A Podcast Summarization System with Translation and PDF Report Generation

Aditi Nikam, Shruti Balekundri, Amna Bijapuri, Vishal Kadalagi and Savita Bagewadi
Department of Computer Science and Engineering
KLE Dr. M S Sheshgiri College of Engineering and Technology, Belagavi, India
02fe23bcs416@kletech.ac.in, 02fe23bcs425@kletech.ac.in, 02fe23bcs427@kletech.ac.in,
02fe23bcs431@kletech.ac.in and savitabagewadi.mss@kletech.ac.in

***Abstract*—YouTube carries with it a huge library of insightful material such as podcasts, interviews, lectures, and educative discussions. But with time limitations, consumers tend to struggle with watching lengthy videos all the way through. This smart summarization system offers a hassle-free solution with the ability to allow users to input a link of a YouTube video and get a readable summary. Upon submission, the system extracts the audio from the video, transcribes the speech to text using OpenAI's Whisper model, and then condenses the transcription into a brief, insightful summary using state-of-the-art natural language processing (NLP) techniques. This significantly reduces the effort and time required to absorb the core ideas of long videos.**

For improved accessibility and user experience, the platform has multilingual translation support, enabling both the transcribed content and the final summary to be translated into different global languages. This renders the tool particularly useful for non-native users and global consumers wanting localized comprehension of content. In addition, users can download the translated or original summary in a clean PDF format for offline reading and sharing. The platform combines secure authentication via Google Firebase, providing personalized and safe access to features. Developed with a ReactJS frontend for user-friendly interface and a Flask backend for controlling audio processing and summarization jobs, the tool represents a cost-effective, accessible, and smart means of transforming long YouTube content into concise, meaningful insights.

Keywords: Youtube Video Summarization, Speech-to-Text Transcription, Text Translation, Text Summarization, PDF Export, Google Firebase Authentication, Flask, ReactJs

I. Introduction

With the rapid pace of the modern digital age, podcasts and videos are now among the best and easiest means to acquire information, learn, and keep up with many subjects. YouTube and other apps featuring podcasts have a massive collection of lengthy content, ranging from education and technology to entertainment and news. But the duration of most podcasts and videos usually makes it challenging for busy users to spend the required time listening or viewing in its entirety. This calls for intelligent tools that can extract key information quickly and efficiently, allowing users to obtain the most critical insights without spending hours listening or viewing the entire episode.

This system is able to fulfill this requirement in a highly effective manner by automatically transcribing spoken content from podcasts and YouTube videos into readable, comprehendible text using state-of-the-art speech-to-text transcription technology. Transcription then instantly creates brief, coherent summaries that identify the major concepts and main points, allowing users to understand the key information in a fraction of the time. Additionally, it allows translation of the transcribed text and the summary to various languages, making the content universally accessible to a diverse global audience. Automatic generation of summaries saves time and effort and enables users to easily review or share valuable information from long audio or video content.

Besides its robust content processing functionalities, the platform also offers secure user access through Google Firebase authentication to ensure privacy and personalized interactions. Users can conveniently export and download generated summaries in PDF format for offline reading, note-taking, or future use. By marrying transcription, summarization, translation, and secure user management into a straightforward, intuitive interface, this solution changes the way podcasts and videos are consumed—giving users the power to remain knowledgeable, learn effectively, and keep up with their time in today's knowledge-rich world.

In addition, the tool is user-centric, providing an easy interface wherein the user just gives a YouTube link to automatically start the extraction and summarization process. This removes any technical obstacles or complex steps, making it convenient for users from all ages and walks of life. The use of contemporary natural language processing and machine learning algorithms guarantees summaries not just quickly but also contextually appropriate and accurate, improving understanding. Targeting podcast content, this site serves a specific but increasingly popular niche of media consumption, which renders it highly valuable to students, businesspeople, and general listeners alike.

The application also introduces multi-language support, catering to non-English speakers and fostering inclusivity by making content accessible across linguistic boundaries. This is particularly useful in educational settings, where students from diverse backgrounds can benefit from translated summaries in their native languages. Furthermore, the system is designed to be extensible, paving the way for potential integration with calendar tools, bookmarking, or content recommendation engines to enhance productivity and user engagement.

Looking ahead, the platform opens up avenues for more advanced features such as voice-based search, personalized content filtering, and summary customization based on user interests or profession. With AI continuing to evolve, the summarizer could also provide sentiment analysis or speaker identification, further enriching the user experience. Overall, the system aims not just to simplify content consumption, but to redefine it, offering a smart, adaptive, and inclusive solution for managing long-form digital audio-visual information.

II. Literature Survey

Recent advancements in Natural Language Processing (NLP) and Artificial Intelligence (AI) have significantly influenced the field of multimedia summarization, particularly for video and audio content. Several notable works have explored diverse approaches to automatic summarization for YouTube videos and podcasts.

Vartakavi and Garg's (2020) study introduced PodSumm, a podcast audio summarization technique that uses a refined PreSumm model based on BERT to combine extractive text summarization and speech-to-text transcription [1]. Using AWS Transcribe, the architecture transcribes podcast episodes, selects important sentences to piece together audio snippets that make up the summary, and processes the transcripts to find sentence-level time stamps. The model is limited by the size of the dataset, errors in sentence segmentation, and reliance on the quality of the transcription, even though it achieved high ROUGE scores (ROUGE-1/2/L: 0.63/0.53/0.63).

The authors of a 2023 project called "YouTube/Podcast Summarizer using AI" developed a system for creating abstractive summaries of YouTube video transcripts by utilizing Hugging Face models and LLMs like Google Gemini Pro. The system was implemented using Streamlit and used heatmap visualizations and cosine similarity metrics. The system's shortcomings included transcript noise, LLM token restrictions, real-time scalability, and generalization problems across content types, despite its excellent summarization performance and user accessibility [2].

In their paper "YouTube Video Summarizer using NLP: A Review," Singh et al. (2023) gave a thorough review of the various NLP approaches currently used for YouTube summarization, such as TF-IDF, BERT, topic modeling, and extractive techniques like LexRank. Their research highlighted the efficiency of supervised learning models for summarization, but also pointed out drawbacks, including high processing costs, difficulties processing noisy or unstructured text, and the lack of semantic depth of a simpler model [3].

Similarly, a useful, real-time summarization method was investigated in the paper "Audio Summarization in Real Time for Podcasts, Speeches and Audiobooks" (2023). The authors created a Streamlit-based web application to give users access to transcripted and summarized podcast content using Python, the Listen Notes API for podcast retrieval, and AssemblyAI for transcription and summarization. This system was limited by its reliance on third-party APIs, small dataset size, and decreased adaptability to a variety of audio inputs, despite its responsiveness and ease of deployment [4].

The PoliTO system (2021), which was showcased at TREC, adopted a multimodal strategy by integrating textual embeddings (from MPNet) and audio features (from OpenSmile) into a deep learning architecture. After employing a regression model to predict sentence relevance, it produced audio summaries through audio clipping and textual summaries using Longformer. NIST's manual evaluation revealed that it performed better than baselines in several quality metrics, but it also displayed problems such as text generation hallucinations and the sporadic omission of important personal names due to model limitations [5].

A significant advancement in multimodal podcast summarization was presented by Vaiani et al. (2022) in their paper "Leveraging Multimodal Content for Podcast Summarization". They proposed MATeR (Multimodal Audio-Text Regressor), an end-to-end deep learning architecture that fuses BERT-based textual and Wav2Vec2-based acoustic features to generate extractive text-audio summaries. Unlike prior methods that focus on either text or speech alone, MATeR jointly processes and aligns both modalities using a multimodal fusion network trained to optimize relevance prediction. Their results on a real-world Spotify dataset demonstrated significant improvements in ROUGE and SBERT scores over both monomodal and existing extractive summarizers. While MATeR achieved superior performance and robustness in handling podcast heterogeneity, its limitations included sensitivity to poorly synchronized inputs and the reliance on pre-annotated podcast descriptions for supervised learning [6].

Further enriching this body of work, Gothankar et al. (2022) proposed a dual-mode summarization system titled "Extractive Text and Video Summarization using TF-IDF Algorithm". Their approach utilizes the Term Frequency–Inverse Document Frequency (TF-IDF) method to extract key sentences from both text-based documents and video transcripts. The system supports input formats such as raw text, articles, documents (PDF, DOCX, TXT), and MP4 videos, using Google Cloud Speech-to-Text API to convert video content into text.

The implementation includes a ReactJS-based frontend and a Python-Flask backend, providing an intuitive user interface and a modular architecture. The algorithm ranks sentences based on TF-IDF scores, selects the most informative ones, and reorders them to maintain coherence. Although the system performs well in generating concise summaries, it struggles with semantic understanding, lacks contextual awareness, and may not adapt well to stylistically diverse or informal content. Furthermore, the approach relies heavily on the accuracy of speech transcription, which can impact summary quality in noisy audio scenarios [7].

Together, these studies highlight persistent issues in the field, especially the requirement for precise transcription, strong multimodal integration, and scalability, none of which have been adequately met by current solutions.

TABLE I
LITERATURE SURVEY

Author	Objective	Methods	Results	Limitation	Advantages
Vartakavi Garg et al. [1]	To generate podcast audio summaries using text guidance.	ASR via AWS Transcribe; extractive summarization using fine-tuned PreSumm (BERT); audio segments stitched from selected sentences.	ROUGE-1/2/L: 0.63/0.53/0.63; improved after fine-tuning and augmentation	Small dataset; ASR errors; sentence segmentation issues; no multimodal learning	Maintains spoken tone; incorporates text and audio; uses task-specific data
Kohli et al. [2]	To summarize YouTube and podcast content using LLMs	Google Gemini Pro, Hugging Face models; cosine similarity; Streamlit UI; multivideo support	High similarity scores with transcripts; Gemini Pro performed best	LLM token limits; transcript noise; API dependence; not real-time scalable	LLM integration; parallel processing; UI with heatmap visualizations
Singh et al. (2023) et al. [3]	To review NLP methods for video summarization	Comparative study of TF-IDF, BERT, extractive/abstractive methods, topic modeling, sentiment analysis	Identified supervised models as most accurate; NLP techniques effective for structured content	High computational cost; weak in semantic nuance; limited multilingual capability	Broad overview of methods; identifies trends and research gaps.
Liya B.S. et al. (2023) [4]	Real Time audio summarization for podcast, speech, audio book.	Listen Notes and AssemblyAI APIs; Streamlit app.	System deployed with realtime support for transcription.	Depends on APIs; limited generalization; small dataset	Real-time use; simple UI; text/audio output.
Vaiani et al. (2021) [5]	Multimodal podcast summarization for TREC	MPNet + OpenSmile features; multimodal model with Longformer	Top text, 2nd audio at TREC; beat baseline in eval	Hallucinations in generated text; misses names/titles; model complexity	Deep multimodal integration; strong performance in benchmark task
Vaiani et al. (2022)	Improve podcast summarization using multimodal inputs	MATeR: BERT (text) + Wav2Vec2 (audio) + multimodal fusion network	Achieved high ROUGE and SBERT scores on real Spotify data	Sensitive to poorly synced inputs; needs pre-annotated descriptions	Robust multimodal summarization; outperforms monomodal models
Gothankar et al. (2022)	Summarize text and video using TF-IDF	TF-IDF, POS tagging, Google Speech API, ReactJS & Flask	Accurate summaries for text & video with minimal redundancy	No semantic understanding; depends on transcription quality	Supports multiple formats; lightweight and easy to use

III. PROPOSED METHODOLOGY

A. Podcast Video Input Capture

The process starts by obtaining a podcast video, typically from a public video-sharing platform such as YouTube. The user-provided video link is processed through a downloading mechanism that uses video scraping libraries. Two Python packages—`yt_dlp` and `pytube`—are employed at this stage.

`yt_dlp` is a high-performance and reliable downloader specifically designed for handling dynamic streaming URLs, while `pytube` offers a simpler, more Pythonic interface and acts as a fallback. These tools ensure successful downloading and local storage of the video files.

To manage unique file naming and avoid overwriting, the `uuid` module is used to assign distinct identifiers to each file. File and directory operations are handled by the `os` module, ensuring organized temporary storage. Additionally, the `re` module is occasionally utilized to clean URLs by removing parameters such as timestamps.

B. Audio Extraction from Video

Once the video is safely downloaded and saved, the second step is to separate the audio stream from the video. This is done with a powerful multimedia processing library that can read the video files and harvest their audio streams with good fidelity. The library employed for this function does video decoding and outputs the resulting audio to a normal format, usually WAV, appropriate for speech recognition purposes. After finishing the audio extraction, the video file is closed to free system resources. A delay of a short duration is inserted through a time management module to allow for the writing process to finish successfully prior to proceeding to transcription. The obtained audio file is then saved in another folder for simplicity and easy reference in the subsequent step.

C. Speech to Text Transcription

Upon audio extraction, the separated audio stream is then subjected to a speech recognition pipeline to achieve the textual transcript of the spoken content. This is achieved with the aid of a state-of-the-art speech-to-text model utilizing the whisper module by OpenAI. This model is famous for its multilingual nature and high transcription accuracy, particularly under noisy conditions. For the sake of better performance and hardware usage, the torch module is employed as the backend to decide whether to execute computation on the CPU or GPU. The model automatically transcribes and segment the audio and outputs a coherent transcript for the spoken words.

This transcript is used as the basis for further natural language processing processes.

D. Text Summarization

The raw transcript could have lengthy, descriptive speech that is not necessarily effective to read through in full. Therefore, a summarization process is added to condense the transcript into a brief informative report. For this, transformers library from HuggingFace is utilized using pretrained language models with abstractive summarization capabilities. These models apply sophisticated encoder-decoder models to detect the salient content and produce reduced, human-readable abstracts that maintain the essence of the message. This model utilizes the direct transcribed text and creates outputs that are both grammatically sound and semantically meaningful, greatly expanding accessibility and readability for the users.

E. Optional Text Translation

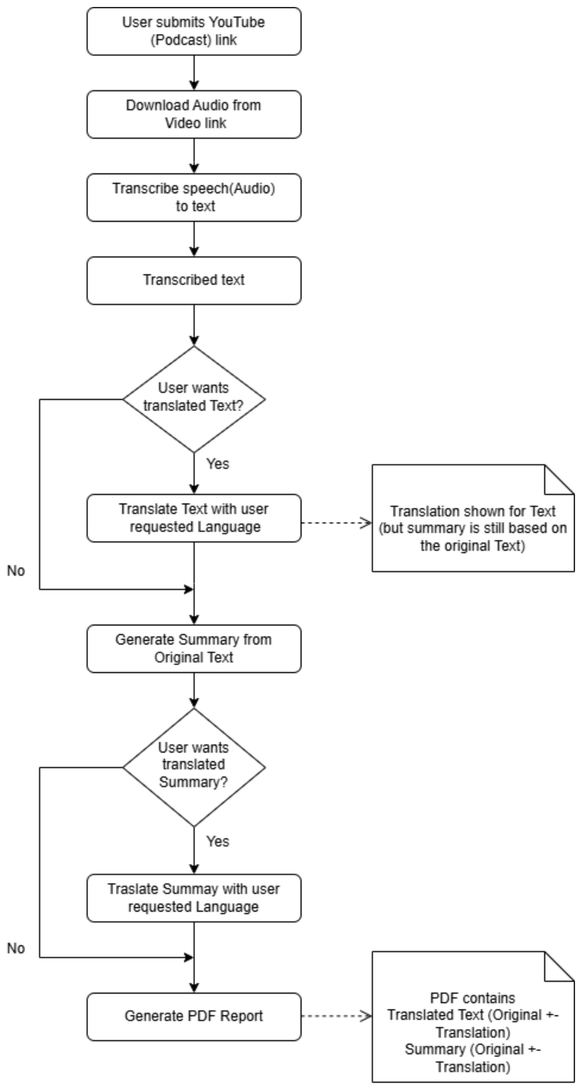
For serving a multi-language audience, the system has an optional googletrans module for translating the extracted text and summary into other target languages. The feature is based on the googletrans module, which uses the Google Translate Non-Official API to carry out neural machine translation. Translated summary preserves the semantic accuracy of the original English summary and extends the application of the system for users preferring other languages. This multi-lingual facility is especially helpful for international podcast listeners who can be less than proficient in the source language of the podcast.

F. PDF Report Generation

Once transcribed, summarized, and optionally translated, the system outputs a formatted PDF report with all processed material. This encompasses the initial transcript, summary, and translated material, if chosen. The PDF is rendered on the client side with the html2pdf.js library, which uses html2canvas for HTML content rendering and jsPDF for presenting the output as a downloadable A4 PDF file. This provides a neat, styled document users can save offline. The os module in the backend takes care of handling temporary files and directories during this operation.

G. Web API and Client Integration

All these functionalities are encapsulated and made available through a light web server framework. The Flask library is utilized as the foundation of the backend infrastructure, providing clearly defined endpoints for video upload, transcription, summarization, and translation. In order to enable easy communication between the frontend client (e.g., a React) and the Flask server, Flask-CORS is utilized to manage Cross-Origin Resource Sharing policy. This makes it possible for applications based on a browser to communicate securely and efficiently with the backend services. The logging module is also used to monitor system activity and debug bugs to enhance overall reliability and maintainability.



User submits YouTube (Podcast) link
Download Audio from Video link
Transcribe speech(Audio) to text
Transcribed text
User wants translated Text?
Yes
No
Translate Text with user requested Language
Translation shown for Text (but summary is still based on the original Text)
Generate Summary from Original Text
User wants translated Summary?
Yes
No
Traslate Summay with user requested Language
Generate PDF Report
PDF contains Translated Text (Original +- Translation) Summary (Original +- Translation)

Fig. 1. System Workflow

This system converts a video or podcast link into short, multi-language summaries. The process starts with extracting the audio from the link and transcript it to text, that the user can have translated to their preferred language while the original text is preserved for proper processing. The original transcript is then summarized to give the user a short, meaningful summary of the content.
Additionally to summarization, users can translate the summary into a selected target language. Finally, the system produces a PDF report which has the original transcript, summary and translated versions.

IV. Results and Discussion

A. Audio Extraction Timeliness:

Audio extraction time was completely inconsistent across examples based on video size and video format. The shortest video (2:46 min) took 10.72 seconds to extract; conversely, the longest video (43:38 min) took 31.39 seconds to extract.

TABLE II
RESULTS

Video Link	Duration (min:sec)	Audio Extraction Time	Transcription Time	Total Time	Transcript Translation (Lang : Time)	Summary Translation (Lang : Time)
https://youtu.be/-3g3CbFq-YA	2:46	10.72 s	12.32 s	23.04 s	Kannada: 3.15 s	Hindi: 2.78 s
https://youtu.be/pWsQtektN-o	6:51	9.25 s	29.81 s	39.06 s	Japanese: 6.24 s	Portuguese: 2.99 s
https://youtu.be/ugMR0MfVmNs	13:09	9.74 s	54.04 s	63.78 s	Punjabi: 8.49 s	French: 2.58 s
https://youtu.be/_2YipyFJ_6E	18:51	11.86 s	140.17 s	152.03 s	German: 11.21 s	Portuguese: 2.35 s
https://youtu.be/Jqo1BM0TfBY	41:50	13.93 s	265.96 s	279.89 s	Malayalam: 18.24 s	Odia: 1.93 s
https://youtu.be/xAt1xcC6qfM	43:38	31.39 s	141.39 s	172.78 s	Bengali: 15.44 s	Arabic: 2.30 s

It is notable that extraction time was not linearly scaled to video duration of which indicates the 'yt-dlp' effectively buffered and optimized streaming.

- Calculation example (efficiency rate):
  For the 41:50 min video:
  Duration = 2510 sec
  Extraction time = 13.93 s
  Efficiency ratio = 2510 / 13.93 = 180.21 $\rightarrow$ This indicates that audio extraction is at a rate of =180× faster than real time playback.

B. Transcription Processing Duration:

Transcription is the most computationally demanding phase. The 41:50 video took 265.96 seconds - the longest transcription we have completed. This reflects Whisper's performance with the transcription at a speed ratio of:
Speed ratio = 2510 / 265.96 = 9.43× real-time

- The 18:51 video (1131 sec) took 140.17 seconds:
  Speed ratio = 1131 / 140.17 = 8.07×.

The consistent 8–10× real-time transcription ratio across all videos demonstrates how well the scalably belongs to (or extends to) the Whisper model.

C. Summary Performance:

Summary generation times varied from 12.35 seconds to 35.05 seconds depending on input length and complexity. The longest summary generation, 35.05 seconds, was for the 13:09 minute video. Even the largest video (41:50 minute) took only 26.65 seconds due to the input truncation to 1000 characters for summary input; showing that summarization time was more dependent on token input than video length.

D. Multilingual Translation Results:

We translated both transcripts and summaries. Transcript translations averaged 10–18 seconds (for example, 18.24 seconds for Malayalam), while summary translations were all significantly faster (typically < 3 s). This is due to the greater size of transcripts vs. summaries.

- Summary of selected translations:
  Transcript (long): Malayalam (18.24 s), Bengali (15.44 s)
  Summary (short): Odia (1.93 s), Arabic (2.30 s), Hindi (2.78 s)

This suggests that summary translations will be more effective for almost real-time deployment situations because of the lower token size.

E. Most Computationally Intensive Example:

The video at https://youtu.be/Jqo1BM0TfBY, which is 41:50 min long turned out to be the most resource demanding video. Analysis:

- Audio Extraction: 13.93 s
- Transcription: 265.96 s
- Total: 279.89 s (4 min 40 s)
- Summarization: 26.65 s
- Translation: Malayalam (18.24 s), Odia (1.93 s)

This workflow took 279.89 + 26.65 + 18.24 + 1.93 = 326.71 seconds (approximately 5 min 27 s) to complete.

Summary of Overall Trends

- Overall extraction time is relatively small compared to video duration (<1.5% on average).
- Transcription is the dominant contributor to total processing time (∼70–85%).
- Summarization and translation are lightweight, enabling quick turnaround.
- Whisper and transformer-based summarizers scale linearly with input length, but are limited by a runtime cap (e.g., `max_length`).
- Translation time depends primarily on the number of input tokens, not the target language.

V. Conclusion and Future Work

This system demonstrates a new way of quickly reworking audio and video content into short summaries and providing quick access to summaries in different languages. The system makes it convenient and simple for the user to take spoken content and transcribe audio content into text via transcription, summarize the text content, and then translate the transcripts and summaries into various local and international languages. The processing time, while the duration of videos and languages were variable, was exceptionally fast and the processing times were extremely consistent. The total processing times were measured in seconds or a few minutes, which saves users time and shortens the accessibility gap for individuals wanting to get access to content in their own language, or in text format. This system supplies the potential for valuing: convenience for the user, a focus on language diversity, and simplifies and localizes huge amounts of multimedia information in only moments via a straightforward, user-friendly interface.

Moving forward, there are many exciting possibilities to make the system even more robust and intelligent. Future enhancements might involve employing visual media such as video frames or facial expressions to develop summaries that are even more nuanced and contextually or emotionally responsive. Real-time enhancements would enable summarization of live content, such as webinars, lectures, or podcasts in progress. Employing summarization models based on newer, larger language models could improve the quality and fluency of summaries, making them more human-like and accessible. In addition, it might be valuable to develop mobile platform support along with the ability to set a summary preference based on user behaviour or expressed interest to make this tool a useful tool for rapid content consumption in today's ever evolving global and multilingual environment.

REFERENCES

[1] A. Vartakavi and A. Garg, "Podsumm–podcast audio summarization," *arXiv preprint arXiv:2009.10315*, 2020.

[2] M. Kohli, A. Choudhary, and D. Gupta, "Youtube, podcast summarizer using ai," 2024.

[3] Y. Singh, R. Kumar, S. Kabdal, and P. Upadhyay, "Youtube video summarizer using nlp: A review," *International Journal of Performability Engineering*, vol. 19, no. 12, p. 817, 2023.

[4] K. Haripriya, S. H. Thabasum, and R. B. Saraswathi, "Audio summarization in real time for podcast, speeches and audiobooks," 2023.

[5] L. Vaiani, M. La Quatra, L. Cagliero, P. Garza *et al.*, "Polito at trec 2021 podcast summarization track." in *TREC*, 2021.

[6] L. Vaiani, M. L. Quatra, L. Cagliero, and P. Garza, "Leveraging multimodal content for podcast summarization," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 863–870. [Online]. Available: https://doi.org/10.1145/3477314.3507106

[7] A. Gothankar, L. Gupta, N. Bisht, S. Nehe, and P. Bansode, "Extractive text and video summarization using tf-idf algorithm," *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, vol. 10, no. 3, pp. 927–932, 2022.