TubeTalk: A Langchain Based QnA Retrieval System For Youtube Videos

Mark Lopes
Third Year Student
Department of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)
crce.9913.ce@gmail.com

Vivian Ludrick
Third Year Student
Department of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)
crce.9914.ce@gmail.com

Jonathan Gomes
Third Year Student
Department of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)
crce.9900.ce@gmail.com

Department of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)

Department of Electronics & Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)

Department of Computer Engineering
Fr. Conceicao Rodrigues College Of Engineering, Bandra(W)

Abstract: With the increasing demand for video accessibility and knowledge extraction, innovative tools are required to analyze and interact with video content effectively. YouTube hosts a vast amount of information, but extracting meaningful insights from videos requires advanced processing techniques. This paper presents a web-based system that enhances video understanding by transcribing and analyzing YouTube video content using natural language processing and vector-based storage. The system generates accurate transcripts, enabling users to engage with video content through an interactive question-answering interface. Additionally, it provides automated quizzes to assess comprehension, fostering deeper learning experiences. The platform is designed for researchers, students, and casual viewers, offering an intuitive way to explore video content efficiently. The results highlight the effectiveness of integrating NLP algorithms with user-centric accessibility features, demonstrating the potential for improved engagement and knowledge retention in video-based learning..

KEYWORDS: ***YouTube video analysis, natural language processing, video transcription, question-answering system, interactive learning.***

1. INTRODUCTION

The growing need for video accessibility and knowledge extraction has led to advancements in tools for analyzing and interacting with video content. YouTube serves as a vast repository of information, but efficiently extracting insights from videos requires sophisticated processing techniques and interactive features. Traditional methods of video analysis rely on manual effort, making it challenging to retrieve specific information or assess comprehension effectively.

This project aims to develop an online system that enables users to explore and engage with YouTube video transcripts using natural language processing (NLP) and vector-based storage. The system processes audio to generate accurate transcriptions, which are then stored in an efficient vector format to support advanced search and retrieval. Additionally, it features a question-answering system that allows users to obtain precise answers based on video content. To enhance learning outcomes, the platform also generates interactive quizzes, enabling users to assess their comprehension of the material.

The key objectives of the project include automated transcription, vector-based transcript storage for efficient retrieval, an interactive interface for visualization and exploration, and a robust question-answering mechanism. By integrating cutting-edge NLP techniques with an accessible user interface, this project provides a valuable tool for researchers, students, and general viewers. It enhances knowledge retention and engagement, fostering deeper learning and more effective interaction with video content.

2. LITERATURE REVIEW

The growing interest in intelligent systems for video content analysis has led to advanced techniques for video transcript generation, semantic retrieval, and question answering. This section outlines the methodologies leveraged in the development of the TubeTalk system, emphasizing dense retrieval, semantic similarity, transcript synthesis, and large language model (LLM) integration.

. Dense Passage Retrieval (DPR) has been employed for efficient open-domain question answering, enhancing the retrieval of relevant content from YouTube transcripts [1]. DPR utilizes dense vector representations to improve semantic understanding during search, significantly outperforming traditional sparse methods. However, it demands substantial training data and high-performance computing resources for effective implementation.

To support large-scale similarity searches within transcript datasets, the **Facebook AI Similarity Search (FAISS)** library was integrated [2]. FAISS enables high-speed retrieval of semantically similar content across high-dimensional embedding spaces, a crucial component for matching user queries with transcript passages. Although FAISS is highly efficient, it poses scalability challenges in resource-constrained environments.

To improve semantic matching, **sentence-level embeddings** were utilized in conjunction with FAISS [3]. This combination allows for context-aware transcript search and enhances the system's ability to deliver accurate responses to user queries. Effectiveness, however, is dependent on the quality and contextual richness of the embeddings.

For generating high-quality transcripts from YouTube videos, **deep learning-based transcript synthesis** methods were applied [4]. These approaches ensure accuracy in transcriptions but require considerable time and computation, particularly when processing lengthy or numerous videos.

The system also incorporates **speech recognition and text summarization techniques** to produce concise video summaries [5]. This hybrid approach balances informativeness with brevity but may lose contextual depth in complex discussions or lengthy videos.

To further refine content retrieval, **embedding-based techniques** drawn from Facebook's search system were employed [6]. These methods improve the relevance of transcript segments presented to the user, though their performance varies based on the embedding model used.

Fundamental to the embedding process is the application of well-established **word embedding models** like Word2Vec and GloVe [7]. These models provide a basis for text vectorization, enabling semantic search across diverse video topics. Nonetheless, their limited context window may impact performance in nuanced or domain-specific content.

User personalization in transcript generation was explored through deep learning models that factor in user interaction data and preferences [8]. These models allow tailored summarization and Q&A experiences, albeit at the cost of requiring granular user data.

Advanced architectures like **VX2TEXT** utilize end-to-end multimodal learning to generate accurate and context-rich video descriptions [9]. These models, while effective, introduce significant architectural and computational complexity.

For video classification and organization, the project draws from **text-only multimodal approaches**, enabling classification without direct video processing [10]. This method reduces computational load but may omit important visual cues.

To orchestrate question processing and response generation, **LangChain** was employed alongside **Large Language Models (LLMs)** [11]. This combination streamlines query parsing, improves response relevance, and supports dynamic Q&A. Despite these benefits, such systems are sensitive to data quality and often require extensive fine-tuning.

Finally, automation capabilities within the system are inspired by **LangChain-based customer service models** [12], enabling real-time interaction and scalable support. While automation enhances efficiency, it may struggle with highly ambiguous or nuanced user questions.

3. METHODOLOGY

The TubeTalk system is designed to transform passive YouTube video consumption into an interactive, educational experience by combining automated transcription, question-answering, and quiz generation. The system architecture integrates state-of-the-art retrieval and embedding techniques with a user-friendly web interface for seamless interaction.

3.1 Data Acquisition and Transcription

The system initiates upon receiving a YouTube video URL from the user or directly through the TubeTalk browser extension, which automatically captures the link from the current tab. The video's audio is extracted using the yt-dlp library. This ensures efficient and high-quality audio retrieval. The audio is then transcribed into text using the AssemblyAI API, enabling downstream analysis.

3.2 Text Preprocessing and Embedding

The transcribed text is segmented using the RecursiveCharacterTextSplitter, which breaks the text into coherent, contextually meaningful chunks. These chunks are embedded into vector representations using GoogleGenerativeAIEmbeddings. The vectors are then indexed with FAISS (Facebook AI Similarity Search) to support fast and accurate similarity-based search operations.

3.3 Semantic Information Retrieval and Question Answering

Upon user input, such as a natural language query, the system employs a **RetrievalQA Chain** architecture. It uses FAISS to identify and retrieve transcript segments with high semantic similarity to the query. The retrieved context is then passed to **Google**

Generative AI, which synthesizes a concise, contextually relevant answer. This two-step approach ensures that the response generation is both accurate and grounded in the actual video content.

This component leverages principles from **Dense Passage Retrieval (DPR)** [1] and embedding-based retrieval systems [3][6], demonstrating significant improvements in open-domain QA accuracy through dense vector similarity matching.

3.4 Comprehension Assessment via Quiz Generation

To enhance the learning experience, the system generates a multiple-choice quiz based on the video transcript. This feature enables users to assess their understanding of the content interactively. The quiz questions are designed using AI-driven techniques to ensure relevance and coverage of key topics discussed in the video.

3.5 Visualization and User Interaction

The system uses Next.js for the frontend and Python for the backend, offering a clean and interactive interface. Users can view transcripts, ask questions, and take quizzes seamlessly. The Python backend handles tasks like transcription, text processing, and retrieval through API routes, while the Next.js frontend ensures a smooth and responsive user experience. This architecture supports scalability and future feature enhancements.

4. COMPONENT SELECTION AND SYSTEM PERFORMANCE

The performance and accuracy of the TubeTalk system heavily rely on the appropriate selection of AI models and libraries. Each component—from transcription to Q&A—was carefully chosen for its efficiency, accuracy, and compatibility with real-time interaction needs.

Table 4.1: Component Selection Overview

Component	**Tool/Library Used**	**Purpose**	**Performance**
Audio Extraction	yt-dlp	Downloads high-quality audio from YouTube	Fast and reliable audio extraction
Transcription	AssemblyAI	Converts audio to text	High accuracy; supports various languages
Text Chunking	RecursiveCharacterTextSplitter	Breaks transcript into manageable chunks	Maintains contextual coherence
Embedding Model	GoogleGenerativeAIEmbeddings	Converts text chunks into vectors	Optimized for semantic search
Vector Storage	FAISS	Stores and searches vector data	Fast similarity-based retrieval
Q&A Generation	GoogleGenerativeAI	Generates answers based on transcript content	Produces contextually relevant responses

System Performance Factors

Several factors affect the performance of the TubeTalk system. These include transcription accuracy, chunking strategy, embedding quality, and model response time. The effectiveness of the system depends on how well these components work together to deliver fast and relevant responses to user queries.

- Transcription Accuracy: Clearer audio and minimal background noise improve the accuracy of the AssemblyAI transcription model, forming a reliable base for further processing.
- Model Latency: The response speed of the Q&A system depends on how quickly relevant chunks are retrieved and processed by the language model. Fast embeddings and a lightweight backend help maintain smooth performance.
- Chunking Strategy: Text is divided using RecursiveCharacterTextSplitter to maintain context. Proper chunk sizes ensure better retrieval and fewer irrelevant answers.
- Embedding Quality: Google Generative AI embeddings enable precise similarity matching. High-quality embeddings directly impact the relevance of answers retrieved from FAISS.

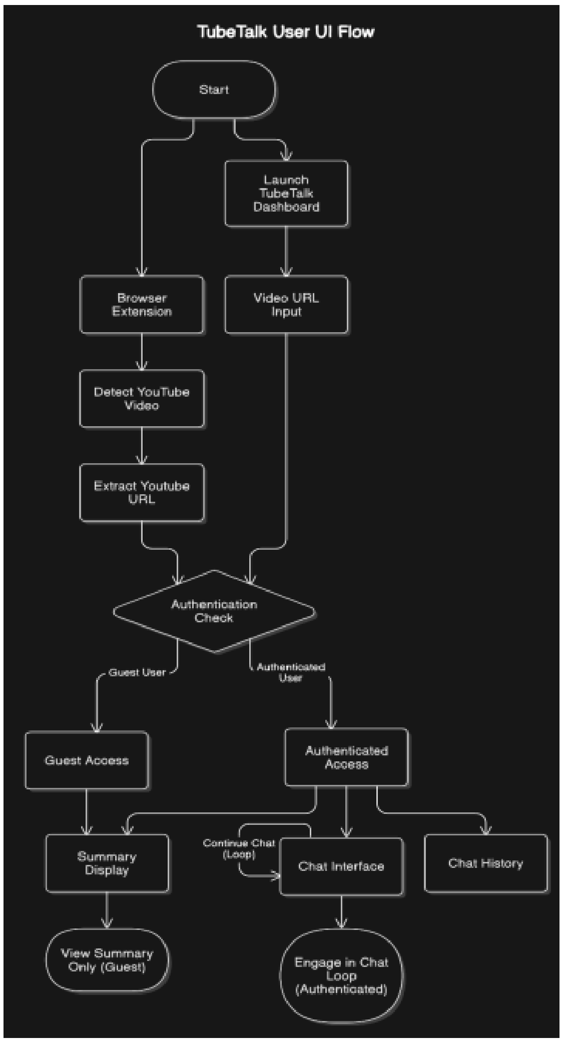

Fig 5.1: System design of Map Generation

5. SYSTEM DESIGN

The following diagram provides the proposed design of the system:

- The user starts by accessing the application through a Chrome Extension or Web Application.
- The user registers or signs in to their account using the authentication deck.
- Upon successful authentication, the system enables interaction with YouTube content.
- The user uploads a YouTube video or provides a link to initiate the transcript extraction process.
- The system processes and extracts the transcript from the video, displaying a summary for quick reference.
- The chat interface allows users to ask questions related to the video transcript, receiving AI-generated responses.
- All processed transcripts, summaries, and chat interactions are stored in a backend database.
- The system provides details on relevant video content sources.

The process begins when a user accesses the platform to extract meaningful insights from YouTube videos. Whether a new or returning user, the journey starts with registration or signing in. This ensures a personalized experience, allowing users to save their progress and revisit their data whenever needed.

I. REGISTER OR SIGN IN

To proceed, the user must either register a new account or sign in with existing credentials. New users can create an account by providing basic information. This ensures that all users have a personalized and secure account, where they can save their data and access it later. If any issues arise during registration or login, the system provides helpful prompts to guide the user through resolving them.

II. INPUT LOCATION

After authentication, users can interact with the system by uploading YouTube videos or providing links. This initiates the transcript extraction process, enabling further analysis. The user submits a video link or file to generate a transcript, and the system then presents a summary of the video transcript, making it easier for users to grasp key points. Through the chat interface, users can interact with the transcript by asking questions and receiving AI-generated responses.

III. CONTENT INTERACTION AND TRANSCRIPT EXTRACTION

After authentication, users can interact with the system by uploading YouTube videos or providing links. This initiates the transcript extraction process, enabling further analysis. The user submits a video link or file to generate a transcript, and the system then presents a summary of the video transcript, making it easier for users to grasp key points. Through the chat interface, users can interact with the transcript by asking questions and receiving AI-generated responses.

.

6. Algorithms

Audio Extraction Algorithm: This algorithm uses yt_dlp to extract audio from a YouTube video URL. It downloads the audio, which is then used for transcription.

Steps:

- Input the YouTube video URL.
- Use yt_dlp to extract the audio in the best available quality.
- Store the audio file in a specified directory for further processing.

Audio Transcription Algorithm: This algorithm leverages the AssemblyAI API to transcribe the audio into text.

Steps:

- Input the audio file obtained from the YouTube video.
- Send the audio to AssemblyAI's transcription service, which uses machine learning to convert audio to text.
- Receive and store the text transcript in a local directory.

Text Embedding and Retrieval Algorithm: This technique employs the FAISS library for effective similarity search and Google Generative AI for embedding. By embedding text fragments and storing them in a vector database for retrieval, it enables question-answering.

Steps:

- To preserve context and maximize retrieval, divide the transcript into smaller text segments using the RecursiveCharacterTextSplitter.
- Use Google Generative AI embeddings to transform each chunk into a high-dimensional vector.
- Store the vectors in an FAISS index to enable efficient similarity search.
- Extract the most relevant pieces from an input query using FAISS based on vector similarity**.**

Question answering algorithm: The Question-Answering Algorithm pulls pertinent text passages from the vector store and feeds them into a language model to produce answers to user questions.

Steps:

- Input the user's question.
- Retrieve the most relevant transcript chunks from the FAISS index using similarity
- search.
- Use Google Generative AI to generate an answer based on the retrieved context. Question-Answering Algorithm

7. RESULTS

The performance of the TubeTalk application was evaluated based on its core functionalities: transcript generation, summary extraction, and question-answering speed. The following metrics were recorded for a YouTube video of approximately **10 minutes** in duration:

1. Transcript Generation Time

If Transcript Available via YouTube Data API:

Time Taken: ~1–3 seconds (Instant)

If Transcript Not Available (Uses AssemblyAI):

Time Taken: ~1 minute

2. Summary Generation Time

Time Taken: ~10–15 seconds

3. Question & Answer (Q&A) Response Time

Time Taken per User Query: ~4–6 seconds

8. CONCLUSION

This paper presented a methodology for extracting, analyzing, and interacting with YouTube video content using advanced natural language processing and vector-based retrieval. By combining audio transcription, text segmentation, embedding, and generative AI, TubeTalk enables users to query video content, receive contextual answers, and take comprehension quizzes—all through an intuitive web interface. The system also includes a browser extension that enhances accessibility by summarizing videos and linking directly to the dashboard. This work demonstrates how modern AI techniques can transform passive video consumption into an active learning experience. While TubeTalk already provides meaningful interactions with video data, there is scope for future enhancements. Planned improvements include integrating Viewing history, real-time Live Stream Transcription, and google form quiz generation to expand usability and adaptability across diverse users and content types.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our guide, Supriya Miss, for her valuable guidance and unwavering support throughout this project. We also extend our heartfelt thanks to Fr. Conceicao Rodrigues College of Engineering for providing the necessary resources and a conducive environment to carry out this research.

REFERENCES

[1] Vladimir Karpukhin, Barlas Oğuz, Sewon Min,"Dense Passage Retrieval (DPR) for OpenDomain Question Answering," arXiv, 2020. [Online].
Available: https://arxiv.org/abs/2004.04906.

[2] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, "The Faiss Library," arXiv, 2024. [Online]. Available: https://arxiv.org/abs/2401.08281.

[3] "Premanand P. Ghadekar; Sahil Mohite; Omkar More., "Sentence Meaning Similarity Detector Using FAISS," IEEE Xplore, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/10392009.

[4] "Ankur Kumar; Priya Yadav; N. Partheeban., "YouTube Transcript Synthesis," IEEE Xplore, 2020. [Online].

Available: https://ieeexplore.ieee.org/document/10541713.

[5] Tirath Tyagi; Lakshaya Dhari; Yash Nigam., "Video Summarization using Speech Recognition and Text Summarization," IEEE Xplore, 2020. [Online].

Available: https://ieeexplore.ieee.org/abstract/document/10169901.

[6] "Jui-Ting Huang, Ashish Sharma, Shuying Sun., "Embedding-based Retrieval in Facebook Search," ACM Digital Library, 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3394486.3403305.

[7] Felipe Almeida, Geraldo Xexéo, "Word Embeddings: A Survey," arXiv, 2019. [Online]. Available:https://arxiv.org/abs/1901.09069. Yihan Wu, Ruihua Song, Xu Chen, "Understanding Human Preferences: Towards More Personalized Video to Text Generation," ACM Digital Library, 2023. [Online].

Available: https://dl.acm.org/doi/10.1145/3589334.3645711.

[8] Xudong Lin; Gedas Bertasius; Jue Wang., "VX2TEXT: End-to-End Learning of VideoBased Text Generation From Multimodal Inputs," IEEE Xplore, 2020. [Online].

Available: https://ieeexplore.ieee.org/document/9578716.

[9] Laura Hanu, Anita L. Verő, James Thewlis., "Language as the Medium: Multimodal Video Classification through Text Only," arXiv, 2023. [Online].

Available: https://arxiv.org/abs/2309.10783.

[10] Adith Sreeram a, Jithendra Sai;"An Effective Query System Using LLMs and LangChain," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/372529063_An_Effective_Query_System_Using_LLMs_and_LangChain.

[11] Keivalya Pandya, Mehfuza Holia, "Automating Customer Service using LangChain," arXiv, 2023. [Online].

Available: https://arxiv.org/abs/2310.05421.