Automated Podcast Transcription And Topic Segmentation

Internship Report

Submitted by

Ismail Sk

Conducted at

Infosys Springboard Internship 6.0 program

Acknowledgement

I would like to express my sincere gratitude to my project mentor for her continuous guidance, support, and encouragement throughout every stage of this project. She patiently helped us clear our doubts, provided valuable insights, and her guidance made the entire project journey smooth and well-structured.

I would also like to extend my heartfelt thanks to **Infosys Springboard** for organizing the **Infosys Springboard Internship 6.0 program** and providing me with this valuable opportunity. This internship program offered an excellent platform to apply theoretical knowledge to real-world problems, gain hands-on experience, and enhance both technical and professional skills. The learning resources, structured guidance, and exposure provided by Infosys Springboard were instrumental in the successful execution of this project.

I am truly thankful to everyone who directly or indirectly contributed to the successful completion of this project.

TABLE OF CONTENTS

LIST OF TABLES & FIGURES

Abstract

With the rapid growth of podcasts and long-duration audio content, there is a strong need for automated systems that can transcribe speech and organize audio into meaningful segments. This project presents a Django-based Audio Podcast Transcription and Topic Segmentation system that provides a complete end-to-end solution from audio upload to structured output.

In the proposed system, the user uploads a podcast audio file through the frontend interface, where multiple audio formats are supported, with .wav format recommended for faster processing. The backend pipeline first performs audio preprocessing, including resampling, noise reduction, silence removal, and voice activity detection to retain only speech-relevant portions. The cleaned audio is then transcribed using the Whisper speech-to-text model, generating an accurate textual transcript along with timestamped segment information.

For topic segmentation, an unsupervised semantic approach is employed using sentence embeddings and cosine similarity to detect topic boundaries within the transcript. To improve readability and usability, a LLaMA-based language model is used to generate concise, human-like titles for each detected segment. The entire workflow is integrated into a single automated pipeline within the Django framework.

As the final output, the user is provided with two downloadable result files: (i) a transcription.txt file containing the full podcast transcript, and (ii) a topic-segmented timestamps .txt file. This makes long podcast episodes easier to navigate, understand, and reuse, highlighting the core benefit and usability of the proposed system.

1. Introduction

Podcasts have become a widely used medium for sharing knowledge, discussions, and educational content. However, most podcast episodes are long and unstructured, making it difficult for listeners to quickly locate specific topics or revisit important sections. Manually transcribing and segmenting such audio content is time-consuming and inefficient, especially for large volumes of data.

This project focuses on building an automated system that simplifies podcast content analysis by converting raw audio into structured and user-friendly outputs. Using the Django framework, the system is designed to handle the complete processing pipeline, starting from audio upload to final result generation. Special attention is given to improving speech quality before transcription, ensuring better accuracy and reliability in real-world podcast recordings.

The proposed solution combines speech processing and natural language understanding techniques to identify topic changes within the podcast. Instead of relying on predefined labels or supervised training, the system uses semantic similarity to detect natural topic boundaries. Additionally, meaningful and concise titles are generated to represent each segment, improving content readability and navigation.

By providing downloadable transcription and topic-segmented timestamp files, the system helps users efficiently explore long podcast episodes. This project demonstrates how modern speech recognition and language models can be effectively integrated into a web-based application to enhance accessibility and usability of audio content.

Table 1.1 Software and Tools used

Package / Tool	Purpose	License	Link
Django	Backend web framework for pipeline integration	BSD	https://www.djangoproject.com
Librosa	Audio loading, resampling, and silence trimming	ISC	https://librosa.org
Silero VAD	Voice activity detection (speech-only extraction)	MIT	https://github.com/snakers4/silero-vad
Noisereduce	Background noise reduction	MIT	https://github.com/timsainb/noisereduce
OpenAI Whisper	Speech-to-text transcription (local ASR)	MIT	https://github.com/openai/whisper
Sentence-Transformers	Text embeddings for semantic segmentation	Apache 2.0	https://www.sbert.net
Scikit-learn	Cosine similarity and segmentation logic	BSD	https://scikit-learn.org
Meta-LLaMA-3.1-8B (GGUF)	LLM for human-like topic title generation	Meta License	https://huggingface.co
SoundFile	Read and write processed audio files	BSD	https://pysoundfile.readthedocs.io

2. Training works undertaken

During the internship, we gained practical knowledge of end-to-end audio processing and natural language understanding techniques required to build a real-world podcast analysis system. The training began with an introduction to audio preprocessing, where we learned how to clean raw audio by resampling, noise reduction, silence removal, and voice activity detection to retain only speech-relevant portions.

We then explored audio-to-text conversion using different speech recognition approaches and models, including traditional frameworks such as Kaldi and modern deep learning–based models. Through experimentation and comparison, we learned that OpenAI Whisper provided better transcription accuracy and robustness for podcast-style audio, and it was therefore selected for this project.

The next phase of training focused on text processing and topic segmentation. We learned how to convert transcribed text into meaningful segments using semantic similarity techniques. This included generating sentence embeddings, measuring similarity between consecutive segments, and detecting topic boundaries without using labeled data.

Additionally, we learned how large language models can be used to improve output quality by generating human-readable topic titles for each segment. Finally, all learned concepts were integrated into a Django-based automated pipeline, allowing the complete workflow to run seamlessly from audio upload to final output generation.

Overall, the internship provided hands-on exposure to real-world audio processing, speech recognition, and NLP techniques, which were directly applied in the successful implementation of this project.

3. Workflow Diagram

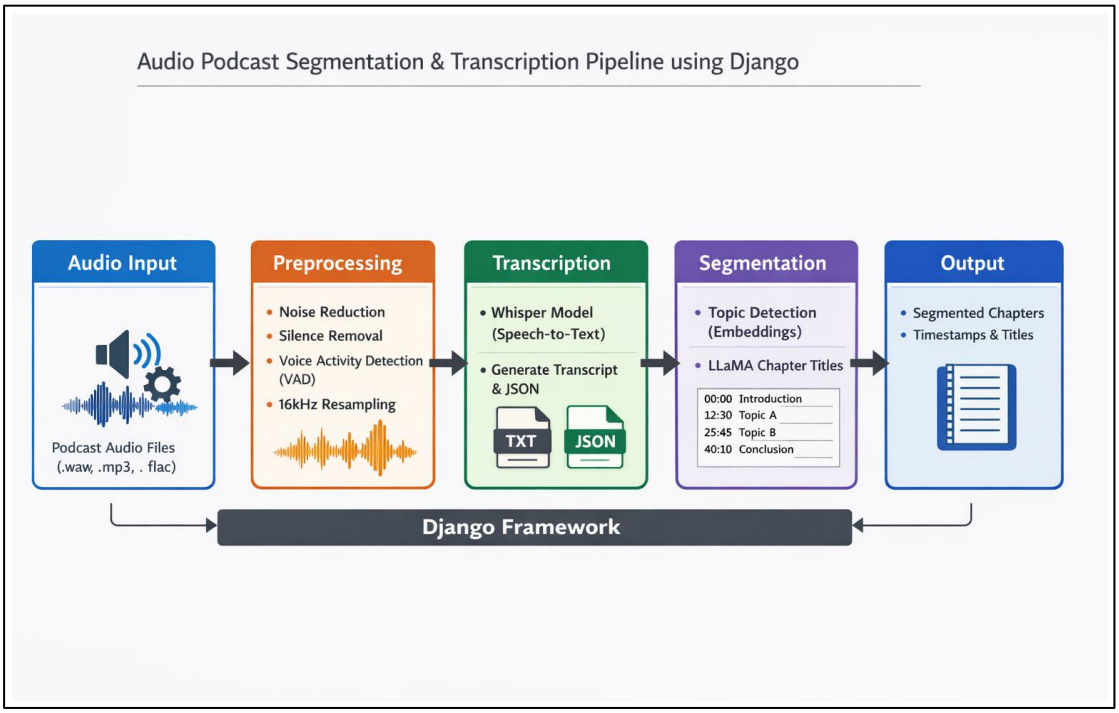

Figure 3.1 Project Workflow

4. Procedure

This section explains the complete end-to-end methodology adopted for implementing the Audio Podcast Transcription and Topic Segmentation system. The workflow is designed in a modular manner and integrated into the Django framework to ensure efficient and automated processing.

A. Audio Upload and Input Handling

The process starts at the frontend, where the user uploads a podcast audio file. Multiple audio formats are supported, although the .wav format is recommended for faster processing. The uploaded audio file is forwarded to the backend pipeline for further operations.

B. Audio Preprocessing

The raw audio is first preprocessed to improve speech recognition accuracy. The audio is resampled to 16 kHz, converted to mono, and normalized. Background noise is reduced, followed by silence trimming to remove non-informative sections. Voice Activity Detection (VAD) using the Silero VAD model is then applied to retain only speech frames. This step ensures that the transcription model processes clean and speech-focused audio.

C. Speech-to-Text Transcription

The preprocessed audio is transcribed using the OpenAI Whisper small model. The small model was selected due to GPU resource limitations, and it performs efficiently for English-language podcasts. It is noted that using Whisper medium or large models can significantly improve transcription accuracy and also enable multilingual transcription capabilities. The transcription process generates both a plain text transcript and a structured JSON output containing segment-level timestamps.

D. Semantic Topic Segmentation

The transcription JSON is used as input for topic segmentation. Each transcript segment is minimally cleaned to preserve semantic meaning and then converted into vector representations using the SentenceTransformer model (all-MiniLM-L6-v2). Cosine similarity is computed between consecutive embeddings, and a predefined similarity threshold is used to detect topic boundaries. Segments with low semantic similarity are grouped into separate topic blocks, enabling unsupervised topic segmentation.

E. Human-like Topic Title Generation

To generate meaningful and readable chapter titles, the system uses the Meta-LLaMA-3.1-8B-Instruct-Q4_K_M.gguf model downloaded from Hugging Face. This large language model generates short, natural-language titles for each detected topic block, resulting in human-like and context-aware segment descriptions.

F. Output Generation

After processing, the system produces two final outputs for the user:

i. A transcription.txt file containing the complete podcast transcript.
ii. A topic-segmented timestamps file (.txt) containing time-aligned chapter titles.

These files can be downloaded directly from the frontend interface.

G. Django-Based Pipeline Integration

All stages of the workflow—preprocessing, transcription, and segmentation—are combined into a single automated pipeline within the Django backend. This design allows one-click execution from audio upload to final result generation, ensuring scalability, maintainability, and ease of use. And also Context-aware question answering from podcast transcripts

5. Results and discussion

The proposed system was evaluated on podcast audio inputs processed using the Whisper small model, selected due to GPU resource limitations. Despite its lightweight architecture, the model produced good and acceptable transcription results for English-language podcast content. The generated transcripts were clear, coherent, and suitable for downstream semantic processing.

Figure 5.1 Timestamps & Topics

The accuracy of the transcription stage plays a critical role in the overall performance of the system. Since topic segmentation is based on semantic similarity of transcribed text, any transcription errors directly affect the quality of segment embeddings and boundary detection. Inaccurate or incomplete transcripts can lead to incorrect topic boundaries and reduced coherence in segmented chapters. Therefore, reliable speech-to-text conversion is a fundamental requirement for effective semantic segmentation.

=== TRANSCRIPTION ===
{'WER': 0.41, 'CER': 0.331}

=== TOPIC SEGMENTATION ===
{'Topic_Coherence': 0.547, 'Boundary_Accuracy': 0.32}

=== PERFORMANCE ===
{'Audio_Duration_sec': 2100, 'Processing_Time_sec': 850, 'RTF': 0.405}

=== USABILITY ===
{'Readability_Word_Count': 5832, 'Search_Readiness': 'Good'}

Evaluation

=== TRANSCRIPTION_QUALITY ===
{'WER': 0.561, 'CER': 0.462}

=== TOPIC_SEGMENTATION_LOGIC ===
{'Topic_Coherence': 0.849, 'Semantic_Boundary_Alignment': 1.0, 'Total_Predicted_Topics': 8}

=== GENAI_USAGE ===
{'ASR_Model': 'Whisper Small', 'LLM_Model': 'Groq API (llama-3.3-70b-versatile)', 'Usage': 'ASR output evaluation + Topic title generation'}

=== SAFETY_HANDLING ===
{'Execution_Mode': 'Hybrid (Local + Cloud)', 'User_Data_Stored': False, 'Cloud_API_Used': 'Groq API'}

=== COST_AWARENESS ===
{'Model_Choice': 'Whisper Small', 'Reason': 'Low GPU usage, zero API cost', 'Inference_Cost': 'Low (ASR local) + API-based (LLM)'}

=== CODE_QUALITY ===
{'Modular_Pipeline': True, 'Reusable_Components': True, 'Readable_Code': True}

=== DOCUMENTATION ===
{'Docstrings': True, 'Inline_Comments': True, 'Clear_File_Structure': True}

=== EXPLAINABILITY ===
{'Pipeline_Explainable': True, 'Stepwise_Processing': True, 'Metric_Transparency': True}

Figure 5.2 Home page

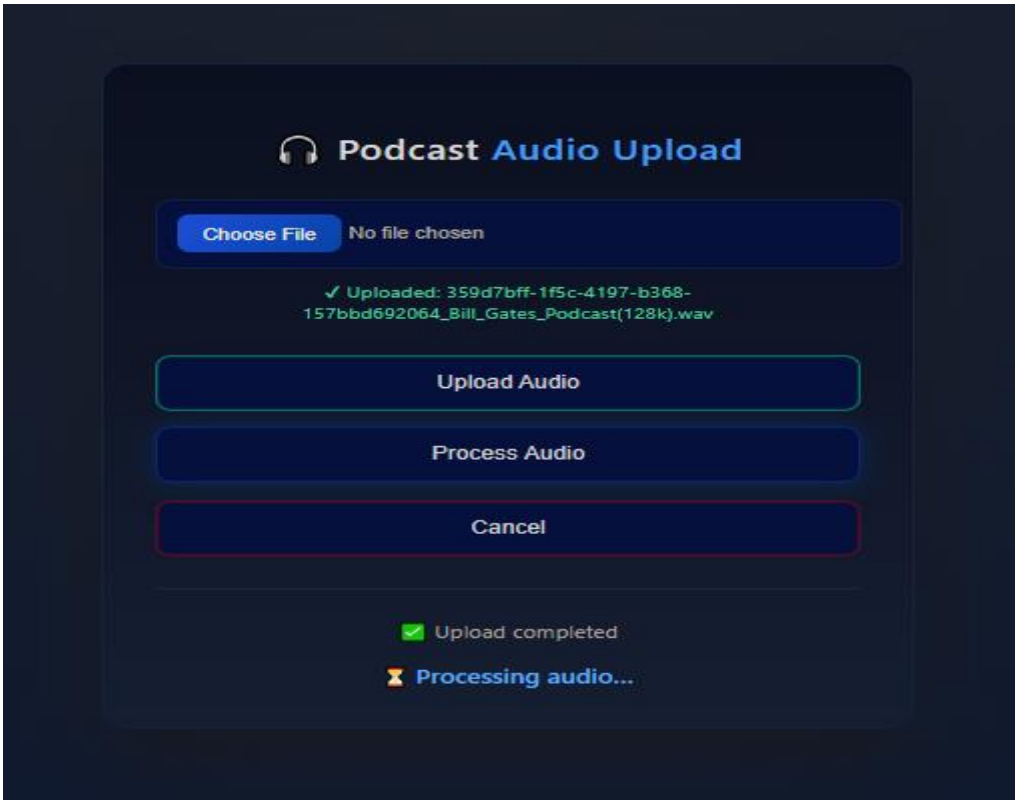

Figure 5.3 Audio uploaded & proceed

Figure 5.4 Transcription & Segmentation download option

Figure 5.5 Ask from Transcript

Using the Whisper small model, the system was able to generate meaningful topic segments with human-like titles. The segmentation output demonstrated logical topic transitions and reasonable timestamp alignment, indicating that the transcription quality was sufficient for unsupervised semantic analysis. Figures illustrating the transcription output and topic-segmented timestamp file show that the system successfully transforms raw audio into structured and user-friendly results.

It is observed that employing more advanced Whisper variants, such as the medium or large models, can significantly enhance transcription accuracy and robustness. Improved transcription quality would lead to more precise sentence embeddings, stronger semantic similarity detection, and consequently, more accurate topic segmentation. Additionally, advanced models support multilingual transcription, which can further extend the applicability of the system.

Overall, the results confirm that even with hardware constraints, the proposed system delivers reliable and usable outputs. The achieved results validate the effectiveness of the end-to-end pipeline, while also highlighting the potential performance gains achievable through the use of more advanced transcription models.

6. Future scope of the work

The proposed Audio Podcast Transcription and Topic Segmentation system can be further enhanced in several directions. One major improvement involves the use of advanced Whisper models (medium or large) to achieve higher transcription accuracy and support multilingual podcast content. Improved transcription quality will directly enhance semantic topic segmentation performance.

The system can be extended to support real-time or near real-time processing, enabling live podcast or webinar analysis. Additionally, integrating speaker diarization can help distinguish between multiple speakers, improving transcript clarity and segment relevance.

Future work may also include adding search and indexing capabilities, allowing users to search within transcripts using keywords or topics. The generated segments can be linked with summaries to provide quick content overviews.

Further optimization of the Django backend and model deployment can improve scalability, making the system suitable for large-scale podcast platforms. These enhancements will increase the usability, accuracy, and practical applicability of the proposed system.

7. Conclusion

This project presented an end-to-end Audio Podcast Transcription and Topic Segmentation system implemented using the Django framework. The system successfully processes raw podcast audio through preprocessing, transcription, and semantic segmentation to generate structured and user-friendly outputs.

Using the Whisper small model, acceptable transcription accuracy was achieved under GPU constraints, enabling effective semantic topic segmentation. The results demonstrate that accurate transcription is essential for reliable topic boundary detection and meaningful chapter generation.

Overall, the system provides a practical and automated solution that allows users to upload audio files and download both transcription and topic-segmented timestamp files, improving accessibility and navigation of long podcast content.

References

[1] Radford *et al.*, "Robust Speech Recognition via Large-Scale Weak Supervision," OpenAI, 2022. [Online]. Available: https://github.com/openai/whisper

[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proc. EMNLP*, 2019.

[3] T. Wolf *et al.*, "Transformers: State-of-the-Art Natural Language Processing," *Proc. EMNLP*, 2020.

[4] S. Hershey *et al.*, "CNN Architectures for Large-Scale Audio Classification," *Proc. ICASSP*, 2017.

[5] Silero Team, "Silero Voice Activity Detection," GitHub Repository. [Online]. Available: https://github.com/snakers4/silero-vad

[6] Meta AI, "Meta LLaMA 3: Open Foundation and Instruction Models," 2024. [Online]. Available: https://huggingface.co

[7] D. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[8] Django Software Foundation, "Django Web Framework," [Online]. Available: https://www.djangoproject.com