# Podcast Summarizer Using Machine Learning

[1]Dr.Lutful Islam, [2]Arshad Ahmed, [3]Ansari Furqan, [4]Usama Mulla, [5]Shaikh Awez
*[1] Professor, [2] Student, [3] Student, [4] Student, [5] Student*
*Department of Computer Engineering, M. H. Saboo Siddik College of Engineering, Mumbai 400008, India*

***Abstract*—The project titled "Podcast Summarizer using ML" aims to develop a sophisticated system that automatically generates concise and accurate summaries of spoken content in podcasts using advanced natural language processing (NLP) and machine learning techniques. The system will leverage speech-to-text technologies to transcribe audio input and employ summarization algorithms to extract the most important information, delivering coherent and informative summaries. This tool will enhance user experience by allowing listeners to quickly grasp the key points of lengthy podcasts, making the vast amount of audio content more accessible and easier to navigate. The project also addresses challenges such as handling diverse accents, varying audio quality, and ensuring summarization accuracy, providing a valuable resource for podcast enthusiasts, researchers, and content creators.**

***Keywords:*** **Summarizer, NLP, BERT, Sentiment Analysis, Podcasts.**

## I. INTRODUCTION

In today's fast-paced digital era, podcasts have become a cornerstone for sharing knowledge, entertainment, and news. However, their popularity comes with a challenge: the average episode spans 43 minutes, deterring time- constrained listeners. Lengthy, informal conversations with digressions and multi-speaker dynamics make it difficult to extract key insights efficiently, often leading to missed information or incomplete engagement.

To address this, an AI-driven solution automates the conversion of lengthy audio into concise summaries. Leveraging speech-to-text technologies like OpenAI's Whisper[6], it transcribes spoken content accurately, even with accents or background noise. Advanced natural language processing techniques then analyze the text, identifying key themes and generating structured summaries. This hybrid approach combines extractive methods to highlight critical sentences and abstractive techniques to paraphrase content, ensuring clarity and brevity while preserving context.

The system tackles technical challenges like noise reduction, speaker identification, and maintaining coherence in multi-topic discussions. By doing so, it enhances accessibility for non- native speakers, hearing-impaired users, and those in noisy environments. This democratizes access to podcast content, enabling users to grasp core ideas quickly and decide which episodes merit deeper exploration.

As podcasts dominate digital media, this tool aligns with the demand for efficient content consumption. By condensing hours of audio into digestible insights, it empowers users to navigate vast libraries effectively. Future advancements aim to integrate real-time summarization for live episodes and personalized outputs based on user preferences, redefining podcast engagement to be more adaptive, inclusive, and aligned with modern needs.

## II. LITERATURE REVIEW

Existing research in podcast summarization has explored various approaches to address the challenges of converting audio content into concise summaries. Several studies have demonstrated promising results using different methodologies. Vartakavi's PodSumm[8] employs a fine-tuned PreSumm model with data augmentation to overcome dataset limitations, while Derkach's work[1] highlights the effectiveness of GPT-2 for abstractive summarization. Kang and Roy's framework integrates LLMs with audio encoders for flexible summarization, outperforming traditional methods. Song et al. combine extractive and abstractive techniques to handle open-domain podcasts, focusing on informal language and diverse topics. Real-time summarization systems by Jeeva et al. and Liya et al. emphasize quick, digestible outputs while addressing audio quality challenges.

However, significant limitations persist in current systems. Many struggles with contextual understanding, often oversimplifying complex

discussions. Audio quality variability and multi-speaker scenarios frequently degrade performance, while cultural and linguistic nuances are frequently missed. These shortcomings highlight critical research gaps, including the need for improved transcription techniques for diverse accents, better evaluation metrics, and platform integration[7]. Ethical considerations around content attribution and the lack of multilingual capabilities also present important challenges that require further investigation to make podcast summarization more robust and inclusive.

Table 1. Research Gap

| Work Cited | Paper | Advantages | Disadvantages |
|---|---|---|---|
| 1. Aneesh Vartakavi | PodSumm: Podcast Audio Summarization | The paper addresses the specific challenge of summarizing long form audio content, which is increasingly relevant given the rise of podcasts. | The accuracy of the summarization depends heavily on the quality of speech-to- text transcription, which may be error-prone, especially for noisy or accented speech. |
| 2. Ilia Derkach | Abstractive Summarization from Audio Transcription | The paper tackles abstractive summarization, which generates more human-like and concise summaries compared to extractive methods that merely copy portions of the transcript. | The abstractive summarization models, particularly those based on neural networks, are computationally intensive and may not be easily scalable or feasible for real-time applications without significant resources. |
| 3. Wonjune Kang, Deb Roy | Prompting Large Language Models with Audio for General Purpose Speech Summarization | The paper leverages the power of LLMs (such as GPT or similar architectures) to enhance the quality of summarization, which can handle a wide range of general-purpose speech data. | Large Language Models are computationally expensive and require significant resources for both training and inference, which may hinder real-time or large-scale implementations. |
| 4. Kaiqiang Song, Xiaoyang Wang, Dong Yu | Automatic Summarization of Open Domain Podcast Episodes | The paper addresses the challenge of summarizing open-domain podcast episodes, which contain diverse and unstructu | The accuracy of the summarization model is dependent on the quality of the audio-to-text transcription. Poor transcription (due to accents, |

| | | | |
|---|---|---|---|
| | | red content, making it highly relevant for real world podcast applications. | background noise, etc.) can negatively affect the final summary. |
| 5. S. Jeeva , GudlaSai Sujan ,ArutlaSiddharth Redd | Audio Summarization in Real Time for Podcasts | The paper focuses on real-time summarization, which allows users to get | Real-time summarization may sacrifice accuracy or depth of the summaries in favor of speed, potentially |
| | | summaries on the fly while listening to podcasts, offering immediate insights and time-saving benefits. | y leading to less informative or less coherent summaries. |
| 6. Liya B.S , Seenuvasan V, Sathya Prakash K | Audio summarization in real time for podcast, speeches and audio books | The paper emphasizes real-time summarization for a variety of audio content, including podcasts, speeches, and audiobooks, which makes it versatile for | Real-time summarization requires considerable computational resources to process and summarize audio efficiently, making it less feasible for resource |
| | | different audio formats. | constrained environments. |

## III. PROPOSED SYSTEM

This project represents a significant advancement in the way we interact with audio content, addressing the challenges posed by the rapidly growing podcast industry. With thousands of episodes available on diverse topics, listeners often face the dilemma of time constraints, making it difficult to sift through lengthy recordings to find pertinent information. This project utilizes sophisticated natural language processing algorithms to analyze and condense podcast episodes into concise summaries that capture the essence of discussions, key insights, and important themes.

By converting spoken language into structured text, the summarizer enhances accessibility for a wider audience, including those who may prefer to skim content rather than listen to every minute. Furthermore, the tool aims to improve user engagement by allowing listeners to personalize their summaries based on interests, which can help them prioritize content that resonates with their preferences. As a result, users can make informed decisions about which episodes to dive into fully, saving time while still gaining valuable.

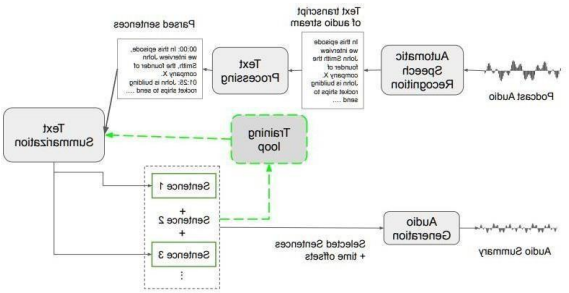


Figure. 1: Architecture Design

Additionally, the project seeks to tackle the issue of multi-speaker dynamics, ensuring that the summarization process accurately reflects the contributions of different hosts and guests, thus maintaining the richness and context of conversations. By integrating this summarization technology with existing podcast platforms, the project envisions a seamless user experience where summaries are readily available at the click of a button. Ultimately, the "Podcast Summarizer using AI" aims not only to streamline the listening experience but also to foster a deeper understanding of diverse topics, empowering users to stay

informed and engaged in an increasingly information-rich environment.

Moreover, the project recognizes the importance of user engagement and interactivity in modern digital experiences. As such, the summarizer could incorporate features that allow users to pose questions or request specific information from the podcast content, leading to more dynamic interactions. This could be especially valuable for educational or informative podcasts, where users might seek clarification on subjects or themes. By making summaries interactive, the project can cater to different learning styles and preferences, further enhancing the utility of the summarization tool.

Additionally, the ethical implications of summarization are an essential aspect of this project. As AI- generated summaries can alter the perception of the original content, it is crucial to ensure that the tool respects creators' rights and maintains the integrity of the source material. This includes providing clear attribution to the original podcasts and potentially offering creators insights into how their content is being summarized and consumed. By addressing these ethical considerations, the project aims to foster a sense of trust and transparency between content creators and listeners.[3]

## IV. IMPLEMENTATION

The implementation of Podcast summarizer was guided by a modular, scalable, and user-centric design philosophy. This section elaborates on the key components of the system, detailing the backend, frontend, API, and database implementations. The goal was to create a seamless platform that effectively bridges the gap between elongated podcasts and user understanding, ensuring robust functionality and a smooth user experience.

### 4.1 System Overview

The Podcast Summarizer using AI is an automated system designed to convert lengthy podcast episodes into concise, informative summaries. Leveraging advanced speech recognition and natural language processing technologies, the platform provides users with quick access to key insights while maintaining the context and essence of the original content.

The system follows a three-tier architecture

- Frontend – A user-friendly web interface that allows users to upload podcasts, view summaries, and customize output preferences. The interface is responsive and accessible across devices.
- Backend – Processes audio input using speech-to-text (Whisper ASR) and applies NLP-based summarization (BERT, PEGASUS) to generate structured summaries. The backend also handles speaker diarization, noise reduction, and contextual analysis.
- Database – Stores user data, podcast transcripts, and generated summaries securely, ensuring fast retrieval and scalability.

By integrating these layers, the system delivers accurate, efficient, and adaptable podcast summarization, catering to diverse user needs while addressing challenges like multi-speaker conversations, varying audio quality, and contextual understanding. Future enhancements include real-time processing and multilingual support.

### 4.2 Backend Implementation

The backend of the Podcast Summarizer serves as the core engine, handling podcast ingestion, transcription, summarization, sentiment analysis, and modular NLP-based processing. Built using Python and Streamlit (with a Flask expansion planned), the backend ensures scalability and modularity to support advanced audio intelligence workflows.

#### 4.2.1 Audio & Video Ingestion

This module enables users to upload podcasts or provide a YouTube video link for processing. Key responsibilities include:

- Upload Support: Accepts audio (.mp3,.wav)or video (.mp4) files via Streamlit's interface.
- YouTube Support: Uses yt-dlp to fetch and download videos from user-provided URLs.
- Audio Extraction: Extracts audio using MoviePy or FFmpeg for downstream processing.
- Storage: Uploaded files are saved with session-based unique names for later use.

4.2.2. Transcription & Speaker Diarization This core module is responsible for converting audio content into structured text:

- Whisper Integration: Utilizes OpenAI's Whisper model for accurate transcription[6], handling multiple languages and long-form speech.
- pyannote.audio: Optional speaker diarization separates text by individual speakers, labeling them (e.g., Speaker 1, Speaker 2).
- Timestamps: Each utterance is aligned with its original time range in the audio.

4.2.3 Summarization Engine

The summarization module extracts the essence of the podcast into a concise, readable format:

- Chunking & MapReduce: Long transcripts are broken into overlapping chunks and summarized using a two- level summarization approach.
- Hugging Face Transformers: Models like BART[5] or T5 are used for generating high-quality summaries.
- LLAMA 2 from Grok API: For ultra- fast inference when supported by hardware.

4.2.4 Integration with APIs

While currently powered by Streamlit, the backend is built to be transitioned into Flask- based REST APIs.

4.2.5 Sentiment & Keyword Extraction

Natural Language Processing techniques analyze the mood and essence of the podcast:

- Sentiment Analysis: Uses TextBlob or VADER to assess polarity and subjectivity.
- Keyword Extraction: Employs NLTK for stopword removal and CountVectorizer for extracting dominant keywords.
- Topic Modeling: Applies LDA (Latent Dirichlet Allocation) to identify major themes across the episode.

4.2.6 Translation & Text-to-Speech

To enhance accessibility and reach:

- Google Translate API: Translates summaries and transcripts to multiple target languages.
- gTTS or Polly: Converts summary into downloadable speech, enabling voice-based summaries.

4.3 Frontend Implementation

The user interface is built using Streamlit for rapid prototyping, with a modular layout and tab- based navigation. Each major NLP task is presented as a separate tab for seamless exploration.

Key Features:

- Upload Section: Drag-and-drop interface for podcasts or YouTube URL input.
- Transcript Viewer: Displays timestamped or speaker-separated transcripts.
- Summary Display: Presents a concise overview of the episode.
- Keyword & Sentiment Tabs: Graphically shows emotional tone, polarity, and key concepts.
- Translation Tab: Allows users to view content in other languages.
- Audio Player: Play the summarized version using generated TTS output.

UI elements are designed for clarity and accessibility, with potential future migration to React + Tailwind for enhanced control.

4.4 Database Management

The Podcast Summarizer platform uses PostgreSQL as the primary relational database to store and manage structured data efficiently. This ensures data integrity, scalability, and ease of querying for complex relationships between podcast sessions, users, transcripts, and analyses.

PostgreSQL:

Stores structured data including user session metadata, uploaded file references, transcription segments, speaker diarization results, sentiment scores, keywords, and summaries.

Well-suited for joining multiple related tables (e.g., podcast_sessions, transcriptions, speaker_segments, nlp_analysis).

Allows use of advanced queries, indexing, and stored procedures for optimized analytics and reporting.

The combination of Firebase and MongoDB enhances the platform's ability to handle dynamic data flows and maintain scalability.

4.5 System Workflow

The Podcast Summarizer is designed for a seamless end-to-end user journey:

- Upload or Link: User uploads a file or

provides a YouTube link.

- Transcription: Audio is transcribed with Whisper, optionally separated by speakers.
- NLP Analysis: Summary, keywords, sentiment, and topics are extracted.
- Translation & TTS: Users can convert content to other languages or speech.
- Output Delivery: The UI presents organized results across tabs, optionally allowing export as text/audio.

4.6 Technology Stack Integration

The development of the Podcast Summarizer platform relies on a modern and scalable technology stack tailored for audio processing, summarization, and intelligent content analysis. The stack ensures high performance for NLP workflows, seamless audio handling, and a modular architecture to support both rapid prototyping and future scalability. Each component plays a strategic role, from transcription to summarization, keyword extraction, and user interaction.

4.6.1 Frontend Technologies

The frontend of Podcast Summarizer is built using React, providing a dynamic and component-driven interface that supports responsive and intuitive user experiences. Features such as audio upload, real-time transcript visualization, summarization output, and segmented tabs for each NLP task are handled through this interactive interface.

Vite serves as the build tool and development server, enabling lightning-fast hot module reloading and optimized bundling for production. It significantly improves developer productivity and ensures the frontend remains lightweight and performant.

Tailwind CSS is used to design a visually clean, mobile-responsive UI with utility-first styling. This allows for rapid prototyping and the creation of reusable design components across the application — including transcript cards, summary containers, keyword lists, and sentiment analysis charts.

Axios facilitates communication between the frontend and backend services by handling HTTP requests efficiently. It ensures smooth data flow when fetching processed transcripts, summaries, or audio files from the backend, with built-in support for error handling and asynchronous operations.

React Query further enhances data management by abstracting fetch logic, caching, and background updates. It simplifies the integration with backend APIs, especially when polling for long-running transcription or summarization jobs.

4.6.1 Backend Technologies

The backend is designed using Python with Flask, a lightweight microframework that handles core API endpoints for file uploads, audio processing, transcription, and text summarization. Flask enables fast development and easy integration with machine learning libraries and external APIs.

FFmpeg and yt-dlp are used for pre-processing audio files — including downloading audio from YouTube, trimming, resampling, and converting audio formats before feeding them into transcription pipelines.

The backend is modular, with each NLP component (transcription, summarization, sentiment analysis, etc.) implemented as independent service logic, enabling flexibility and easy scalability.

4.6.2 Databases

PostgreSQL is used as the primary relational database to store user uploads, job status, processed transcripts, summary data, and system logs. SQLAlchemy ORM ensures smooth integration with Python, making it easier to interact with database tables via Pythonic classes and models.

GitHub manages source control, collaboration, and CI/CD pipelines, ensuring continuous updates and version control for the application codebase.

Streamlit is used as an internal prototyping tool during development and testing of individual NLP components such as keyword extractors, topic models, and summarizers. This allows for rapid experimentation and visualization of outputs

4.6.3 Integration Workflow

- Frontend Communication: The frontend, built with React and styled with Tailwind CSS, uses Axios to interact with backend APIs. Redux Toolkit manages the application state for smooth transitions between components.
- Backend Processing: Spring Boot and Flask power the backend logic, handling requests, processing data, and interacting with the databases.

## V. RESULTS AND DISCUSSIONS

A. RESULTS

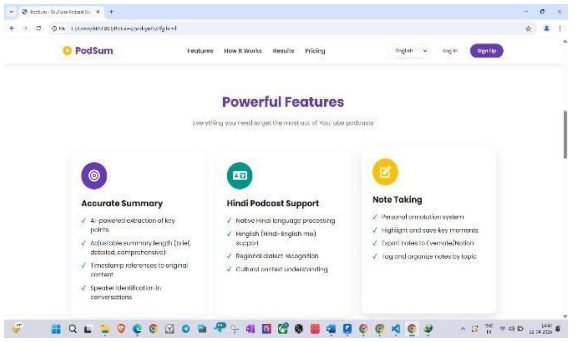


Figure 2: Feature Overview Page

Figure 2 showcases the core features, a YouTube podcast summarization tool. The platform offers accurate AI-powered summaries with adjustable length, timestamp references, and speaker identification. It supports Hindi and English podcasts with native language processing and regional dialect recognition. Additionally, this includes a personalized note-taking system that allows users to highlight, save, and export key moments. These features demonstrate a user-friendly approach to podcast summarization and serve as a baseline for comparison in evaluating our proposed system.

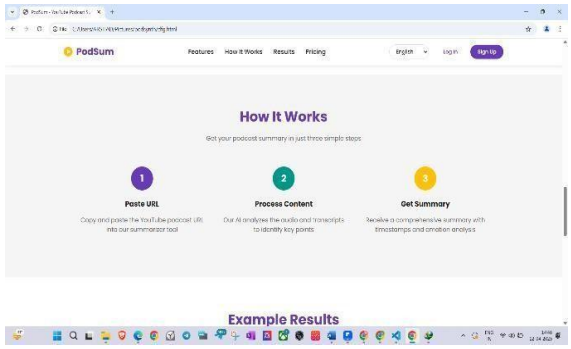


Figure 3: Workflow of the Summarization

This figure illustrates the simple three-step workflow of the system tool for summarizing YouTube podcasts. The process begins by pasting the podcast URL into the summarizer interface. Next, the system analyzes the audio and transcripts using AI to extract key points and relevant content. Finally, it generates a comprehensive summary enriched with timestamps and emotion analysis. This user- friendly workflow highlights automation and accessibility, serving as a reference model for comparison in the design of our proposed system.

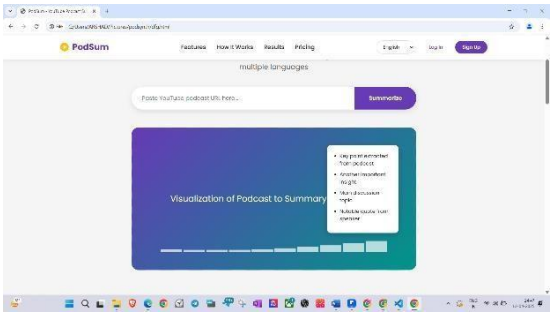


Figure 4: Visualization of Podcast to Summary Conversion

This figure demonstrates how system visualizes the transition from a full-length podcast to a concise summary. After the user pastes a YouTube podcast URL and initiates the summarization, the tool extracts key points, discussion topics, and speaker quotes. The timeline graphic at the bottom represents segments of the podcast, each annotated with important insights. This visualization helps users quickly identify and navigate through essential parts of the content. It enhances usability and offers a clear overview of the summarized material in a visually intuitive manner.

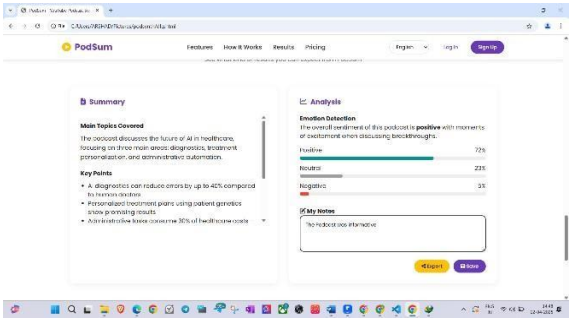


Figure 5: Summary Generation

This figure presents the summarized output and emotion analysis generated by the system tool for a podcast on AI in healthcare. The summary highlights main topics such as diagnostics, personalized treatment, and automation, with key points outlining the impact and advantages of AI. On the right, an emotion detection chart shows that 72% of the content conveys a positive tone, 23% is neutral, and only 5% is negative. The analysis suggests excitement when discussing breakthroughs. A notes section allows users to add personal reflections to the podcast. The export and saving features enhance usability for future reference.

B. DISCUSSIONS

The proposed system, “Podcast Summarizer using Machine Learning,” successfully demonstrates the

integration of multiple Natural Language Processing techniques to automate the summarization of long-form audio content. The results from our implementation highlight the practicality and usefulness of such a system, especially in educational, professional, and infotainment domains where time saving is essential.

One of the key strengths of our model is the use of pre-trained models like Whisper[6] for transcription and BART[5] for summarization. These models provide high accuracy and fluency in output without requiring extensive training on custom datasets. The system also ensures that the summaries generated retain the core context and structure of the original podcasts, making them useful for quick comprehension.

However, there are a few limitations observed. The quality of the summaries can degrade for audio inputs with poor sound quality, multiple speakers, or heavy background noise. Additionally, highly technical podcasts may lose some precision in their summaries due to domain- specific terminology not being handled adequately.

Despite these limitations, our project opens avenues for future research and development. Integrating speaker diarization, emotion detection, and multi-lingual support could enhance the robustness and versatility of the summarizer. Furthermore, real-time summarization or keyword-based summarization are promising directions for future improvement.

## VI. CONCLUSION

Our Podcast Summarizer Project demonstrates the effective integration of modern technologies to address the growing demand for digestible audio content. By combining a Flask backend with a React.js frontend, the system provides a modular and scalable architecture. Core features like speech-to-text transcription using Whisper, speaker diarization, summarization via BART/Groq APIs, sentiment analysis, translation, and text-to-speech conversion are seamlessly orchestrated to deliver an intuitive and intelligent platform. With PostgreSQL handling structured data and Docker streamlining deployment, the application ensures reliability, performance, and ease of maintenance. This project stands as a comprehensive solution for podcast consumers and researchers seeking fast and focused audio content insights.

## ACKNOWLEDGEMENT

We extend our heartfelt gratitude to our esteemed Director, Dr. Mohiuddin Ahmed, and our In- Charge Principal Dr. Ganesh Kame at M.H. Saboo Siddik College of Engineering for providing the facilities, unwavering support, and an excellent environment essential for fulfilling our project requirements. Our deepest thanks go to our internal project guide, Prof. Dr Lutful Islam for her invaluable guidance and willingness to share her extensive knowledge. Her continuous support and insights enabled us to gain a profound understanding of the project and successfully complete it. Additionally, we are grateful to the staff of the Computer Department, particularly the Laboratory staff, for granting us access to the labs and providing the resources required for the project. Lastly, we acknowledge the guidance of other supervisors and project mentors. Their constructive feedback and helpful suggestions greatly enhanced our presentation skills and overall project execution.

## REFERENCES

[1] Chujie Zheng, Harry Jiannan Wang, Kunpeng Zhang, and Ling Fan, “A baseline analysis for podcast abstractive summarization,” PodRecs: The Workshop on Podcast Recommendations, September 25 2020.

[2] Damiano Spina, Johanne R. Trippas, Lawrence Cavedon, and Mark Sanderson, “Extracting audio summaries to support effective spoken document search,” Journal of the Association for Information Science and Technology, September 16 2017.

[3] Nikhil Garg, Benoit Favre, Korbinian Reidhammer, and Dilek Hakkani-Tur, “Clusterrank: a graph based method for meeting summarization,” ¨ in Tenth Annual Conference of the International Speech Communication Association, February 18 2009.

[4] Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy, “Deep reinforcement learning for sequence-to-sequence models,” IEEE Transactions on Neural Networks and Learning Systems, vol. 31, July 7 2019.

[5] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou, “Summarunner: A recurrent neural network based sequence model for extractive summarization of documents,” in 31st AAAI

Conference on Artificial Intelligence, San Francisco, CA, USA, February 4 2017.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, June 2 2019.

[7] Chin-Yew Lin, “ROUGE: A package for automatic evaluation of summaries,” in Workshop on Text Summarization Branches Out, Barcelona, Spain, July 26 2004.

[8] A. Vartakavi, A. Garg and Z. Rafii, "Audio Summarization for Podcasts," 29th European Signal Processing Conference (EUSIPCO), 2021 27.

[9] R. K. Yadav, R. Bharti, R. Nagar and S. Kumar, "A Model For Recapitulating Audio Messages Using Machine Learning," International Conference for Emerging Technology (INCET), 2020.

[10] P. G. Shambharkar and R. Goel, "Analysis of Real Time Video Summarization using Subtitles," International Conference on Industrial Electronics Research and Applications (ICIERA), 2021.

[11] OpenAI, “Whisper: Robust Speech Recognition via Large-Scale Weak Supervision,” 2022. [Online]. Available: https://openai.com/research/whisper