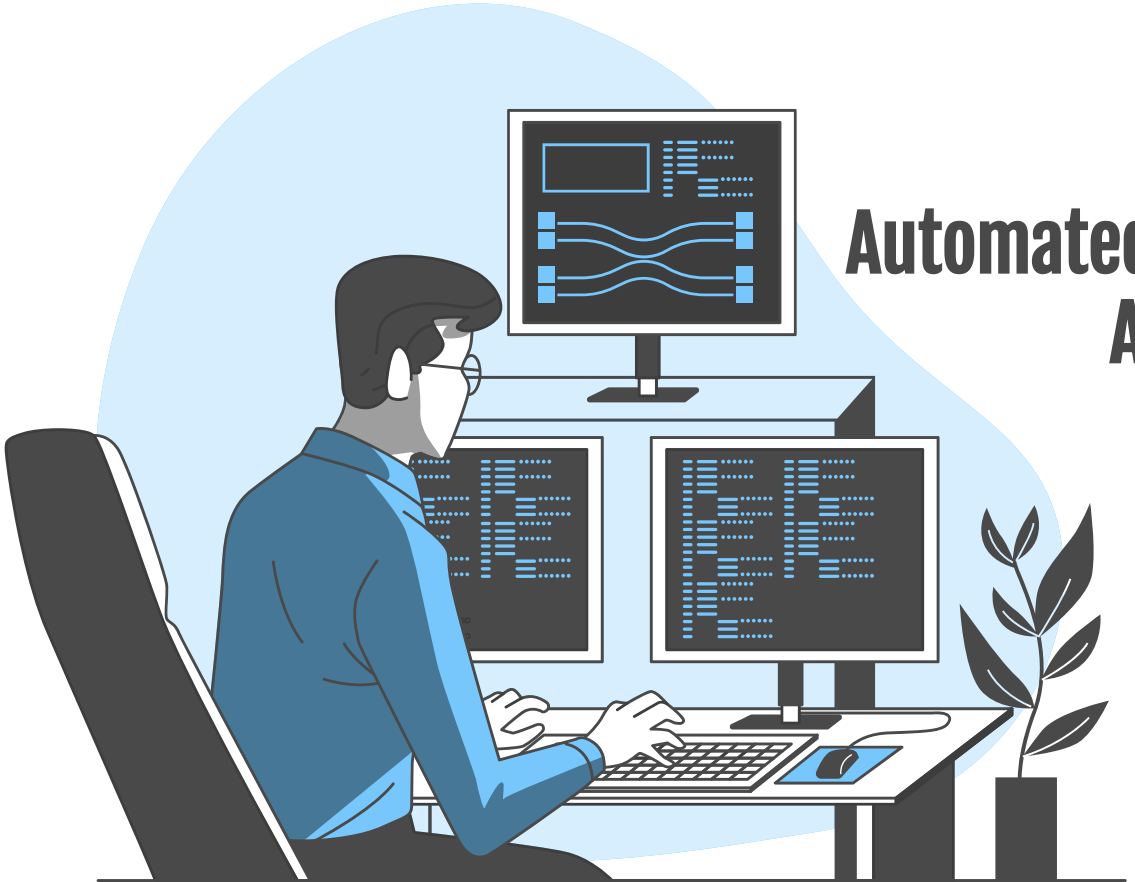


Automated Podcast Transcription And Topic Segmentation

Ismail Sk

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.

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.

Audio Preprocessing

1. Load raw audio file
2. Resample audio to **16 kHz**
3. Convert audio to **mono**
4. Normalize audio amplitude
5. Apply **background noise reduction**
6. Remove silent portions (silence trimming)
7. Perform **Voice Activity Detection (Silero VAD)**
8. Retain only speech segments
9. Save the cleaned audio as a processed .wav file

Audio Transcription

Steps Involved

- Load **preprocessed audio** files
- Initialize **Whisper small model** on GPU
- Perform **speech-to-text transcription**
- Generate full **text transcript (.txt)**
- Generate **timestamped JSON output**
- Store outputs for **topic segmentation**

Model Note

- Whisper small used due to GPU limits
- Performs well for English audio
- Medium/Large models improve accuracy and enable multilingual support

Topic Segmentation & Chapter Generation

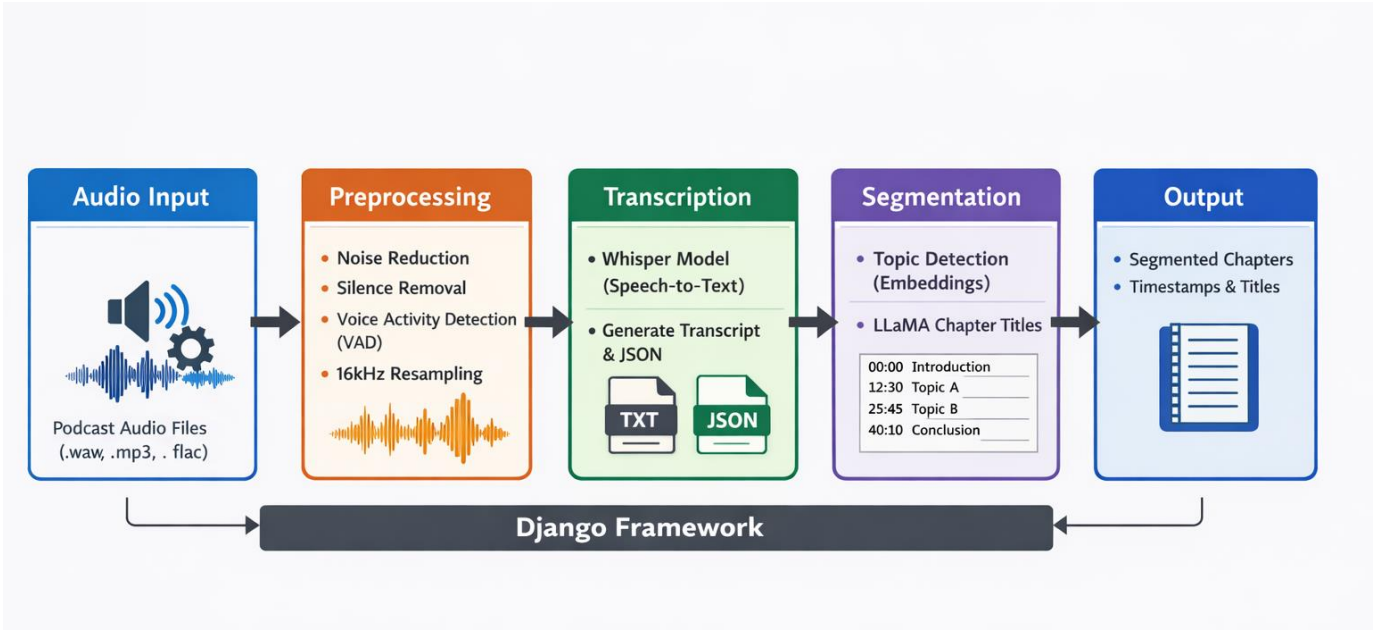
Steps Involved

- Load Whisper JSON transcript
- Clean text (minimal preprocessing)
- Convert text to embeddings using SentenceTransformer (all-MiniLM-L6-v2)
- Compute cosine similarity between consecutive segments
- Detect topic boundaries using similarity threshold
- Group segments into topic blocks
- Generate short chapter titles using LLaMA-3.1-8B (GGUF)
- Create timestamped topic file (.txt)

Models Used

- SentenceTransformer → semantic understanding
- Meta-LLaMA-3.1-8B-Instruct → human-like titles

Workflow Diagram



Results

- Achieved **accurate English transcription** using Whisper small model
- Successfully generated **topic-wise segmented chapters**
- Produced **two downloadable outputs**: transcript and timestamped topics
- Demonstrated effective **end-to-end automation** using Django pipeline

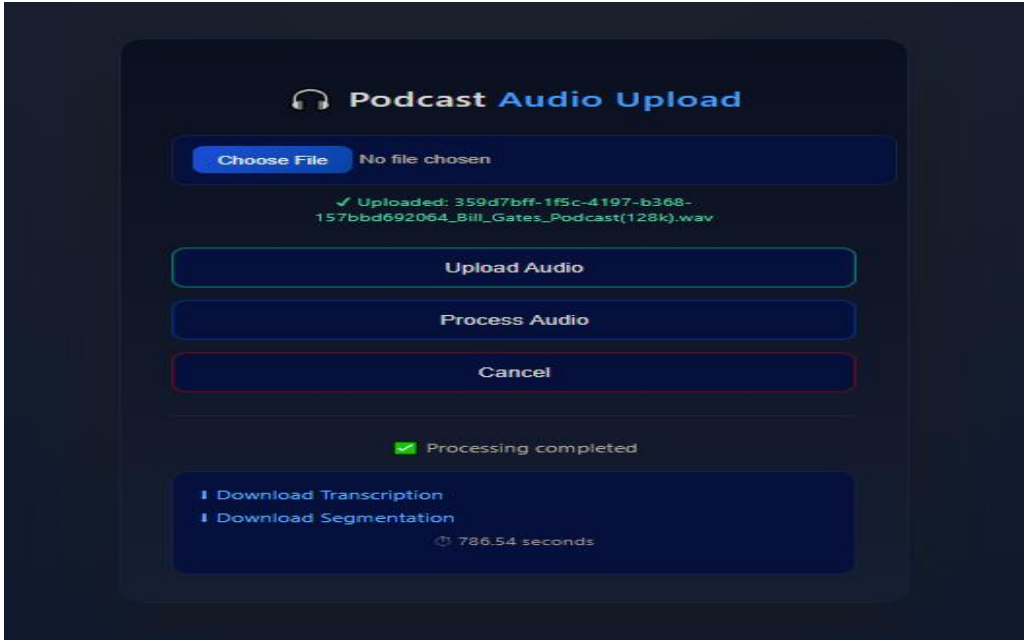
Results

Home page

Results

Audio uploaded & proceed

Results



Transcription & Segmentation download option

Results

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

=== TOPIC_SEGMENTATION_LOGIC ===
{'Topic_Coherence': 0.547, 'Boundary_Accuracy': 0.32, 'Total_Predicted_Topics': 25}

=== GENAI_USAGE ===
{'ASR_Model': 'Whisper Small', 'LLM_Model': 'Meta-LLaMA-3.1-8B-Instruct (GGUF)', 'Usage': 'Transcription + Human-like topic titles'}

=== SAFETY_HANDLING ===
{'Execution_Mode': 'Fully Local', 'User_Data_Stored': False, 'Cloud_API_Used': False}

=== COST_AWARENESS ===
{'Model_Choice': 'Whisper Small', 'Reason': 'Low GPU usage, zero API cost', 'Inference_Cost': 'Minimal'}

=== 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}

Future Work

- Use **Whisper medium/large models** for higher accuracy and multilingual support
- Enable **real-time or live podcast processing**
- Add **speaker diarization** for multi-speaker podcasts
- Implement **search and indexing** within transcripts
- Optimize backend for **scalability and large-scale deployment**

—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

- Radford *et al.*, “Robust Speech Recognition via Large-Scale Weak Supervision,” OpenAI, 2022. [Online]. Available: https://github.com/openai/whisper
- N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” *Proc. EMNLP*, 2019.
- T. Wolf *et al.*, “Transformers: State-of-the-Art Natural Language Processing,” *Proc. EMNLP*, 2020.
- S. Hershey *et al.*, “CNN Architectures for Large-Scale Audio Classification,” *Proc. ICASSP*, 2017.
- Silero Team, “Silero Voice Activity Detection,” GitHub Repository. [Online]. Available: https://github.com/snakers4/silero-vad
- Meta AI, “Meta LLaMA 3: Open Foundation and Instruction Models,” 2024. [Online]. Available: https://huggingface.co
- D. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- Django Software Foundation, “Django Web Framework,” [Online]. Available: https://www.djangoproject.com

Thanks!