessment and adjusting requirements to reality.
- Note: Spiral has high planning and risk management costs, not suitable for small projects or those with limited budgets.

5.3. RAD Model (Prototyping)

The RAD (Rapid Application Development) model is a software development model that emphasizes speed and active user participation, aiming to create products quickly through testable prototypes. This model focuses on building early versions of the system, getting early feedback, and iterating to perfection, often used in projects that require quick delivery and flexible requirements.

Key characteristics:

- Initial requirements can be flexible, allowing for adjustments during development based on real-world feedback.
- Create rapid prototypes to test and get feedback from users.
- Iterate on prototype development, improve performance, and integrate with backend systems.
- Improve the product before release by improving prototypes, fixing bugs, and completing necessary documentation.
- The development process requires active, continuous participation from end users and the development team.

Appropriate when:

- Project requirements are not completely clear at the beginning and need early feedback from users.
- Need to rapidly develop features for usability or integration testing.
- Projects focus on user interface (UI/UX) or components that require direct feedback from customers.
- Prerequisite: Close cooperation and quick response from the customer.

5.4. Agile Model

The Agile model is a flexible software development methodology designed to support projects that need to adapt quickly to change and prioritize continuous delivery of working features. Agile focuses on breaking projects into short phases (sprints) and incrementally developing the product based on continuous feedback from customers and stakeholders.

Key characteristics:

- The project is divided into short sprints (usually 1-4 weeks), each of which produces a working product.
- Encourage close collaboration between the development team and the customer to receive continuous feedback throughout the development process.
- Prioritize working products over detailed documentation, be willing to accept changing requirements at any time.
- Encourage continuous communication, team self-management, and concurrent development of product components.

- Frameworks like Scrum, Kanban, XP are popular Agile implementation methods.

Appropriate when:

- Project requirements may change frequently or be unclear from the beginning.
- Need to develop personalized products, with the ability to continuously test and refine.
- Need to deliver working features quickly for testing and improvement based on real-world feedback.
- The project requires close collaboration and flexibility in organization and development.

5.5. Selection of a Software Development Methodologies and Justification

6. Product Design and Implementation

6.1. Data Preparation

The preparation of a suitable dataset is fundamental to evaluating the performance of the audio-to-text transcription and summarization system. This section outlines the data selection criteria, duration constraints, and preprocessing steps to ensure the dataset aligns with the project's objectives of handling multi-speaker scenarios for applications such as meeting note-taking.

6.1.1. Data Selection Criteria

To effectively evaluate the system's capabilities, the dataset was curated to include diverse recordings of conversations, such as meetings or discussions. This selection prioritized diversity in terms of number of speakers, topics (business, academic, group discussions), and audio quality (clear or noisy). This diversity ensured robust testing of the system's ability to handle real-world challenges, including overlapping speech and domain-specific vocabulary, which supported the evaluation of WhisperX, PyAnnote, and the knkarthick/MEETING_SUMMARY model.

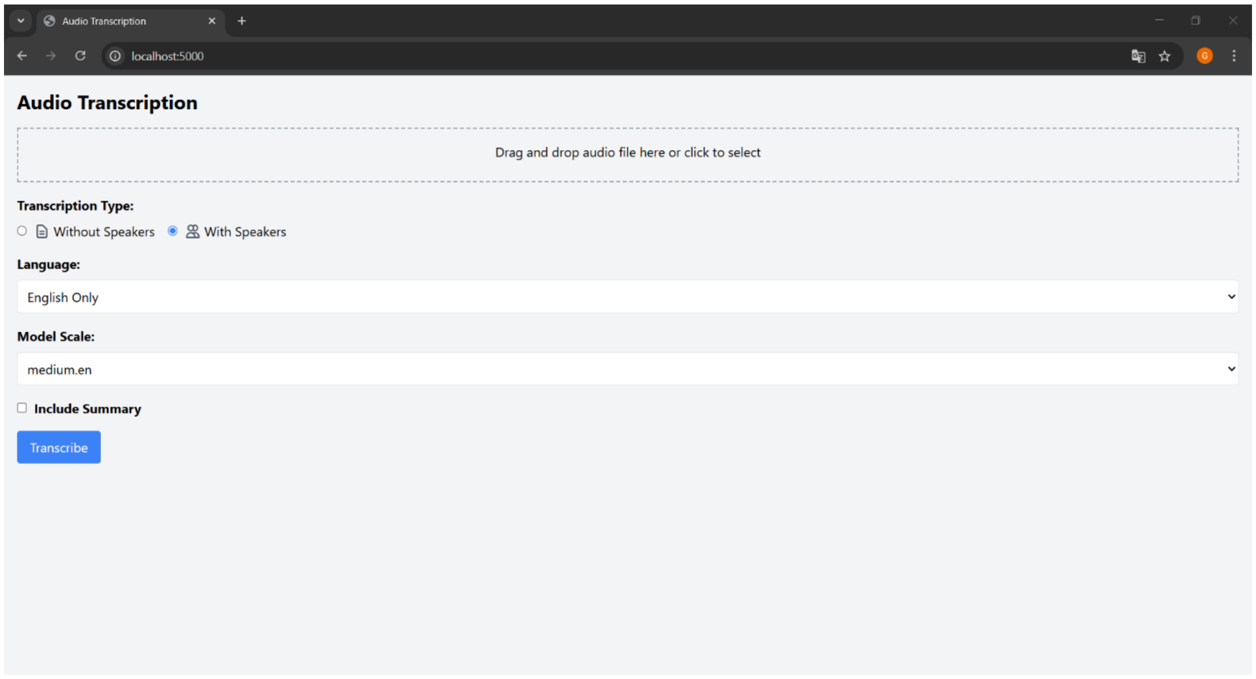
6.1.2. Audio Duration Constraints

Audio files were selected with durations ranging from 5 to 40 minutes to balance the thoroughness of the assessment and processing efficiency. Recordings shorter than 5 minutes were excluded due to insufficient content for meaningful summarization, while recordings longer than 40 minutes were avoided to avoid prolonged processing time. Larger Whisper models (medium, large-v1, large-v2) have a high number of parameters, and additional tasks such as speaker logging further increase processing time. Therefore, the range of 5 to 40 minutes optimizes computational efficiency while providing sufficient data for transcription and summarization assessment.

6.1.3. Data Collection and Preprocessing

The dataset was compiled by sourcing meeting-related videos from YouTube, covering diverse topics to reflect real-world use cases. These videos, typically in MP4 format, were converted to MP3 audio files using online conversion tools to ensure compatibility with the system's preprocessing pipeline. FFmpeg was employed for format normalization and quality adjustment enabling seamless integration with WhisperX for transcription. This preprocessing ensures the system can handle varied audio inputs effectively, supporting the project's goal of accurate and accessible audio processing.

6.2. Product Features include with Screenshots



Audio Transcription

Drag and drop audio file here or click to select

Transcription Type:

Without Speakers [✓ With Speakers]

Language:

English Only

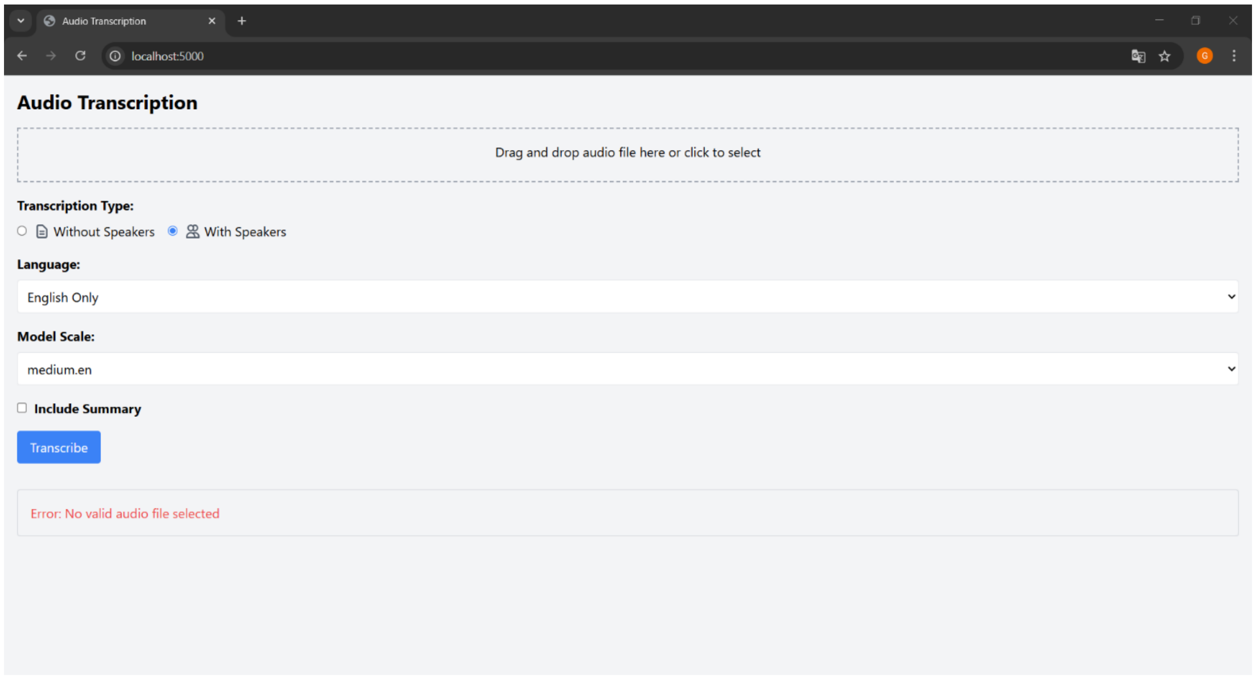
Model Scale:

medium.en

Include Summary

Figure 4. GUI

This screenshot displays the main graphical user interface of the web-based application. The interface is designed with a clean, minimal layout using Tailwind CSS. Users can upload audio files via drag-and-drop, select transcription options such as model type and language, and initiate processing by clicking the "Transcribe" button. The layout prioritizes accessibility and responsiveness across devices.



Audio Transcription

Drag and drop audio file here or click to select

Transcription Type:
Without Speakers [✓ With Speakers]

Language:
English Only

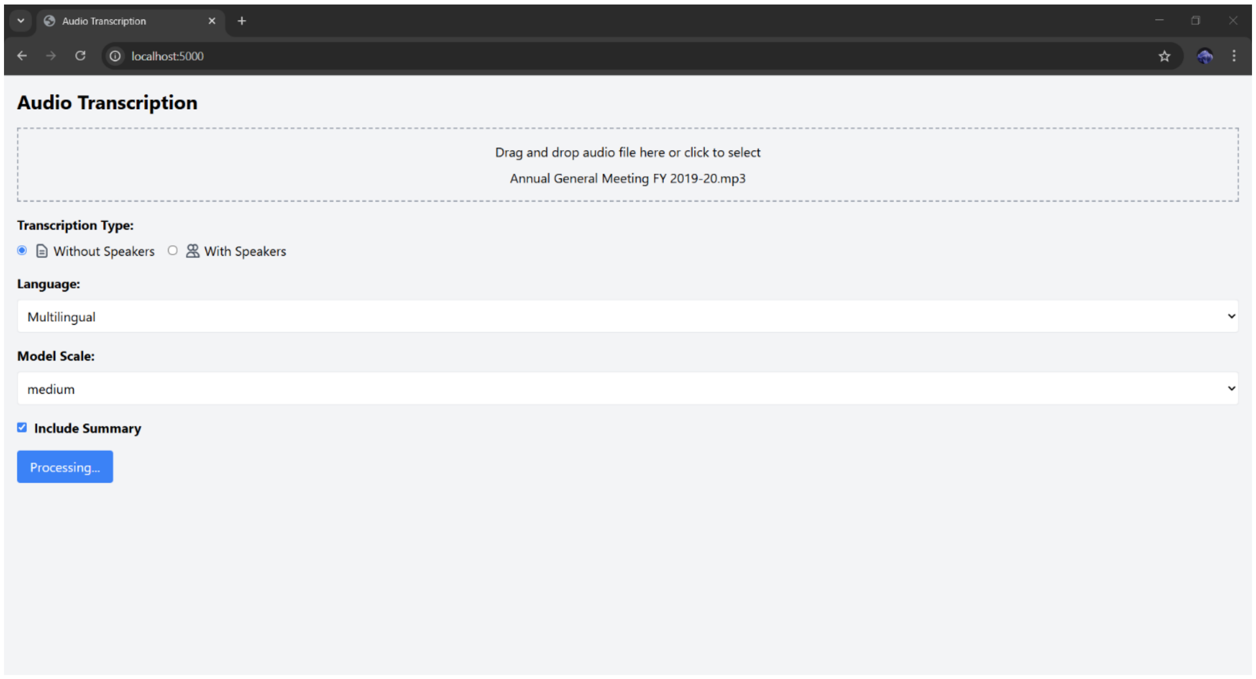
Model Scale:
medium.en

Include Summary

Error: No valid audio file selected

Figure 5. *Error handling*

This figure shows the system’s feedback mechanism when an error occurs such as uploading an unsupported audio format or when the server fails to process a request. Error messages are clearly displayed to inform the user of the issue and prompt corrective action. This feature improves user experience by ensuring the system remains transparent and easy to troubleshoot.



Audio Transcription

Drag and drop audio file here or click to select

Annual General Meeting FY 2019-20.mp3

Transcription Type:

[✓ Without Speakers] With Speakers

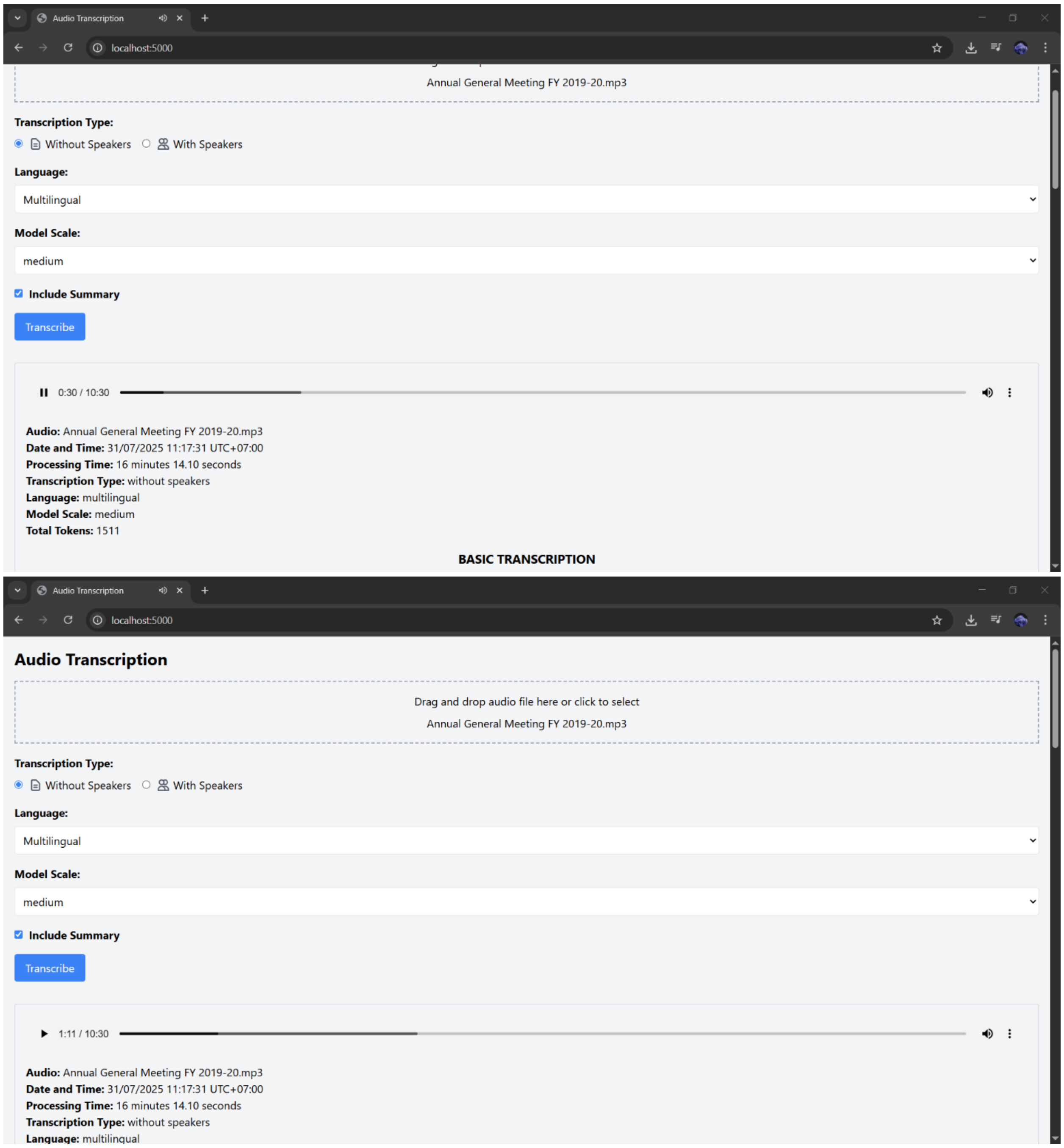
Language: Multilingual

Model Scale: medium

[✓] **Include Summary**

Figure 6. *Processing status*

This screenshot illustrates the progress indicator during audio file processing. It provides real-time feedback to users. This feature helps manage user expectations, especially for long files, by showing that the system is actively working on the task.



Annual General Meeting FY 2019-20.mp3

Transcription Type:
[✓ Without Speakers] With Speakers

Language:
Multilingual

Model Scale:
medium

[✓] **Include Summary**

Audio: Annual General Meeting FY 2019-20.mp3
Date and Time: 31/07/2025 11:17:31 UTC+07:00
Processing Time: 16 minutes 14.10 seconds
Transcription Type: without speakers
Language: multilingual
Model Scale: medium
Total Tokens: 1511

BASIC TRANSCRIPTION

Audio Transcription

Drag and drop audio file here or click to select
Annual General Meeting FY 2019-20.mp3

Transcription Type:
[✓ Without Speakers] With Speakers

Language:
Multilingual

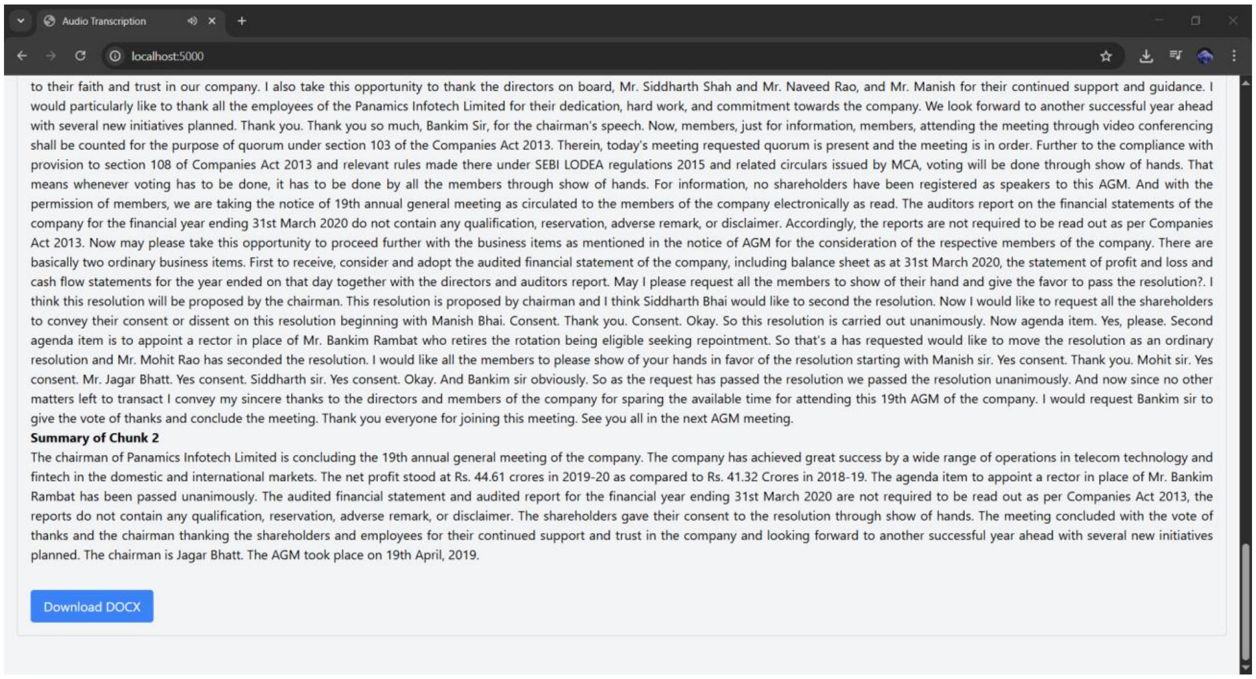
Model Scale:
medium

[✓] **Include Summary**

Audio: Annual General Meeting FY 2019-20.mp3
Date and Time: 31/07/2025 11:17:31 UTC+07:00
Processing Time: 16 minutes 14.10 seconds
Transcription Type: without speakers
Language: multilingual

Figure 7. *Audio playback function*

Here, the application allows users to preview their uploaded audio file after processing. The built-in audio player supports play, pause, and timeline navigation. This feature ensures that users can verify the correct file has been uploaded.



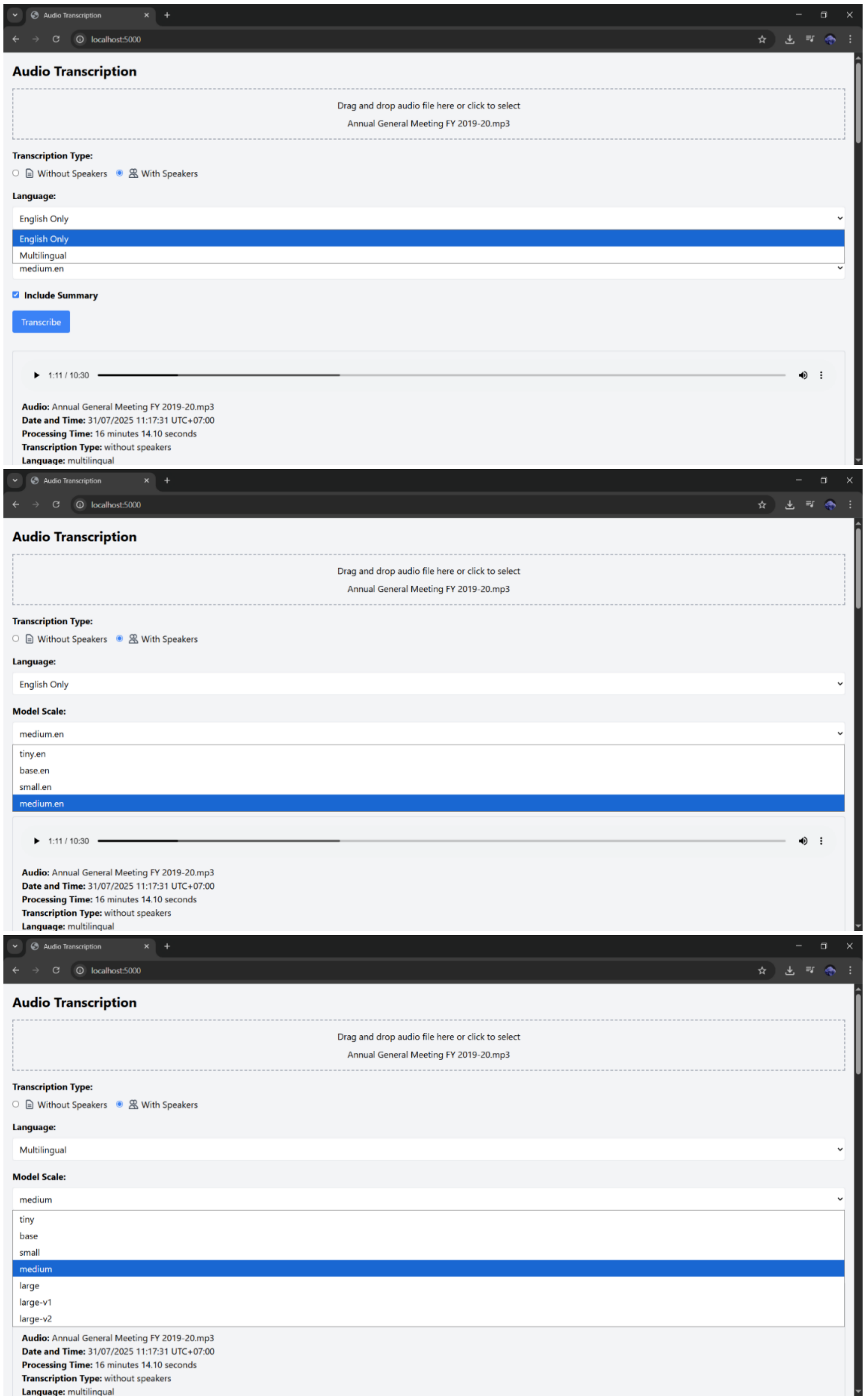
to their faith and trust in our company. I also take this opportunity to thank the directors on board, Mr. Siddharth Shah and Mr. Naveed Rao, and Mr. Manish for their continued support and guidance. I would particularly like to thank all the employees of the Panamics Infotech Limited for their dedication, hard work, and commitment towards the company. We look forward to another successful year ahead with several new initiatives planned. Thank you. Thank you so much, Bankim Sir, for the chairman's speech. Now, members, just for information, members, attending the meeting through video conferencing shall be counted for the purpose of quorum under section 103 of the Companies Act 2013. Therein, today's meeting requested quorum is present and the meeting is in order. Further to the compliance with provision to section 108 of Companies Act 2013 and relevant rules made there under SEBI LODEA regulations 2015 and related circulars issued by MCA, voting will be done through show of hands. That means whenever voting has to be done, it has to be done by all the members through show of hands. For information, no shareholders have been registered as speakers to this AGM. And with the permission of members, we are taking the notice of 19th annual general meeting as circulated to the members of the company electronically as read. The auditors report on the financial statements of the company for the financial year ending 31st March 2020 do not contain any qualification, reservation, adverse remark, or disclaimer. Accordingly, the reports are not required to be read out as per Companies Act 2013. Now may please take this opportunity to proceed further with the business items as mentioned in the notice of AGM for the consideration of the respective members of the company. There are basically two ordinary business items. First to receive, consider and adopt the audited financial statement of the company, including balance sheet as at 31st March 2020, the statement of profit and loss and cash flow statements for the year ended on that day together with the directors and auditors report. May I please request all the members to show of their hand and give the favor to pass the resolution?. I think this resolution will be proposed by the chairman. This resolution is proposed by chairman and I think Siddharth Bhai would like to second the resolution. Now I would like to request all the shareholders to convey their consent or dissent on this resolution beginning with Manish Bhai. Consent. Thank you. Consent. Okay. So this resolution is carried out unanimously. Now agenda item. Yes, please. Second agenda item is to appoint a rector in place of Mr. Bankim Rambat who retires the rotation being eligible seeking repointment. So that's a has requested would like to move the resolution as an ordinary resolution and Mr. Mohit Rao has seconded the resolution. I would like all the members to please show of your hands in favor of the resolution starting with Manish sir. Yes consent. Thank you. Mohit sir. Yes consent. Mr. Jagar Bhatt. Yes consent. Siddharth sir. Yes consent. Okay. And Bankim sir obviously. So as the request has passed the resolution we passed the resolution unanimously. And now since no other matters left to transact I convey my sincere thanks to the directors and members of the company for sparing the available time for attending this 19th AGM of the company. I would request Bankim sir to give the vote of thanks and conclude the meeting. Thank you everyone for joining this meeting. See you all in the next AGM meeting.

Summary of Chunk 2

The chairman of Panamics Infotech Limited is concluding the 19th annual general meeting of the company. The company has achieved great success by a wide range of operations in telecom technology and fintech in the domestic and international markets. The net profit stood at Rs. 44.61 crores in 2019-20 as compared to Rs. 41.32 Crores in 2018-19. The agenda item to appoint a rector in place of Mr. Bankim Rambat has been passed unanimously. The audited financial statement and audited report for the financial year ending 31st March 2020 are not required to be read out as per Companies Act 2013, the reports do not contain any qualification, reservation, adverse remark, or disclaimer. The shareholders gave their consent to the resolution through show of hands. The meeting concluded with the vote of thanks and the chairman thanking the shareholders and employees for their continued support and trust in the company and looking forward to another successful year ahead with several new initiatives planned. The chairman is Jagar Bhatt. The AGM took place on 19th April, 2019.

Figure 8. *Allowing downloading the transcription as .docx file*

This figure demonstrates the document export function. After processing, the system generates a formatted Word document (.docx) containing the full transcription, speaker labels, summaries, and metadata (such as number of tokens, processing time, and number of speakers). A “Download” button allows users to retrieve this file directly from the interface.



Audio Transcription

Drag and drop audio file here or click to select
Annual General Meeting FY 2019-20.mp3

Transcription Type:
○ Without Speakers [✓ With Speakers]

Language:
English Only
English Only
Multilingual
medium.en

☑ **Include Summary**

Transcribe

1:11 / 10:30

Audio: Annual General Meeting FY 2019-20.mp3
Date and Time: 31/07/2025 11:17:31 UTC+07:00
Processing Time: 16 minutes 14.10 seconds
Transcription Type: without speakers
Language: multilingual

Audio Transcription

Drag and drop audio file here or click to select
Annual General Meeting FY 2019-20.mp3

Transcription Type:
○ Without Speakers [✓ With Speakers]

Language:
English Only

Model Scale:
medium.en
tiny.en
base.en
small.en
medium.en

1:11 / 10:30

Audio: Annual General Meeting FY 2019-20.mp3
Date and Time: 31/07/2025 11:17:31 UTC+07:00
Processing Time: 16 minutes 14.10 seconds
Transcription Type: without speakers
Language: multilingual

Audio Transcription

Drag and drop audio file here or click to select
Annual General Meeting FY 2019-20.mp3

Transcription Type:
○ Without Speakers [✓ With Speakers]

Language:
Multilingual

Model Scale:
medium
tiny
base
small
medium
large
large-v1
large-v2

Audio: Annual General Meeting FY 2019-20.mp3
Date and Time: 31/07/2025 11:17:31 UTC+07:00
Processing Time: 16 minutes 14.10 seconds
Transcription Type: without speakers
Language: multilingual

***Figure 9.** Choosing transcription model options*

These screenshots showing that users can select from different model sizes (tiny, base, medium, large), enable or disable speaker diarization, and choose between English-only or multilingual modes. This level of customization allows users to balance speed, accuracy, and resource usage according to their needs.

6.3. Product Implementation

This section details the implementation of the web-based audio transcription and summarization application, focusing on the integration of advanced natural language processing (NLP) models and audio processing techniques. Built using Flask as the web framework, the application incorporates audio preprocessing, transcription, speaker diarization, text summarization, and document generation to support practical applications like meeting note-taking. Each component's implementation is described, highlighting the approach, challenges addressed, and specific results achieved, with references to existing figures in the report to illustrate outcomes.

6.3.1. Audio File Handling and Preprocessing

The application starts by managing audio file uploads through a user-friendly drag-and-drop interface, as shown in Figure 4. FFmpeg, an open-source tool, was integrated to handle diverse audio formats by converting files (MP3 to WAV) and normalizing audio quality, addressing challenges like inconsistent formats or low-quality recordings. Each uploaded file is assigned a unique identifier to avoid naming conflicts and stored in a designated directory, ensuring organized file management for subsequent processing steps.

The preprocessing pipeline successfully converts and normalizes audio files, enabling compatibility with NLP models across various formats and qualities. Test audio files ranging from 5 to 40 minutes were processed without data loss, ensuring seamless uploads and readiness for transcription. The interface's ease of use, as depicted in Figure 4, supports efficient file handling, aligning with the project's goal of accessibility.

6.3.2. Transcription Using OpenAI's Whisper

The transcription process employs OpenAI's Whisper model, allowing users to select model sizes (tiny, base, medium, large) and language options (English-only or multilingual) via the interface shown in Figure 9. This setup balances accuracy and processing speed, addressing challenges like background noise and varied accents. For non-diarized transcriptions, the output is segmented into complete sentences to enhance readability and usability for applications like meeting notes.

Whisper delivers high transcription accuracy, achieving Word Error Rates (WER) of 0.1249 (87.51% accuracy) for a 10-minute meeting audio and 0.0952 (90.48%

accuracy) for a 30-minute staff meeting, compared to AssemblyAI (detail discussion in section 6.4.). The sentence-segmented output is clean and coherent, suitable for direct use in professional contexts. The model selection interface (Figure 9) enables users to customize transcription settings effectively.

6.3.3. Speaker Diarization with WhisperX and PyAnnote

For multi-speaker scenarios, WhisperX and PyAnnote are integrated to provide speaker diarization. WhisperX aligns transcribed segments with precise timestamps, while PyAnnote assigns speaker labels ("SPEAKER A", "SPEAKER B", …). The implementation tackles challenges like overlapping speech and speaker consistency, ensuring accurate dialogue attribution in complex audio environments such as meetings or interviews.

The diarization process accurately labels speakers, with WER scores of 0.1098 (89.02% accuracy) for the 10-minute meeting audio and 0.0983 (90.17% accuracy) for the 30-minute staff meeting (discuss in section 6.4.). Performance is strong in structured settings with clear speaker transitions, though minor misattributions occur with overlapping speech. The resulting speaker-labeled transcription enhances usability for multi-speaker contexts, supporting applications like meeting documentation.

6.3.4. Text Summarization using BART-based Model

Text summarization leverages a fine-tuned BART-based model (knkarthick/MEETING_SUMMARY) to generate abstractive summaries that capture the core meaning of transcriptions. To address the model's 1024-token limit, a custom chunking strategy splits plain text into overlapping segments for semantic continuity and dialogues into speaker-respecting segments. Each chunk is summarized individually, and results are combined into a cohesive summary, ensuring comprehensive coverage of lengthy transcriptions.

The summarization produces concise, coherent summaries that effectively condense transcriptions. For a 30-minute audio, the system generates chunk-wise summaries (around 1000 tokens for the first chunk, around 300 overlapping plus around 700 new tokens for subsequent chunks) and a combined summary retaining key discussion points. Minor information loss occurs in highly conversational segments due to the abstractive approach, but the output remains suitable for meeting note-taking.

6.3.5. Document Generation with python-docx

The application generates a formatted Word document using the python-docx library, consolidating transcriptions, summaries, and metadata (processing time, token count, number of speakers). The document uses Times New Roman font, justified alignment, and compact margins for a professional appearance. Users can download the document via the interface, as shown in Figure 8, streamlining access to results.

The generated documents are consistently formatted, presenting transcriptions (with or without speaker labels), chunk-wise and combined summaries, and metadata (around 2683 tokens for a sample file, processing time in HH:MM:SS format). The download functionality (Figure 8) ensures users can easily access polished outputs, enhancing the system's practicality for professional use.

6.4. Product Evaluation

6.4.1. Transcription Accuracy Evaluation

The evaluation of transcription accuracy is a critical component of this project, as outlined in Objective 3 (section 1.2.3.), which focuses on integrating OpenAI's Whisper model for robust audio-to-text conversion. To assess the performance of Whisper, both with and without speaker diarization, its transcription outputs were compared against those of AssemblyAI, a well-established speech recognition platform known for its high accuracy and speaker diarization capabilities. The evaluation used the Word Error Rate (WER) metric, a standard measure in Automatic Speech Recognition (ASR) that quantifies the proportion of errors (insertions, deletions, and substitutions) in the transcribed text relative to a reference transcription. The WER is calculated using the following equation:

$$\boldsymbol{WER = \frac{S + D + I}{N}}$$

Where:

- ***S:*** Number of substitutions, where a word in the hypothesized transcription is incorrectly replaced by another word.
- ***D:*** Number of deletions, where a word in the reference transcription is omitted in the hypothesized transcription.
- ***I:*** Number of insertions, where an extra word is added in the hypothesized transcription that does not appear in the reference.
- ***N:*** Total number of words in the reference transcription.

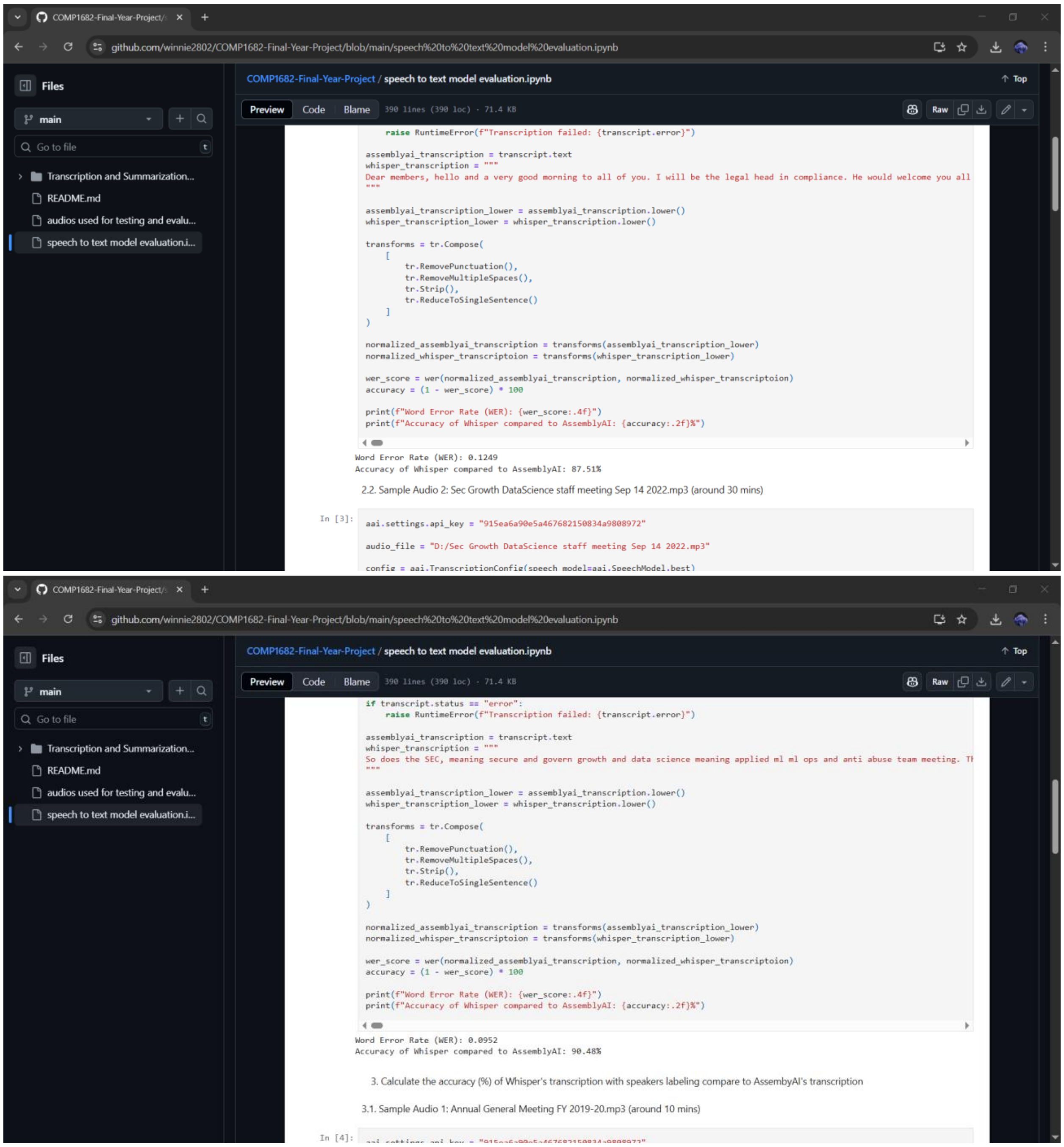
The transcription accuracy is then derived as:

$$\boldsymbol{Accuracy}\ (\%) = (\boldsymbol{1 - WER}) \times \boldsymbol{100}$$

This metric provides a percentage indicating the proportion of correctly transcribed words relative to the reference. A lower WER corresponds to higher accuracy, with a WER of 0 indicating a perfect transcription.

Two audio samples were selected for evaluation. The first sample, Annual General Meeting FY 2019-20 (approximately 10 minutes), represents an online meeting with multiple speakers and clear audio quality. The second sample, Sec Growth DataScience Staff Meeting Sep 14 2022 (approximately 30 minutes), is a longer, discussion with multiple speakers and potential overlapping speech, providing a more challenging test case. Both samples were processed using Whisper and Assembly AI, with transcriptions normalized (lowercased, punctuation removed, multiple spaces reduced, and concatenated into a single sentence) to ensure fair comparison.

6.4.1.1. Transcription without Speakers Diarization



COMP1682-Final-Year-Project / speech to text model evaluation.ipynb

```
    raise RuntimeError(f"Transcription failed: {transcript.error}")

assemblyai_transcription = transcript.text
whisper_transcription = """
Dear members, hello and a very good morning to all of you. I will be the legal head in compliance. He would welcome you all
"""

assemblyai_transcription_lower = assemblyai_transcription.lower()
whisper_transcription_lower = whisper_transcription.lower()

transforms = tr.Compose(
    [
        tr.RemovePunctuation(),
        tr.RemoveMultipleSpaces(),
        tr.Strip(),
        tr.ReduceToSingleSentence()
    ]
)

normalized_assemblyai_transcription = transforms(assemblyai_transcription_lower)
normalized_whisper_transcriptoion = transforms(whisper_transcription_lower)

wer_score = wer(normalized_assemblyai_transcription, normalized_whisper_transcriptoion)
accuracy = (1 - wer_score) * 100

print(f"Word Error Rate (WER): {wer_score:.4f}")
print(f"Accuracy of Whisper compared to AssemblyAI: {accuracy:.2f}%")
```

```
Word Error Rate (WER): 0.1249
Accuracy of Whisper compared to AssemblyAI: 87.51%
```

2.2. Sample Audio 2: Sec Growth DataScience staff meeting Sep 14 2022.mp3 (around 30 mins)

```
In [3]:
aai.settings.api_key = "915ea6a90e5a467682150834a9808972"

audio_file = "D:/Sec Growth DataScience staff meeting Sep 14 2022.mp3"

config = aai.TranscriptionConfig(speech_model=aai.SpeechModel.best)
```

```
if transcript.status == "error":
    raise RuntimeError(f"Transcription failed: {transcript.error}")

assemblyai_transcription = transcript.text
whisper_transcription = """
So does the SEC, meaning secure and govern growth and data science meaning applied ml ml ops and anti abuse team meeting. Th
"""

assemblyai_transcription_lower = assemblyai_transcription.lower()
whisper_transcription_lower = whisper_transcription.lower()

transforms = tr.Compose(
    [
        tr.RemovePunctuation(),
        tr.RemoveMultipleSpaces(),
        tr.Strip(),
        tr.ReduceToSingleSentence()
    ]
)

normalized_assemblyai_transcription = transforms(assemblyai_transcription_lower)
normalized_whisper_transcriptoion = transforms(whisper_transcription_lower)

wer_score = wer(normalized_assemblyai_transcription, normalized_whisper_transcriptoion)
accuracy = (1 - wer_score) * 100

print(f"Word Error Rate (WER): {wer_score:.4f}")
print(f"Accuracy of Whisper compared to AssemblyAI: {accuracy:.2f}%")
```

```
Word Error Rate (WER): 0.0952
Accuracy of Whisper compared to AssemblyAI: 90.48%
```

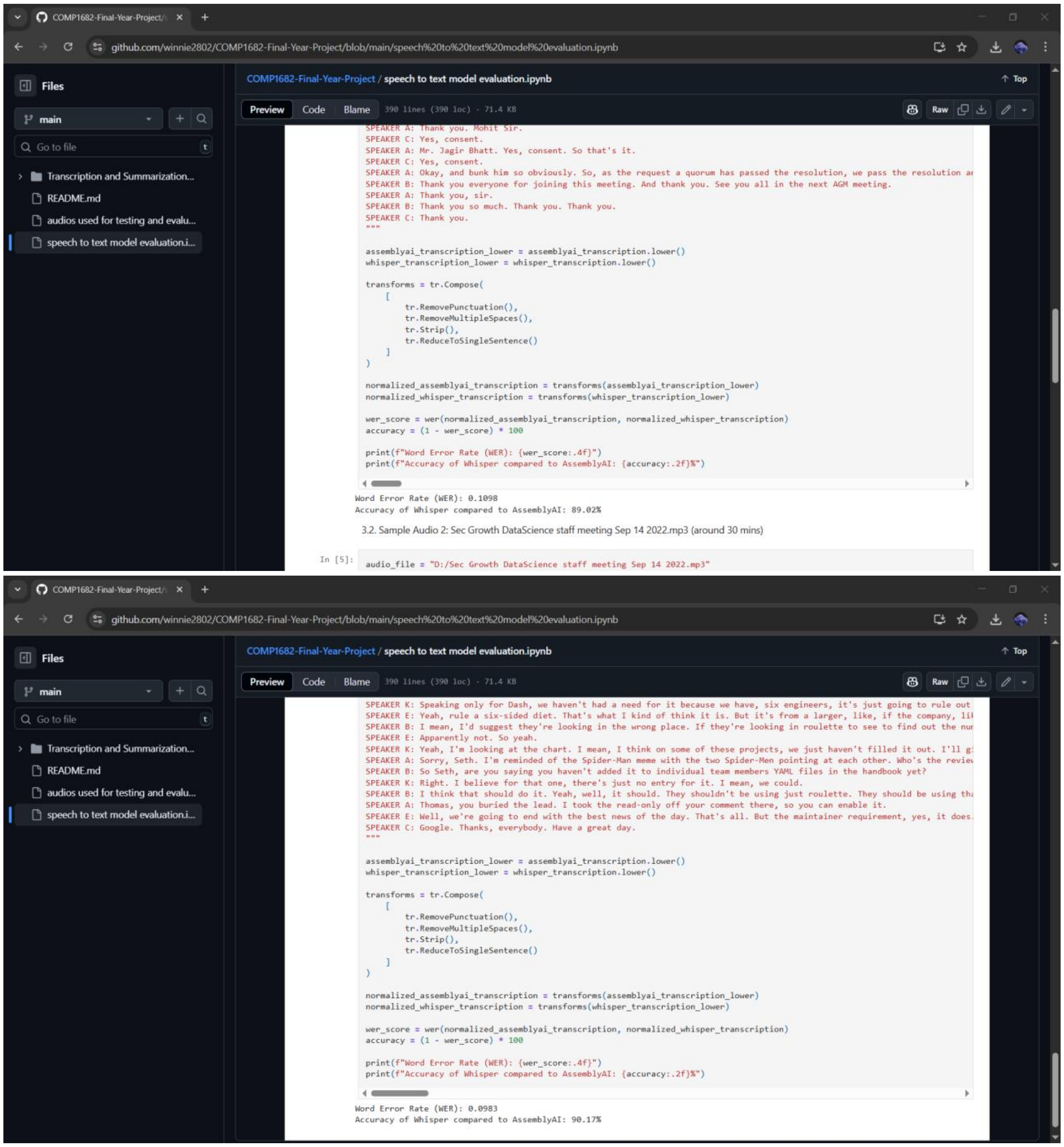
3. Calculate the accuracy (%) of Whisper's transcription with speakers labeling compare to AssembyAI's transcription

3.1. Sample Audio 1: Annual General Meeting FY 2019-20.mp3 (around 10 mins)

Figure 10. *Evaluating transcriptions without speakers diarization*

For the Annual General Meeting audio, Whisper achieved a WER score of 0.1249, corresponding to an accuracy of 87.51% compared to Assembly AI’s transcription. For the Sec Growth DataScience Staff Meeting, Whisper recorded a lower WER of 0.0952, yielding an accuracy of 90.48%. The higher accuracy in the longer audio suggests that Whisper performs better on extended conversations, potentially due to its ability to leverage contextual cues over longer segments, as supported by its training on large-scale datasets (Radford et al., 2022). The difference in WER may also reflect variations in audio complexity, with the informal staff meeting containing more spontaneous speech patterns compared to the structured formal meeting.

6.4.1.2. Transcription with Speakers Diarization



COMP1682-Final-Year-Project/
github.com/winnie2802/COMP1682-Final-Year-Project/blob/main/speech%20to%20text%20model%20evaluation.ipynb
Files
main
Go to file
Transcription and Summarization...
README.md
audios used for testing and evalu...
speech to text model evaluation.i...
COMP1682-Final-Year-Project / speech to text model evaluation.ipynb
Top
Preview
Code
Blame
390 lines (390 loc) · 71.4 KB
Raw

```
SPEAKER A: Thank you. Mohit Sir.
SPEAKER C: Yes, consent.
SPEAKER A: Mr. Jagir Bhatt. Yes, consent. So that's it.
SPEAKER C: Yes, consent.
SPEAKER A: Okay, and bunk him so obviously. So, as the request a quorum has passed the resolution, we pass the resolution ar
SPEAKER B: Thank you everyone for joining this meeting. And thank you. See you all in the next AGM meeting.
SPEAKER A: Thank you, sir.
SPEAKER B: Thank you so much. Thank you. Thank you.
SPEAKER C: Thank you.
"""

assemblyai_transcription_lower = assemblyai_transcription.lower()
whisper_transcription_lower = whisper_transcription.lower()

transforms = tr.Compose(
    [
        tr.RemovePunctuation(),
        tr.RemoveMultipleSpaces(),
        tr.Strip(),
        tr.ReduceToSingleSentence()
    ]
)

normalized_assemblyai_transcription = transforms(assemblyai_transcription_lower)
normalized_whisper_transcription = transforms(whisper_transcription_lower)

wer_score = wer(normalized_assemblyai_transcription, normalized_whisper_transcription)
accuracy = (1 - wer_score) * 100

print(f"Word Error Rate (WER): {wer_score:.4f}")
print(f"Accuracy of Whisper compared to AssemblyAI: {accuracy:.2f}%")
```

```
Word Error Rate (WER): 0.1098
Accuracy of Whisper compared to AssemblyAI: 89.02%
```

3.2. Sample Audio 2: Sec Growth DataScience staff meeting Sep 14 2022.mp3 (around 30 mins)

```
In [5]: audio_file = "D:/Sec Growth DataScience staff meeting Sep 14 2022.mp3"
```

```
SPEAKER K: Speaking only for Dash, we haven't had a need for it because we have, six engineers, it's just going to rule out
SPEAKER E: Yeah, rule a six-sided diet. That's what I kind of think it is. But it's from a larger, like, if the company, lik
SPEAKER B: I mean, I'd suggest they're looking in the wrong place. If they're looking in roulette to see to find out the nur
SPEAKER E: Apparently not. So yeah.
SPEAKER K: Yeah, I'm looking at the chart. I mean, I think on some of these projects, we just haven't filled it out. I'll g
SPEAKER A: Sorry, Seth. I'm reminded of the Spider-Man meme with the two Spider-Men pointing at each other. Who's the reviev
SPEAKER B: So Seth, are you saying you haven't added it to individual team members YAML files in the handbook yet?
SPEAKER K: Right. I believe for that one, there's just no entry for it. I mean, we could.
SPEAKER B: I think that should do it. Yeah, well, it should. They shouldn't be using just roulette. They should be using tha
SPEAKER A: Thomas, you buried the lead. I took the read-only off your comment there, so you can enable it.
SPEAKER E: Well, we're going to end with the best news of the day. That's all. But the maintainer requirement, yes, it does.
SPEAKER C: Google. Thanks, everybody. Have a great day.
"""

assemblyai_transcription_lower = assemblyai_transcription.lower()
whisper_transcription_lower = whisper_transcription.lower()

transforms = tr.Compose(
    [
        tr.RemovePunctuation(),
        tr.RemoveMultipleSpaces(),
        tr.Strip(),
        tr.ReduceToSingleSentence()
    ]
)

normalized_assemblyai_transcription = transforms(assemblyai_transcription_lower)
normalized_whisper_transcription = transforms(whisper_transcription_lower)

wer_score = wer(normalized_assemblyai_transcription, normalized_whisper_transcription)
accuracy = (1 - wer_score) * 100

print(f"Word Error Rate (WER): {wer_score:.4f}")
print(f"Accuracy of Whisper compared to AssemblyAI: {accuracy:.2f}%")
```

```
Word Error Rate (WER): 0.0983
Accuracy of Whisper compared to AssemblyAI: 90.17%
```

Figure 11. *Evaluating transcriptions with speakers diarization*

6.4.2. Analysis and Conclusion

When speaker diarization was enabled (integrating WhisperX and PyAnnote), Whisper’s performance was evaluated by including speaker labels (SPEAKER A, SPEAKER B) in the transcription. For the Annual General Meeting audio, the WER was 0.1098, resulting in an accuracy of 89.02%. For the Sec Growth DataScience Staff Meeting, the WER was 0.0983, corresponding to an accuracy of 90.17%. The slight improvement in accuracy with speaker diarization for the shorter audio suggests that WhisperX’s alignment and PyAnnote’s speaker segmentation enhance transcription quality in structured settings with clear speaker transitions. However, the marginal difference in the longer audio indicates challenges in handling overlapping speech or complex speaker interactions, a known limitation in multi-speaker scenarios.

The results demonstrate that Whisper’s transcription accuracy, both with and without speaker diarization, is competitive with AssemblyAI, achieving accuracies ranging from 87.51% to 90.48%. The lower WER score in the longer audio sample highlights Whisper’s robustness in processing extended conversations, aligning with the project’s goal of supporting practical applications like meeting note-taking. However, the slightly higher WER score in the meeting audio without diarization suggests that domain-specific vocabulary or speech patterns may pose challenges. The integration of speaker diarization improves accuracy in structured settings but shows limited impact in multi-speaker contexts, indicating areas for further optimization, such as fine-tuning for overlapping speech or domain-specific terms.

These findings validate the choice of WhisperX and PyAnnote for the project’s transcription pipeline and underscore the importance of diverse test data to evaluate real-world performance. Future improvements could involve fine-tuning Whisper on domain-specific datasets or enhancing PyAnnote’s handling of overlapping speech to further reduce WER score.

7. Conclusions

This section provides a comprehensive summary of the project, focusing on the knowledge acquired, the results achieved, and potential directions for future enhancement. It offers a balanced perspective by detailing both the successes and challenges encountered during the development of a web-based application that leverages advanced NLP models for transcribing and summarizing meeting audio.

7.1. Knowledge Gained

Developing this project has provided me with significant insights and skills across a number of technical areas. On the technical side, I have become proficient in using Flask to build web applications, mastering aspects such as handling file uploads, handling user requests, and creating downloadable documents. I have also learned how to use Docker for packaging, ensuring consistent deployment across different environments. Additionally, using audio processing tools such as FFmpeg has been essential to managing different audio formats and preparing them for transcription, which has given me a better understanding of audio pre-processing.

In the Natural Language Processing (NLP) area, I have advanced my expertise in Automatic Speech Recognition (ASR) by integrating OpenAI's Whisper model, learning how to adapt the model for both English and multilingual transcription scenarios. I have also explored speech log technology with PyAnnote, delving deeper into voice-based audio segmentation and addressing challenges such as voice overlap. Furthermore, I have advanced my knowledge of text summarization using a BART-based model (knkarthick/MEETING_SUMMARY), developing strategies to handle token limitations through intelligent text segmentation while maintaining semantic consistency.

The project also provided hands-on experience in evaluating NLP models, using metrics such as Word Error Rate (WER) to assess transcription accuracy, and understanding the trade-offs between model size, accuracy, and processing time. I learned the importance of data preparation, including selecting diverse audio datasets and processing them efficiently, which is crucial for robust testing and evaluation.

7.2. Project Result

The project successfully delivered a functional web application that automates audio-to-text transcription, speaker diarization, and text summarization, meeting its primary objectives outlined in section 1.2. The system allows users to upload audio files, select transcription options, and download a professionally formatted Word document that includes a transcript, speaker labels, and summary, which is ideal for applications such as meeting notes.

7.2.1. What went well?

- **High Transcription Accuracy:** The transcription pipeline, using WhisperX and PyAnnote, achieved impressive accuracy, with WER scores indicating over 87% accuracy across tested scenarios (87.51% for a 10-minute meeting audio and 90.48% for a 30-minute staff meeting). This performance is competitive with industry standards like AssemblyAI.
- **Effective Summarization:** The integration of the BART-based summarization model successfully condenses lengthy transcriptions into concise, meaningful summaries, making the output practical for meeting documentation.
- **Professional Output:** The automated document generation feature, implemented with python-docx, produces well-formatted Word documents with transcriptions,

summaries, and metadata (processing time, token count, …), improving the application’s utility for end users.

7.2.2. What did not go well?

- **Overlapping Speech Challenges:** The diarization process struggled with overlapping speech, leading to occasional misattributions of speaker labels, particularly in dynamic, multi-speaker discussions. This limitation was evident in the evaluation results (section 6.4.).
- **Summarization Nuances:** The summarization model occasionally missed subtle details in highly conversational segments, resulting in minor information loss in the summaries, a challenge inherent to abstractive summarization.
- **Processing Time for Long Files:** Transcribing audio files longer than 30 minutes, especially with larger Whisper models, was time-intensive, posing a potential drawback for real-time or high-throughput applications.
- **Integration Complexity:** Combining multiple NLP models (Whisper, PyAnnote, BART) and ensuring their seamless interoperability required significant troubleshooting and effort, highlighting the complexity of the pipeline.

7.3. Further Development

By addressing the identified limitations and enhance the application’s capabilities, several avenues for future development are proposed:

- **Model Fine-Tuning:** Fine-tune the Whisper and BART-based summarization models on domain-specific datasets (business meetings, academic discussions, …) to improve accuracy on specialized vocabulary and conversational patterns, potentially reducing WER and enhancing summary quality. Enhance speaker diarization by fine-tuning PyAnnote on datasets with complex speaker interactions.
- **Performance Optimization:** Optimize the processing pipeline to reduce transcription time for longer audio files, possibly through parallel processing or adopting more efficient model variants, making the system viable for real-time applications.
- **Additional NLP Features:** Incorporate advanced functionalities such as keyword extraction or sentiment analysis to provide deeper insights from transcribed text, enriching the application’s utility for analysis-driven use cases.

By pursuing the above improvements, the application can evolve into a more robust, efficient, and versatile tool, better serving a wide range of practical scenarios and user needs.

REFERENCES

1. Verified Market Research. (2023). *Audio Transcription Service Market Size, Growth, Market Insights & Forecast*. https://www.verifiedmarketreports.com/product/audio-transcription-service-market/

2. Bain, J., El Assady, M., & others (2023) *WhisperX: Extended Automatic Speech Recognition with Word-level Alignment and Diarization*. Available at: https://github.com/m-bain/whisperX

3. Bredin, H., Laurent, A., & Yin, R. (2020) *PyAnnote Audio: Neural building blocks for speaker diarization*. Proceedings of the ICASSP 2020, Barcelona, Spain

4. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., & Levy, O. (2020) *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. Proceedings of ACL 2020. Available at: https://arxiv.org/abs/1910.13461

5. Radford, A., Teehan, R., & Knight, M. (2022) *Robust Speech Recognition via Large-Scale Weak Supervision (Whisper)*. OpenAI. Available at: https://cdn.openai.com/papers/whisper.pdf

6. Zhang, Y., Kong, Q., Watanabe, S., & Wang, Z. (2021) *A Hybrid Approach for Meeting Summarization with Pre-trained Transformers*. Proceedings of Interspeech 2021. Available at: https://arxiv.org/abs/2109.06191

7. Otter.AI (2024). *Otter Voice Meeting Notes*. [online] Otter Voice Meeting Notes. Available at: https://otter.ai

8. Turboscribe.ai. (2023). *TurboScribe*. [online] Available at: http://turboscribe.ai

9. Assemblyai.com. (2025). *AssemblyAI | AI models to transcribe and understand speech*. [online] Available at: http://assemblyai.com

PROJECT PROPOSAL

My Project Proposal is available here:

https://drive.google.com/file/d/1tXSREIDntCrvP4EskeAdf9I7w6MvtswV/view?usp=sharing

APPENDIX

All the source code are available on my GitHub repository:

https://github.com/winnie2802/COMP1682-Final-Year-Project