Comparative Analysis of Language Model choices in a Video Search Pipeline: Evaluating Cross-lingual and Native Language Approaches

Keshav Bhupathy Vignesh Jayaprakasam[a;*], Karthikkumar V[a;], Aju John Thomas[a;***], Aditya Vipul Pradhan[a;****] and Srishti Upadhyay[a;*****]**

[a]Indian Institute of Science, Bengaluru
ORCiD ID: Keshav Bhupathy Vignesh Jayaprakasam https://orcid.org/0009-0006-3607-5372,
Karthikkumar V https://orcid.org/0009-0004-7113-7060, Aju John Thomas https://orcid.org/0009-0005-3945-2893,
Aditya Vipul Pradhan https://orcid.org/0000-0002-8212-3084,
Srishti Upadhyay https://orcid.org/0009-0009-7431-9325

Abstract. This project presents a comparative study of different model performance in Retrieval-Augmented Generation (RAG) for Indic language video content. We evaluate models from providers like Sarvam, Google, OpenAI and relevant open models across relevant components in multiple RAG strategies. We implement a RAG Pipeline that processes Indic language videos, has speech-to-text transcription, implements various embedding approaches, and enables multilingual querying via Text or Audio inputs. We also aim to implement advanced Audio/Video Summarization & Search techniques wherever possible. Through systematic evaluation of retrieval accuracy, response quality, and language preservation, this study aims to determine the optimal model and pipeline configuration for Indic video RAG applications, providing insights into the trade-offs between specialized language models and established global alternatives in multilingual video search scenarios. For the purposes of this limited duration study, we will restrict the language choices to Tamil, Malayalam and Hindi.

1 Introduction

In recent years, the consumption and production of Indic language video content have grown exponentially, driven by the increasing accessibility of digital platforms and the diverse linguistic landscape of India. This surge has created a rich repository of videos in Indian languages such as Tamil, Malayalam, and Hindi. The data also indicates that most of the users in India use Indic Languages to search, access and engage with this content [15]. This exponential growth in Indic video content consumption necessitates the development of technologies that can effectively process and analyze such data.

Indic languages, each with their distinct grammar, phonetics, dialects and cultural nuances, pose unique challenges for AI systems. These languages demand not only accurate transcription but also meaningful retrieval and summarization that respect linguistic integrity and context. Addressing these challenges requires innovation in both language-specific models and methods to adapt global AI advancements to regional needs. Large Multimodal Models today have the capability to accept and produce content in several languages. But due to the disparity of publicly available internet data (a large portion of the data until recent years was in English), these models may not always work very well for Indian Language Interactions because of the above reasons. There are several active threads in India today from teams such as AI4Bharat[2], Bhashini[3] and SarvamAI[31] that are tackling this problem of making AI and digital services accessible to Indian citizens in their own language.

In this project, we propose and implement an Indic Video Search Pipeline based on Retrieval Augmented Generation techniques. We package this pipeline in a Streamlit application. Users will have the option to search for relevant videos by text or voice in their native language. Such pipelines can help simplify the search and access for Indic Video Content in their native language. The following sections of the paper detail the architecture and key components, alternatives compared for each component and final choices, evaluation methodology and conclusions.

2 Architecture

The key idea in our approach to solve this problem was to convert Indic Language videos to English text summaries and vectorize them. Similarly, user inputs in Indic languages (via voice or text) need to be converted to English text. Then a RAG pipeline can be defined on this English corpus with English queries. Once the answer is generated, it can be converted back to the User's chosen language in their preferred medium. The key components of this version of the app are listed below. This flow is also depicted in Figure 1.

- *Video Chunking & Summarizatio*n - Videos in Indic languages must be segmented into smaller chunks. Each of these chunks should then be converted into detailed text summaries that can capture the diverse info available in the video image and audio
- *Indic Language Speech-to-text (STT) & Text-to-Speech(TTS)* - These components play an important role in ensuring that the user

* Corresponding Author. Email: keshavj@iisc.ac.in
** Corresponding Author. Email: karthikkuma1@iisc.ac.in
*** Corresponding Author. Email: ajuthomas@iisc.ac.in
**** Corresponding Author. Email: adityavp@iisc.ac.in
***** Corresponding Author. Email: srishtiu@iisc.ac.in

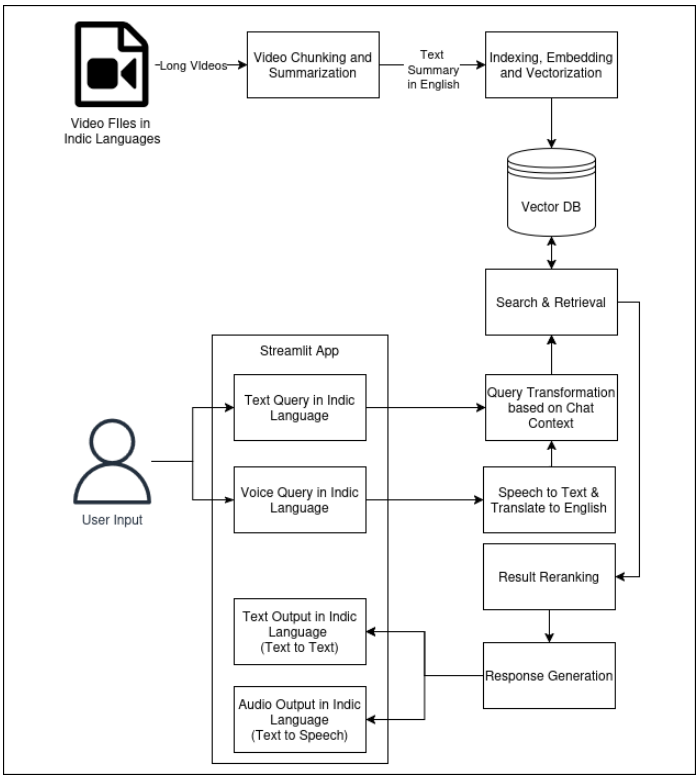

Figure 1. Simplified Component Architecture of the App

interactions with the system work reliably. These modules must capture the query from the user correctly & convey a response without loss of information

- *Indexing, Vectorization & Embedding* - The text summaries must be indexed, stored and vectorized as per the search technique requirements. Choices of the right embedding model & indexing approaches are important here
- *Search & Retrieval Techniques* - Identification of a search technique that can reliably retrieve relevant pieces of information from the knowledge corpus is a crucial part of the overall RAG system. This module ensures that the question is always answered with the most relevant information
- *Context Aware Chat & Query Transformations* - This module will keep track of user's intent, understand topic shifts and formulate meaningful queries from the user's chat context for searching. This will add to the frictionless interaction experience for the user

3 Comparative Analysis of Key Components

This section briefly summarizes the analysis, findings, and choices for the key components listed above. More details, whenever applicable, are described in the Appendix.

3.1 Speech-to-Text Model

A detailed evaluation of Speech-to-Text (STT) systems was carried out using multilingual audio data recorded by the author in English, Hindi, and Malayalam. Among the models evaluated—**Sarvam AI, Gemini, AI4Bharat, and OpenAI Whisper—Sarvam AI** demonstrated the most consistent performance across all languages. It achieved low Word Error Rate (WER) and Character Error Rate (CER) in both English and Indian languages. Gemini showed commendable speed and responsiveness but struggled with WER in Malayalam. AI4Bharat had slightly better accuracy in English but suffered from slower processing times. OpenAI Whisper could not be evaluated due to persistent API errors, and was excluded from the final analysis.

Given the overall accuracy, Sarvam AI was selected as the preferred STT model. Its 30-second native audio input limitation can potentially be bypassed using its batch processing mode. Further evaluation results are presented in **Appendix C**

3.2 Text-to-Speech Model

For the Text-to-Speech (TTS) evaluation, intelligibility was assessed by transcribing the synthesized audio using **Google Cloud STT** and computing the resulting WER and CER. Among Sarvam AI, Gemini, and AI4Bharat, **Sarvam AI** emerged as the most natural-sounding and intelligible voice generator across all three languages. It outperformed the others in Malayalam and showed competitive performance in English and Hindi. AI4Bharat's outputs, although acceptable, had higher WER and CER values and sounded relatively less natural. But it should be noted that AI4Bharat was translating all the audio to pure native language and that was the reason for higher WER scores.

Based on these evaluations, Sarvam AI was chosen as the final TTS engine. Detailed performance comparisons are provided in **Appendix C**.

3.3 Video Summarization

We had compared Gemini 2.5 Pro, Gemini 2.5 Pro and GPT-4o using the evaluation methodology described in **Appendix E** for video summarization. **Gemini 2.5 Pro** was selected on the basis of the results. Additionally, its ability to process entire video chunks, including animations, and provide a comprehensive and accurate summary makes it the preferred model. Its support for direct MP4 video summarization and effective integration of audio analysis further enhance its suitability for this task. While GPT-4o offers valuable context creation from frame information, its limitations in handling direct video files and potential clarity issues make Gemini 2.5 Pro the more robust option for video summarization.

3.4 Embedding Model

The video summarization finally outputs document chunks with a maximum sequence length of 256. Since this output is in English, the Sentence Transformer **MiniLM-L6-v2** encoder model was used to generate 384 dim vectors in this implementation. Embedding models that are fine-tuned or pre-trained for Indic languages specifically were also explored. **IndicSBERT** [6] was found to perform the best when we pitted it against other similar openly-available indic multiligual pre-trained language models. We bench marked the models in supervised fashion, upon multilingual queries and corpus which is explained in detail in **Appendix:F**.

3.5 Indexing, Searching and Retrieval

A custom query dataset based on select Indic Videos was created and an objective analysis was carried out to determine the best Search & Retrieval Approach. **Precision@5, Recall@5, F1 Score, MRR and nDCG@5** scores were calculated based on the ground truth

in this dataset. The goal was to ensure that relevant chunks to the user's query were present in the top 5 retrieved documents from the database, and ideally the most relevant chunk should be the first document out of the 5. Cosine Similarity, Max Marginal Relevance (MMR), Hybrid Search 1 (Cosine + MMR) & Hybrid Search 2 (Keyword Search using BM25+ MMR) with Reciprocal Rank Fusion (RRF) were considered for this analysis. It was found that a **hybrid search - BM25 + MMR - with RRF** often gave the best results as the original corpus becomes large and contains documents that are diverse and unrelated in nature. For every query, 10 document chunks were retrieved using this search and a final reranking using **MiniLM L6 V2 Crosscoder for MS Marco** was added to select the top 5 most relevant document chunks. Details about the dataset creation, comparisons and scores can be found in **Appendix: D**

3.6 Context Aware Chat and Query Transformations

To support coherent multi-turn interactions in a multilingual video retrieval setting, we evaluated four query transformation strategies: Query Decomposition, Hypothetical Document Embeddings (HyDE), Multi-Query Expansion, and RAG-Fusion. Each method was assessed using the same metrics as the Search & Retrieval Strategy evaluations - Precision@5, Recall@5, nDCG@5, MRR & F1-Score While HyDE produced semantically rich outputs, it incurred higher latency due to hypothetical answer synthesis. Therefore, HyDE is notably less efficient than all other methods, despite having the fastest search time. While Query Expansion provides marginally better retrieval quality, **RAG fusion achieves nearly identical effectiveness with approximately 12% better time efficiency**. The effectiveness differences are in the third decimal place, while the efficiency advantage of RAG fusion is more substantial. Therefore, based on our limited custom dataset evaluation, **RAG fusion would likely be the preferred choice** due to its better balance of effectiveness and efficiency. More details about the Analysis can be found in **Appendix G**

4 Conclusion and Future Work

As the growth of Indic Language Users & digital Content increases, solutions tailored to such users will gain prominence. The solutions that handle the variety of dialects and cultural nuances effectively would appeal to more users. LMMs would also continue to evolve and incorporate a significant amount of this data in their training and become more capable at handling such nuances. But today, pipelines that integrate the best model for a narrow capability are still relevant. We have implemented an end-to-end RAG based chat app for Indic Language Videos. The user can interact with the chat app in an Indic Language of Choice. We primarily tested the app with Tamil, Malayalam, and Hindi Language queries and videos. While this app performed reasonably on the small data corpus we tested it with, several improvements such as advanced Indexing, Search & Retrieval techniques, more nuanced Video Summarization approaches and natively multimodal pipelines that don't rely on text conversions can be explored to further improve performance. Streamlit was only used as a prototype UI and more robust solutions need to be implemented. The app would also need to incorporate guardrails and distributed computing techniques to truly scale to a larger audience.

Acknowledgements

We would like to thank Prof. Deepak Subramani and the DA225o course team for introducing us to this topic and related concepts in class, which inspired and enabled us to pursue this project.

References

[1] Tech that Works Aditya Kumar. Maximal marginal relevance to re-rank results in unsupervised keyphrase extraction, 2019. https://medium.com/tech-that-works/maximal-marginal-relevance-to-rerank-results-in-unsupervised-keyphrase-extraction-22d95015c7c5.

[2] AI4Bharat. Ai4bharat, a research lab at iit madras, is dedicated to advancing ai technology for indian languages through open-source contributions, 2025. https://ai4bharat.iitm.ac.in/.

[3] Bhashini. Bhashini is an ai powered language translation platform, bridging literacy, language, and digital divides, 2025. https://bhashini.gov.in/about-bhashini.

[4] Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder, 'Rethinking embedding coupling in pre-trained language models', *arXiv preprint arXiv:2010.12821*, (2020).

[5] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov, 'Unsupervised cross-lingual representation learning at scale', *arXiv preprint arXiv:1911.02116*, (2019).

[6] Samruddhi Deode, Janhavi Gadre, Aditi Kajale, Ananya Joshi, and Raviraj Joshi, 'L3cube-indicsbert: A simple approach for learning cross-lingual sentence representations using multilingual bert', in *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pp. 154–163, (2023).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding', in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, (2019).

[8] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang, 'Language-agnostic bert sentence embedding', *arXiv preprint arXiv:2007.01852*, (2020).

[9] Luyu Gao, Zhenzhong Lu, Zhuyun Dai, Jamie Callan, and Jimmy Lin, 'Hyde: Hypothetical document embeddings for zero-shot document ranking', *arXiv preprint arXiv:2204.05032*, (2022).

[10] Google. Google developers. https://developers.google.com/, 2025. Accessed: June 20, 2025.

[11] Google AI. Ai studio: Generate speech. https://aistudio.google.com/generate-speech, 2025. Accessed: June 20, 2025.

[12] Google AI. Gemini api: Audio. https://ai.google.dev/gemini-api/docs/audio, 2025. Accessed: June 20, 2025.

[13] Google Gemini. Google gemini cookbook. https://github.com/google-gemini/cookbook/, 2025. Accessed: June 20, 2025.

[14] Zellig S Harris, 'Distributional structure', *Word*, **10**(2-3), 146–162, (1954).

[15] India Digital Summit. Led by surge in indic language adoption internet users in india set to cross 900 million, 2025. Press Release https://www.indiadigitalsummit.in/wp-content/uploads/2025/01/Led-by-Surge-in-Indic-Language-Adoption.pdf.

[16] Gautier Izacard, Seyed Mehran Hosseini, Timo Schick, Divyanshu Dwivedi-Yu, Fabio Petroni, and Patrick Lewis, 'Few-shot learning with retrieval augmented language models', *arXiv preprint arXiv:2208.03299*, (2022).

[17] Sparsh Jain, Ashwin Sankar, Devilal Choudhary, Dhairya Suman, Nikhil Narasimhan, Mohammed Safi Ur Rahman Khan, Anoop Kunchukuttan, Mitesh M Khapra, and Raj Dabre. Bhasaanuvaad: A speech translation dataset for 13 indian languages, 2024.

[18] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul NC, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar, 'Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages', in *Findings of the association for computational linguistics: EMNLP 2020*, pp. 4948–4961, (2020).

[19] Simran Khanuja, Diksha Bansal, Sarvesh Mehtani, Savya Khosla, Atreyee Dey, Balaji Gopalan, Dilip Kumar Margam, Pooja Aggarwal, Rajiv Teja Nagipogu, Shachi Dave, et al., 'Muril: Multilingual representations for indian languages', *arXiv preprint arXiv:2103.10730*, (2021).

[20] Sankalp KJ, Ashutosh Kumar, Laxmaan Balaji, Nikunj Kotecha, Vinija Jain, Aman Chadha, and Sreyoshi Bhaduri, 'Indicmmlu-pro: Bench-

marking indic large language models on multi-task language understanding', *arXiv preprint arXiv:2501.15747*, (2025).
[21] Krutrim Team. Bharat bench: A benchmark for indic language models. Krutrim Tech Blog, Feb 2024.
[22] Yimeng Liu, Zhecheng Wu, Menglin Ren, Zhizheng Zhang, Thomas H. Li, Satwik Meena, Wenhu Ding, and Zhou Yu. Agentic keyframe search for video question answering, 2024.
[23] Mourad Mars, 'From word embeddings to pre-trained language models: A state-of-the-art walkthrough', *Applied Sciences*, **12**(17), 8805, (2022).
[24] Gurucharan MK. Unlocking the power of cosine similarity: the heart of text understanding, 2025. https://medium.com/@charan4u/unlocking-the-power-of-cosine-similarity-the-heart-of-text-understanding-eed427df745a.
[25] OpenAI. Gpt-4 technical report, 2024.
[26] OpenAI. Openai platform: Overview. https://platform.openai.com/docs/overview, 2025. Accessed: June 20, 2025.
[27] OpenAI. Openai platform: Rate limits. https://platform.openai.com/docs/guides/rate-limits, 2025. Accessed: June 20, 2025.
[28] Zackary Rackauckas, 'Rag–fusion: A new take on retrieval-augmented generation', *arXiv preprint arXiv:2402.03367*, (2024).
[29] Raghavan Rajkumar, Abhigyan Kumar, Gurunath Parameshwara, Divyanshu Agnivesh, and Srikanth Madikeri. Enhancing whisper's accuracy and speed for indian languages through prompt-tuning and tokenization, 2023.
[30] Adrian H. Raudaschl. Forget rag, the future is rag–fusion. https://towardsdatascience.com/forget-rag-the-future-is-rag-fusion-1147298d8ad1, 2023.
[31] Sarvam. At sarvam, we're on a mission to make generative ai real for bharat, 2025. https://www.sarvam.ai/about-us.
[32] Sarvam AI. Sarvam ai documentation. https://docs.sarvam.ai/, 2025. Accessed: June 20, 2025.
[33] Deval Shah. Reciprocal rank fusion (rrf) explained, 2024. https://medium.com/@devalshah1619/mathematical-intuition-behind-reciprocal-rank-fusion-rrf-explained-in-2-mins-002df0cc5e2a.
[34] Sanket Shah, Kavya Ranjan Saxena, Kancharana Manideep Bharadwaj, Sharath Adavanne, and Nagaraj Adiga, 'Indicst: Indian multilingual translation corpus for evaluating speech large language models', in *2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, pp. 1–5. IEEE, (2025).
[35] Aatman Vaidya, Tarunima Prabhakar, Denny George, and Swair Shah, 'Analysis of indic language capabilities in llms', *arXiv preprint arXiv:2501.13912*, (2025).
[36] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei, 'Multilingual e5 text embeddings: A technical report', *arXiv preprint arXiv:2402.05672*, (2024).
[37] Wikipedia. Okapi bm25, 2025. https://en.wikipedia.org/wiki/Okapi_BM25.
[38] Xueguang Zhu, Yue Yang, Luyu Gao, and Jimmy Lin. Multiview prompting for generating query variants. https://arxiv.org/abs/2304.10149, 2023.

A Contributions

- **Keshav Bhupathy Vignesh J** - High Level Architecture, Base End-to-end App setup and GitHub maintenance, Searching & Retrieval Strategies Comparison, Team Guidance
- **V Karthikkumar** - Speech to Text, Text to Speech Implementations for Indic Languages, Model alternatives identification & performance comparisons
- **Aju John Thomas** - Video Processing Pipeline (Input Video Chunking and Summarization), Model alternatives identification & performance comparisons
- **Aditya Vipul Pradhan** - Embedding Model Alternatives Identification, Performance Comparison, App Deployment
- **Srishti Upadhyay** - Context Aware Chat & Query Transformation techniques comparison and Implementation

B Full Code & Live App

More information, screenshots and complete source code for the final app with the chosen alternatives can be found in our GitHub Repository: https://github.com/jkeshav-bvignesh/IndicVideoSearch. App Screenshots are also added at the end of this document.

The live app will be made available here: https://indicvideosearch.streamlit.app/ for a limited time for review.

C Text-to-Speech and Speech-to-Text Model Comparisons

This section details the evaluation methodology, comparative performance analysis, and final selection of Speech-to-Text (STT) and Text-to-Speech (TTS) models for the project. The goal was to identify the most suitable models based on accuracy, Indian language support, and operational characteristics within the project's scope. The evaluation of models, particularly for Indic languages, draws upon the broader context of ongoing research in understanding and enhancing these capabilities in Large Language Models (LLMs) [35, 34].

C.1 Evaluation Framework

A systematic approach was taken to evaluate the models across multiple languages using a custom dataset.

C.1.1 Models Evaluated

The following models were considered for evaluation:

- Sarvam AI model [32]
- Google's GEMINI model [12, 13]
- AI4BHARAT Indic Speech model [17]
- OpenAI's Whisper model [26]

C.1.2 Language Scope

The models were tested for their performance in three languages:

- English
- Hindi
- Malayalam

The challenges and nuances of working with Indic languages like Hindi and Malayalam are an active area of research, with efforts focused on creating comprehensive datasets and evaluation benchmarks [17, 34].

C.1.3 Evaluation Dataset

The dataset used for evaluation consisted of audio recordings and corresponding transcriptions/translations created by the author. This custom dataset was designed to reflect the specific use cases and linguistic differences, covering all three target languages (English, Hindi, and Malayalam).

C.1.4 Metrics for Performance Assessment

The primary metrics chosen for evaluating both STT and TTS model performance were Word Error Rate (WER) and Character Error Rate (CER).

Word Error Rate (WER) is a standard metric for Automatic Speech Recognition (ASR) that measures the number of errors (substitutions, insertions, deletions) at the word level, normalized by the total number of words in the reference transcript. It provides an intuitive measure of the overall accuracy of the transcription.

Character Error Rate (CER) operates similarly to WER but at the character level. This metric is particularly useful for languages with complex morphology or where word segmentation can be ambiguous. It complements WER by capturing finer-grained errors, such as misspellings within correctly identified words.

These metrics were chosen due to their widespread adoption, ease of interpretation, and direct relevance to the perceived quality of STT and TTS outputs. While jiwer and other libraries support metrics like Match Error Rate (MER), Word Information Lost (WIL), etc., they often build upon similar principles to WER and CER.

C.1.5 Text-to-Speech (TTS) Evaluation Judge

For evaluating the intelligibility of the synthesized speech from the TTS models, **Google Cloud TTS** was employed as an automated **"judge"** [10]. The audio output from each candidate TTS model was fed into Google Cloud's STT service, and the resulting transcript was compared against the original text to calculate WER and CER. This provided an objective measure of how well the synthesized speech could be understood by a high-quality ASR system.

C.2 Speech-to-Text (STT) Model Evaluation

The STT capabilities of the selected models were rigorously tested.

C.2.1 Individual Model Performance Notes

- **Sarvam AI**: This model [32] demonstrated excellent performance in terms of Word Error Rate (WER). A notable limitation is its native support for audio segments up to 30 seconds. While this can be overcome using a batch processing model, further exploration would be required for longer audio inputs. For the current project scope, the 30-second limit was deemed acceptable.
- **GEMINI**: GEMINI [12, 13] was observed to be fast and responsive across all test cases. However, its STT accuracy, particularly WER, was slightly lower for Indian languages like Hindi and Malayalam compared to other models or its English performance. The general capabilities of LLMs in Indic languages remain an area of active study [35].
- **AI4BHARAT**: The AI4BHARAT model, part of a broader initiative to enhance Indian language technology [17], showed promising results but generally lagged behind in overall WER. A significant issue observed was its tendency to translate words into the target language during transcription tasks, which incorrectly inflated its WER for STT evaluation. Additionally, unlike other models, AI4BHARAT did not consistently handle punctuations such as full stops and commas in its output. The model was also noted to be relatively slow in processing.
- **OpenAI Whisper**: Attempts to use OpenAI's Whisper API [26] were consistently met with an `insufficient_quota` error, despite verifying that sufficient quota was available. This issue aligns with potential challenges related to API rate limits [27]. Due to time constraints and persistent debugging challenges, evaluation of the Whisper API model had to be abandoned for this project.

C.2.2 STT Comparative Results

The following tables summarize the WER and CER for the STT models across the tested languages.

Table 1. STT Word Error Rate (WER) Summary

Language	ai4bharat	gemini	sarvam
English	0.1321	0.1698	0.1698
Hindi	0.1379	0.1207	0.1552
Malayalam	0.5714	0.4671	0.4571

Table 2. STT Character Error Rate (CER) Summary

Language	ai4bharat	gemini	sarvam
English	0.0383	0.0437	0.0437
Hindi	0.0719	0.0437	0.0469
Malayalam	0.1230	0.0518	0.0508

Visual comparisons of these metrics are provided in Figure 2.

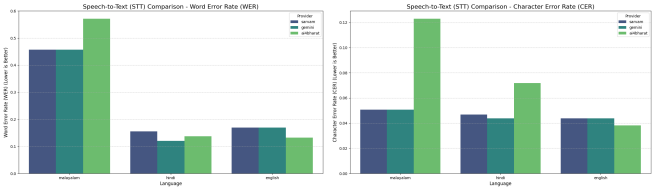

Figure 2. STT Performance: WER (left) and CER (right) Comparisons Across Models.

C.3 Text-to-Speech (TTS) Model Evaluation

The TTS capabilities were evaluated for intelligibility using an ASR judge.

C.3.1 Individual Model Performance Notes

- **Sarvam AI**: This model [32] produced good and natural-sounding audio. Its performance in terms of intelligibility (WER as judged by Google Cloud ASR) was also excellent. The 30-second audio generation limit per request is a known constraint but acceptable for this project.
- **GEMINI**: While generally performing well, GEMINI was found to not officially support TTS for the Malayalam language at the time of evaluation, but it performed reasonable in audio generation. [11].

- **AI4BHARAT**: The AI4BHARAT TTS model showed promising results in some cases, but its overall intelligibility scores (WER/CER) varied across languages. Evaluating TTS for diverse Indic languages is an ongoing challenge [34].

C.3.2 TTS Comparative Results

The intelligibility of the TTS outputs, as judged by Google Cloud ASR, is summarized in the tables below.

Table 3. TTS Intelligibility WER Summary (Judged by ASR)

Language	ai4bharat	gemini	sarvam
English	0.6415	0.1698	0.5283
Hindi	0.1034	0.0517	0.0862
Malayalam	0.8286	0.6571	0.4857

Table 4. TTS Intelligibility CER Summary (Judged by ASR)

Language	ai4bharat	gemini	sarvam
English	0.3716	0.0683	0.5683
Hindi	0.0469	0.0281	0.0344
Malayalam	0.7219	0.2995	0.1364

Visual comparisons of these metrics are provided in Figure 3.

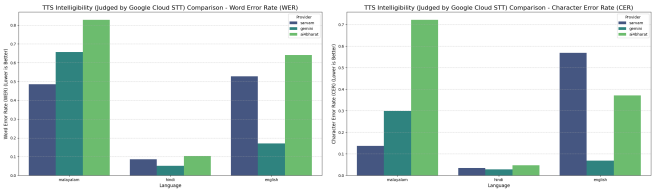

Figure 3. TTS Intelligibility Performance: WER (left) and CER (right) Comparisons Across Models (Judged by ASR).

C.4 Final Model Selection and Justification

Based on the comprehensive evaluation encompassing STT accuracy, TTS intelligibility, language support, and operational characteristics, **Sarvam AI** [32] was chosen as the primary model for both Speech-to-Text and Text-to-Speech tasks in this project.

The decision was driven by the following key factors:

- **Strong STT Performance**: Sarvam AI demonstrated competitive Word Error Rates for STT across the tested languages, particularly for English and Malayalam (tying with Gemini). While its Hindi STT WER was slightly higher than Gemini's in this specific test, its overall STT profile was robust.
- **Effective TTS Performance**: Sarvam AI also yielded good results for TTS intelligibility, especially notable in Malayalam where it significantly outperformed other models. The synthesized audio quality was perceived as natural.
- **Balanced Capabilities**: Considering both STT and TTS, Sarvam AI offered a strong, balanced performance across the required languages.
- **Acceptable Limitations**: The 30-second audio processing limit for Sarvam AI's native API was deemed acceptable for the scope of this project. The potential to overcome this with batch processing offers a path for future scalability if needed.

While GEMINI [12, 11] showed excellent speed and strong performance in English STT and Hindi STT/TTS, its lack of more Indian language TTS support and slightly lower STT accuracy in Indian languages were drawbacks. AI4BHARAT [17], despite showing promise, faced challenges with STT accuracy (partially due to translation behavior and punctuation handling) and processing speed, making it less suitable as the primary choice. The issues encountered with OpenAI's Whisper API [26, 27] precluded its full evaluation.

Therefore, Sarvam AI emerged as the most well-rounded and effective solution for the project's requirements.

D Search and Retrieval Strategies Comparison

Search & Retrieval strategies are an important component for a good RAG system. This section describes the various approaches that were tried along with comparisons. Here's a brief overview of the relevant techniques:

D.1 Cosine Similarity

Cosine similarity works by calculating the angle between two vectors in an n-dimensional space. In RAG, Cosine similarity is applied to calculate the similarity score between the query vector and each document vector. The top-k documents that are closest to the query vector are retrieved in this approach. More information about cosine similarity based search implementations can be found here: [24]. Cosine Similarity is often the simplest approach for such searches but does work reasonably for small data corpuses.

D.2 Maximum Marginal Relevance (MMR)

While cosine similarity works, it also always fetches the closest vectors. But what if the answer to the question requires a more comprehensive picture? For example: consider this simple scenario: We are retrieving features regarding a new product from a catalog. The output may look something like this: *[Good Product, Great Product, Nice Product, Excellent Product, Easy Install, Nice UI, Light weight]*. The first 4 phrases are ranked higher but all 4 are talking about the same aspect of the product. If we restrict our search to top-5 chunks, we would miss critical information such as "light weight" & "Easy Install". This is the problem MMR tries to solve. Aditya Kumar's Tech that works blog article[1] gives a good quick overview of the problem and the algorithm. MMR introduces a *diversity* hyperparameter on top of Similarity search to retrieve more relevant documents. In our app, MMR has been implemented with a diversity value of 0.3

D.3 BM25 & Hybrid Search

BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document [37]. It works by calculating a score based on the Term Frequency-Inverse Document Frequency (TF-IDF) between the query and document corpus. Based on these scores, the top-k relevant document chunks can be retrieved from the corpus. This algorithm has a high chance of finding document chunks that have the same key terms present in the user query. These key-terms may not necessarily have any semantic meaning (e.g., Document ID, UID, Names, Nouns etc.) A pure semantic search may therefore miss these documents. The trade-off with this approach is that semantic meaning is completely lost. Also, if the

Table 5. A snippet from the Video Search Evaluation Dataset that shows how one query is mapped to multiple chunks in a video

Dataset ID	Query ID	Query	Video Name	Chunk
activation_functions	1	What are the three types of activation functions explained in the video?	Hindi_Activation Function Part-11 Linear, Heviside Step, Sigmoid Functions Explained In Hindi.mp4	1
activation_functions	1	What are the three types of activation functions explained in the video?	Hindi_Activation Function Part-11 Linear, Heviside Step, Sigmoid Functions Explained In Hindi.mp4	2
activation_functions	1	What are the three types of activation functions explained in the video?	Hindi_Activation Function Part-11 Linear, Heviside Step, Sigmoid Functions Explained In Hindi.mp4	10

data corpus is very huge, initialization will take a long time as the index has to be calculated. If a new video is processed, the entire index has to be recalculatated which is an additional overhead. In this implementation the NLTK (https://www.nltk.org/) and RankBM25 (https://pypi.org/project/rank-bm25/) Python libraries were used. The best way to use BM25 is to combine it with a semantic search approach and merge the results using an algorithm such as RRF.

D.4 Reciprocal Rank Fusion (RRF)

When a hybrid search approach is implemented, results from multiple search algorithms are obtained. RRF is an algorithm that can be used to merge these results efficiently and calculate a unified score (more details here: [33]). In our app, we use a hybrid search approach that combines BM25 & Similarity Search or MMR. The results obtained from both the search approaches are merged together using RRF to get a unified result. Since we retrieve top-k from each approach, we finally get upto 2k chunks (there may be duplicates). Since this ranking is purely statistical, it doesn't take much additional computation

D.5 Reranking

The results from Hybrid Search + RRF was found to contain the most relevant chunks in most cases but sometimes, the most relevant document ends up lower in the list of top-2k chunks. An additional reranker that ranks every result with the query can help to re-sort the list and bring the most relevant chunks to the top. Post this step, the top-k chunks are finally filtered and returned to an LLM to generate a response. In this implementation this model - https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2 - was used to rerank the results.

D.6 Custom Dataset Creation

While retrieval Benchmark datasets such as BeIR dataset (https://github.com/beir-cellar/beir) exist, no dataset specific to Indic Language content was identified. BeIR is a heterogeneous benchmark containing diverse IR tasks. While this is not a video-RAG specific dataset, the information retrieval capability alone can be tested using this. But given that the goal was to test how Retrieval would work for the summaries that were generated a custom dataset was created.

The dataset consists of Queries and multiple relevant answer chunks that could answer that query. This was needed because unlike the BEIR datasets such as Scidoc & HotpotQA, the information in a video is continuous and could be spread out over multiple chunks. To factor this into account, a dataset as shown in Table 5 was created. This enabled the effective calculation on Precision@5, Recall@5, F1 Score, MRR & nDCG@5

D.7 Metrics Explained

Precision@5: The proportion of retrieved video chunks in the top-5 results that are relevant to the query:

$$\text{Precision@5} = \frac{\text{no. of relevant chunks in top-5}}{\text{total no. of retrieved chunks}} \tag{1}$$

Recall@5: The proportion of all relevant video chunks that appear in the top-5 results:

$$\text{Recall@5} = \frac{\text{no. of relevant chunks in top-5}}{\text{total no. of relevant chunks for the query}} \tag{2}$$

F1 Score: The harmonic mean of precision and recall, balancing both metrics:

$$\text{F1} = \frac{2 \times \text{Precision@5} \times \text{Recall@5}}{\text{Precision@5} + \text{Recall@5}} \tag{3}$$

MRR (Mean Reciprocal Rank): MRR measures how early the first relevant chunk appears in the results:

$$\text{MRR} = \frac{1}{\text{rank of first relevant chunk}} \tag{4}$$

Higher values (closer to 1) indicate that relevant chunks appear earlier in results.

nDCG@5 (Normalized Discounted Cumulative Gain): Evaluates the ranking quality of the top-5 results:

$$\text{nDCG@5} = \frac{\text{DCG@5}}{\text{IDCG@5}} \tag{5}$$

where:

$$\text{DCG@5} = \sum_{i=1}^{5} \frac{\text{rel}_i}{\log_2(i+1)} \tag{6}$$

$\text{rel}_i = 1$ if chunk at position i is relevant, 0 otherwise.

IDCG@5 is the ideal DCG value if all relevant chunks were ranked first. Higher values (closer to 1) indicate better ranking of relevant chunks.

In this video search context, these metrics help evaluate how well the system retrieves relevant video chunks for a query. For example, if a query about "activation functions" has 3 relevant chunks (chunks 1, 2, and 10) as shown in the dataset snippet, and the implemented system returns chunks [1, 3, 10, 5, 7]: Precision@5 = 2/5 = 0.40 (2 relevant chunks in 5 results) Recall@5 = 2/3 = 0.67 (2 of the 3 relevant chunks were found) MRR = 1/1 = 1.00 (first result is relevant) nDCG@5 would be high since relevant chunks appear at positions 1 and 3

D.8 Final Results

The above metric scores were calculated for the small custom dataset (118 queries-relevant chunk pairs) that was created for the purposes

Table 6. Precision, Recall, and F1 Score for Different Search Methods

Method	Precision@5	Recall@5	F1 Score
similarity	**0.35**	0.73	**0.46**
mmr	0.28	0.78	0.40
hybrid (Cosine+MMR)	0.29	0.85	0.41
BM25+MMR	0.30	**0.88**	0.42

Table 7. Ranking Quality Metrics for Different Search Methods

Method	MRR	nDCG@5
similarity	0.68	0.66
mmr	0.72	0.69
hybrid (Cosine + MMR)	0.72	0.71
BM25+MMR	**0.76**	**0.74**

of this project. The results of all approaches were re-ranked before evaluation. While this is a limited analysis, the results still show a very clear difference. We can see that:

- Similarity has the highest precision (0.35) and F1 score (0.46), meaning it returns the most relevant results with less noise, but misses some relevant items.
- BM25+MMR performs best in:
  - Recall@5 (0.88): finds the most relevant items
  - MRR (0.76): ranks the first relevant result higher
  - nDCG@5 (0.74): best overall ranking quality considering both relevance and position

  Hybrid approach performs well in recall but is generally outperformed by BM25+MMR
- MMR has the lowest precision but improved recall compared to similarity.

Therefore it is concluded that **BM25+MMR** is the better search approach overall because it:

- Finds significantly more relevant results (highest recall)
- Places relevant results higher in the ranking (highest MRR)
- Produces the best overall ranking quality (highest nDCG)
- Has reasonable precision despite prioritizing recall

For a video search system, BM25+MMR+Reranking would provide users with the most comprehensive results while ensuring the most relevant videos appear at the top of search results. Since we run 2 searches, there is a marginal increase in total search time.

D.9 Future Work

While the above approach achieves the best overall performance, several more advanced embedding and search approaches could further enhance the video retrieval system such as:

- **Splade (Sparse Lexical and Dense Expansion):** Combines sparse and dense representations for more nuanced semantic understanding
- **Knowledge Graphs**: Could capture relationships between concepts across videos, enabling more contextual searches and revealing connections between related educational content
- **Previous Chunk-Next Chunk Retrieval**: Would improve temporal continuity by considering adjacent chunks when retrieving video segments, providing more coherent results

These approaches were not explored in the current project due to time limitations, and the complexity of implementation. Future iterations could incorporate these techniques to potentially achieve higher precision while maintaining the strong recall performance observed in our current best model.

E Video Summarization strategies Comparisons

Video summarization involves understanding the video content, analyzing the audio, and examining the interrelation between video and audio. We have stored various types of Indic language videos, including Hindi, Malayalam, and Tamil to use as information context to answer user queries. Video summarization is done through a comparative study between Gemini AI - Gemini 2.5-pro-preview-06-05 and OpenAI - GPT-4o models. The video is segmented into multiple 30-second intervals. While GPT-4o does not support direct MP4 video summarization, Gemini 2.5 Pro does. GPT-4o utilizes a keyframe model for video summarization, with audio translated via the Sarvam AI model. This translated audio is then incorporated into the context for complete chunk summarization. Both models are provided with structured prompts to ensure similar outputs for comprehensive analysis. The output is then stored in a directory for subsequent processes, such as video embeddings.

E.1 Video Analysis

The video is ingested using OpenCV to extract essential video information like frame count, FPS, frame width, and height. For GPT-4o [25], the focus is on extracting keyframes from video and audio from another model like Sarvam or Whisper. Keyframe selection is the paramount focus in this analysis where Uniform sampling or VideoTree using CLIP can be performed to find keyframes [22]. VideoTree computes image features and K-means clustering to detect frame-to-frame differences, thereby identifying visual changes. Uniform sampling, on the other hand, implements time-based sampling with respect to the number of frames captured. Base64 is used as a bridge between binary image data and text formats. One of the challenges faced is the token limitation with encoded frames and associated cost during comparative analysis. To address this, we have reduced the size and image quality of frames to meet the token requirements. For the Gemini model, we have shared chunk-wise clips with the model to generate video summarization as per the prompt. For audio analysis, Sarvam AI is used for audio translation, and in Gemini computation, the audio is sent along with the video. Although OpenAI Whisper-M model also can be used for audio translation, its baseline PT model has a high Word Error Rate (WER) percentage (104%) for the Indic audio translation [29].

E.2 Gemini 2.5 Pro vs Gemini 1.5 Pro

Gemini 2.5 Pro and Gemini 1.5 Pro both accept video and are good fits for video model comparison. Gemini 2.5 Pro has a slight edge over the old model in capturing minute details. For instance, while checking for video key events, Gemini 2.5 Pro has inferred extra features of the video and given a well-detailed event summary. For instance, in the Transformer related video, the author is giving input 'GPT-3 as an autoregressive language model'. Here, Gemini 2.5 Pro has captured the details of autoregressive whereas Gemini 1.5 Pro has just given the idea that 'GPT-3's ability to generate human-like text'. In another video, transition of slides with the addition of formula in the same context was identified by Gemini 2.5 Pro whereas Gemini 1.5 Pro was only explaining the formula instance while checking key frame inputs. As per Google reports, Gemini 2.5 Pro has TextVQA score of 74.6% and 73.5% on image understanding.

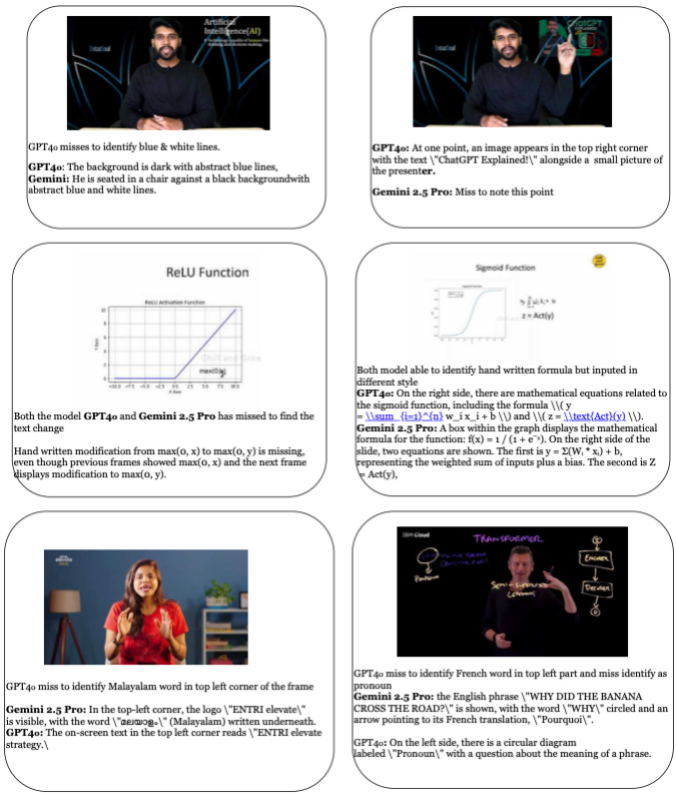

Figure 4. KeyFrame Evaluation

E.3 Evaluation

In this comparative study, we conducted frame-by-frame analysis comparing GPT-4o key frames summary with Gemini 2.5 Pro summary output. We found that Gemini 2.5 Pro is capable of understanding the entire video, including any animations, as it accepts the full video file. This allows Gemini to provide a more comprehensive and accurate summary of the video content. On the other hand, GPT-4o can extract frame information, but may miss small animations and some details due to low image quality. However, GPT-4o is still able to create context from extracted frames, which contributes to its summarization capabilities. The summaries generated by both models included visual descriptions of scenes, people, objects, and text. Audio content analysis was also performed to provide a summarized view, rather than a word-for-word translation. Future improvements could include smarter chunk selection for scene detection, combining frame-based and chunk-based methods for optimal results, enhanced keyframe extraction techniques, and adaptive token management.

F Embedding Model Comparisons

Word embeddings are dense vector-based feature representations of words in a given dimensional space [14]. Pre-trained language models (PLM) are one variation of them, they provide state-of-the-art performance in numerous Natural Language Processing tasks like summarisation, information retrieval, text classification, etc. [4] [23]

We shortlisted some top openly available PLM's for our study which included general multilingual models which included indic languages(XLM-RoBERTa[5], mBERT[7], LaBSE[8], Multilingual-E5[36]), models specifically pre-trained for indic languages (IndicSBERT[6], IndicBERT[18], MuRIL[19], Vyakyarth[21]) and monolingual model MiniLM which is designed only for English. For testing the models, we created a multilingual corpus (English, Hindi, Malayalam and Tamil) of 50 documents with significant overlap and 20 challenging queries to test for deeper semantic understanding. The queries ranged from cross-lingual information retrieval, intent based questions and negative constraints. For MiniLM we translated all queries and documents to English while for other models they were preserved in their native languages and then converted to dense vector embeddings. For searching we used Cosine similarity to generate a list raked by the most semantically relevant documents at the top. The evaluation was performed in a supervised manner with a pre-defined mapping from the query to the correct document. For evaluation we used two scoring metrics, recall and Mean Reciprocal Ranks (MRR). Recall indicates if the correct document appears in the top five results, while MRR is the average of the reciprocal rank at which the correct document was found. It rewards the model for placing the correct document higher in the list.

Table 8. Performance Table (Model vs. Query)

model	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20
IndicBERT	0.00	0.33	1.00	0.00	1.00	0.33	0.20	1.00	0.00	0.50	1.00	0.00	1.00	0.00	0.20	0.10	0.10	0.10	0.33	0.20
L3Cube-IndicsBERT	1.00	1.00	0.33	0.20	0.50	0.50	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.10	0.10	0.10	0.10	1.00	1.00
LaBSE	1.00	0.00	0.00	0.50	1.00	0.20	1.00	1.00	0.00	0.33	1.00	1.00	1.00	1.00	0.25	0.10	0.10	0.10	0.50	0.33
MiniLM (Translated)	0.50	1.00	0.00	0.20	0.50	0.60	1.00	1.00	1.00	0.50	1.00	1.00	1.00	0.00	1.00	0.10	0.10	0.10	0.50	1.00
MuRIL	0.50	1.00	0.33	0.00	1.00	0.33	0.50	1.00	0.20	1.00	1.00	0.00	1.00	1.00	1.00	0.10	0.10	0.10	0.25	1.00
Multilingual-E5	1.00	1.00	0.00	0.00	0.50	0.60	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.10	0.10	0.10	0.20	1.00
XLM-RoBERTa	0.33	0.00	0.20	0.00	0.00	0.33	0.50	0.00	0.00	1.00	1.00	0.00	0.33	0.20	0.10	0.10	0.10	0.00	0.00	0.00
mBERT	1.00	0.25	0.25	0.00	0.50	0.00	0.33	1.00	0.00	1.00	1.00	0.00	1.00	1.00	1.00	0.10	0.10	0.10	0.20	0.50

Table 2 shows a detailed breakdown of each model's performance over twenty distinct enquiries. Table 3 presents an overview of this performance ordered by Mean Reciprocal Rank (MRR). From the above comparison study we found that L3Cube-IndicSBERT which is based on MuRIL[6] performed the best with a MRR score of 0.7417 and perfect recall score. Figure 2. demonstrates a superior grasp by the IndicSBERT model in underlying semantic structures in the word analogy tests. Its shows that it creates a well-organized vector space that preserves linguistic relationships. As the model also performs much better than the MiniLM model we can also conclude that it is better to store the embeddings in the the native language than translating it to another language and using a monolingual embedding, as it could lead to a loss in the semantic nuance during translation.

Table 9. Overall Performance (Sorted by MRR)

model	MRR	Average_Recall_at_5
L3Cube-IndicsBERT	0.7417	1.0000
Multilingual-E5	0.6500	0.8500
MiniLM (Translated)	0.5750	0.8500
MuRIL	0.5708	0.9000
LaBSE	0.5208	0.8500
mBERT	0.4667	0.8000
IndicBERT	0.3700	0.7500
XLM-RoBERTa	0.2600	0.6000

To ensure reproducibility and control, the scope of this study was purposely limited to publicly available models. Future studies might benefit from using commercial embedding APIs, such as those from OpenAI, to benchmark their performance on these difficult multilingual tasks and gain a more comprehensive understanding of the SOTA environment, similar to the IndicMMLU-Pro benchmarking study.[20]

Figure 5. Visual Representation of Word Analogy Results

```
English Analogies:
  Gender (avg similarity: 0.877):
    son − boy + girl = daughter → 0.938
    brother − boy + girl = sister → 0.907
    actor − man + woman = actress → 0.899
  Capitals (avg similarity: 0.328):
    moscow − russia + brazil = brasilia → 0.457
    paris − france + italy = rome → 0.423
    tokyo − japan + china = beijing → 0.388
  Plurals (avg similarity: 0.768):
    foot − feet + tooth = teeth → 0.878
    cat − cats + dog = dogs → 0.806
    book − books + pen = pens → 0.806
  Comparative (avg similarity: 0.737):
    fast − faster + slow = slower → 0.792
    big − bigger + small = smaller → 0.758
    hot − hotter + cold = colder → 0.740

Hindi Analogies:
  Gender (avg similarity: 0.744):
    raja − aadmi + aurat = rani → 0.880
    pati − aadmi + aurat = patni → 0.834
    pita − aadmi + aurat = mata → 0.830
  Relations (avg similarity: 0.734):
    mama − bhanja + bhanji = mami → 0.854
    bap − beta + beti = maa → 0.711
    dada − pota + poti = dadi → 0.637
  Opposites (avg similarity: 0.592):
    din − raat + ujala = andhera → 0.652
    garmi − sardi + aag = paani → 0.612
    upar − niche + aage = piche → 0.512

Tamil Analogies:
  Gender (avg similarity: 0.515):
    magan − paiyan + ponnu = magal → 0.598
    arasan − aan + pen = arasi → 0.551
    anna − paiyan + ponnu = akka → 0.479
  Relations (avg similarity: 0.676):
    appa − magan + magal = amma → 0.729
    thatha − peran + petti = paatti → 0.623
  Opposites (avg similarity: 0.389):
    pagal − iravu + velicham = iruttam → 0.439
    veppam − kulir + thee = neer → 0.340

Malayalam Analogies:
  Gender (avg similarity: 0.395):
    ammavan − aan + pen = ammavi → 0.483
    chettan − aan + pen = chechi → 0.451
    rajan − aan + pen = rani → 0.446
  Relations (avg similarity: 0.716):
    muthashan − makanmakan + makalmakal = muthassi → 0.778
    achan − makan + makal = amma → 0.655
  Opposites (avg similarity: 0.504):
    chooduppu − thanduppu + thee = vellam → 0.555
```

G Context Aware Chat and Query Transformation Comparisons

Query transformation is a core component of Retrieval-Augmented Generation (RAG) systems, especially in conversational, multi-turn settings where the system must interpret the current query in light of previous dialogue. This appendix examines four transformation strategies—Query Decomposition, HyDE, Multi-Query Expansion, and RAG-Fusion—as applied to educational or exploratory video search. Each method is assessed on Context Retention, Reformulation Diversity, Precision@5, Latency, and overall Efficiency.

G.1 Query Decomposition

Query Decomposition splits a complex query into several sub-queries, each targeting a distinct aspect of user intent. This enhances recall and interpretability but introduces overhead and latency, as separate retrieval calls are needed. Moreover, unless explicitly included, conversational context may be lost. Its limitations in multi-turn settings have been noted in recent RAG pipeline studies [16].

G.2 Hypothetical Document Embeddings (HyDE)

HyDE first "hallucinates" a plausible answer using an LLM and retrieves documents similar to that generated text [9]. This can boost precision—especially for vague queries—but adds computational cost and may introduce hallucinated context. It also doesn't automatically utilize dialogue history unless specifically prompted.

G.3 Multi-Query Expansion

Multi-Query Expansion generates multiple paraphrases of the user's query and aggregates the results [38]. While effective for increasing recall, it often leads to redundancy and lacks sensitivity to conversational context, which is critical in chained query scenarios.

G.4 RAG-Fusion with Context-Aware Summarization

RAG-Fusion extends query expansion by leveraging both recent dialogue context and lightweight summarization of earlier turns. The system maintains a rolling memory window: when the dialogue length exceeds a threshold, it summarizes older prompts. It then constructs a prompt that produces multiple semantically distinct reformulations (e.g., focusing on definitions, examples, methods, or pitfalls). These reformulations are cleaned, deduplicated, then executed through a hybrid retriever and fused using Reciprocal Rank Fusion (RRF) [28, 30]. This approach preserves conversational continuity and yields diverse, relevant search results in real-time. The use of RRF specifically enhances result quality in hybrid retrieval scenarios [33].

G.5 Comparative Summary

To compare these methods fairly, we use the same evaluation methodology as the RAG method evaluations and calculate **Precision@5, Recall@5, nDCG@5, MRR & F1 Score**. We keep the search method constant (BM25 + MMR + Reranking as that was concluded as the best option) and only change the query transformation approaches for our custom dataset. The results are as follows:

Table 10. Effectiveness Metrics

Method	Precision@5	Recall@5	F1 Score	MRR	nDCG@5
rag_fusion	0.3178	0.9795	0.4608	0.7643	0.7428
hyde	0.2922	0.8961	0.4229	0.7349	0.7276
decomposition	0.3096	0.9669	0.4509	0.5220	0.4820
expansion	0.3178	0.9886	0.4619	0.7665	0.7460

Additionally, we also calculate some Efficiency metrics as described below:

- **Transform Time**: Average time required to transform the original query using a particular method.
- **Search Time**: Average time taken to perform the actual search after query transformation
- **Total Time**: Transform Time + Search Time
- **Efficiency Score**: Ratio of F1 score to total processing time, measuring quality of results per unit time.

Table 11. Efficiency Metrics

Method	Transform Time	Search Time	Total Time	Eff. Score
rag_fusion	0.997 sec	0.129 sec	1.126 sec	0.40902
hyde	2.498 sec	0.040 sec	2.538 sec	0.16663
decomposition	1.032 sec	0.097 sec	1.130 sec	0.39915
expansion	1.158 sec	0.126 sec	1.284 sec	0.35969

- **Expansion slightly outperforms RAG fusion across all effectiveness metrics, though the differences are marginal:**
  - Same precision (0.3178), Better recall (0.9886 vs 0.9795), Slightly better F1 score (0.4619 vs 0.4608)
  - Slightly better MRR (0.7665 vs 0.7643) and Slightly better nDCG@5 (0.7460 vs 0.7428)
- **Both methods significantly outperform HyDE and decomposition**, particularly in ranking quality (MRR and nDCG).
- **RAG Fusion is more efficient than expansion**:
  - Lower transformation time (0.997 vs 1.158 seconds) and Similar search time. This implies overall Lower total time (1.126 vs 1.284 seconds)
  - Higher efficiency score (0.40902 vs 0.35969)
- **HyDE is notably less efficient** than all other methods, despite having the fastest search time.

While expansion provides marginally better retrieval quality, **RAG fusion achieves nearly identical effectiveness with approximately 12% better time efficiency**. The effectiveness differences are in the third decimal place, while the efficiency advantage of RAG fusion is more substantial. Therefore, based on our limited custom dataset evaluation, **RAG fusion would likely be the preferred choice** due to its better balance of effectiveness and efficiency.

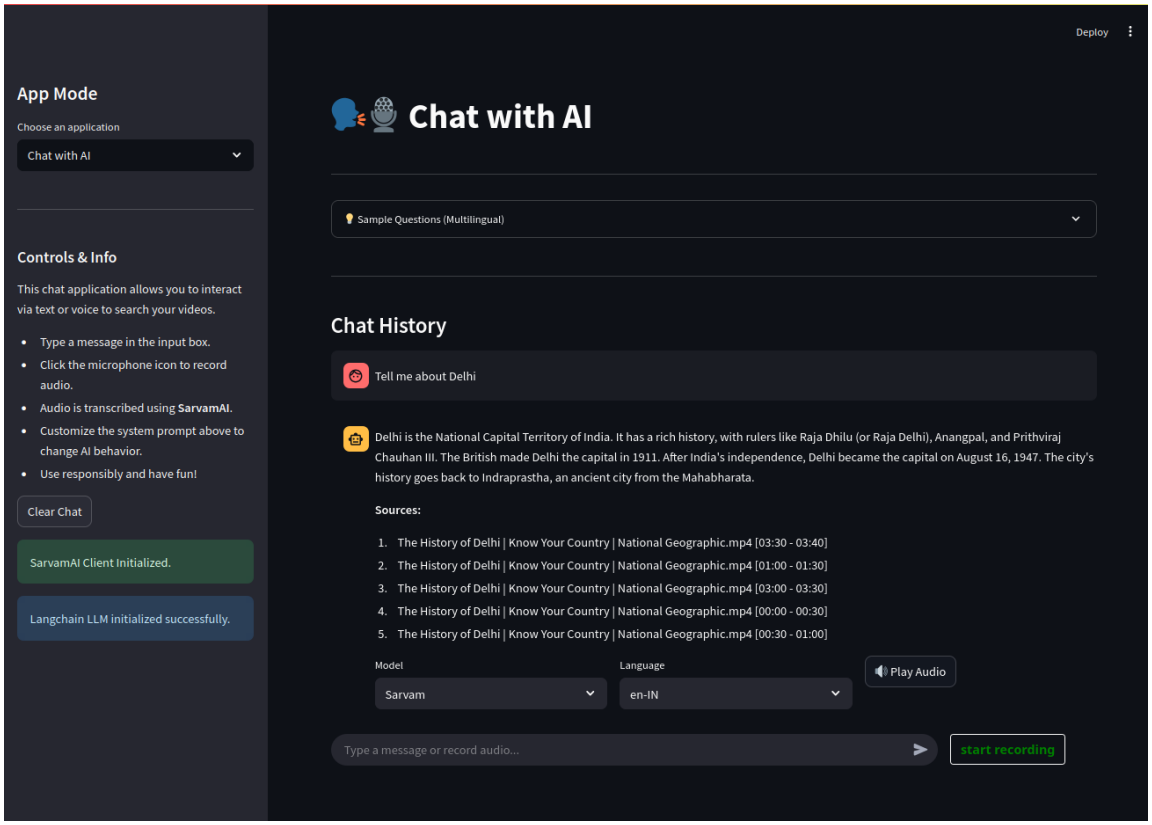

Figure 6. Chat App UI with Text input and Text +Voice output

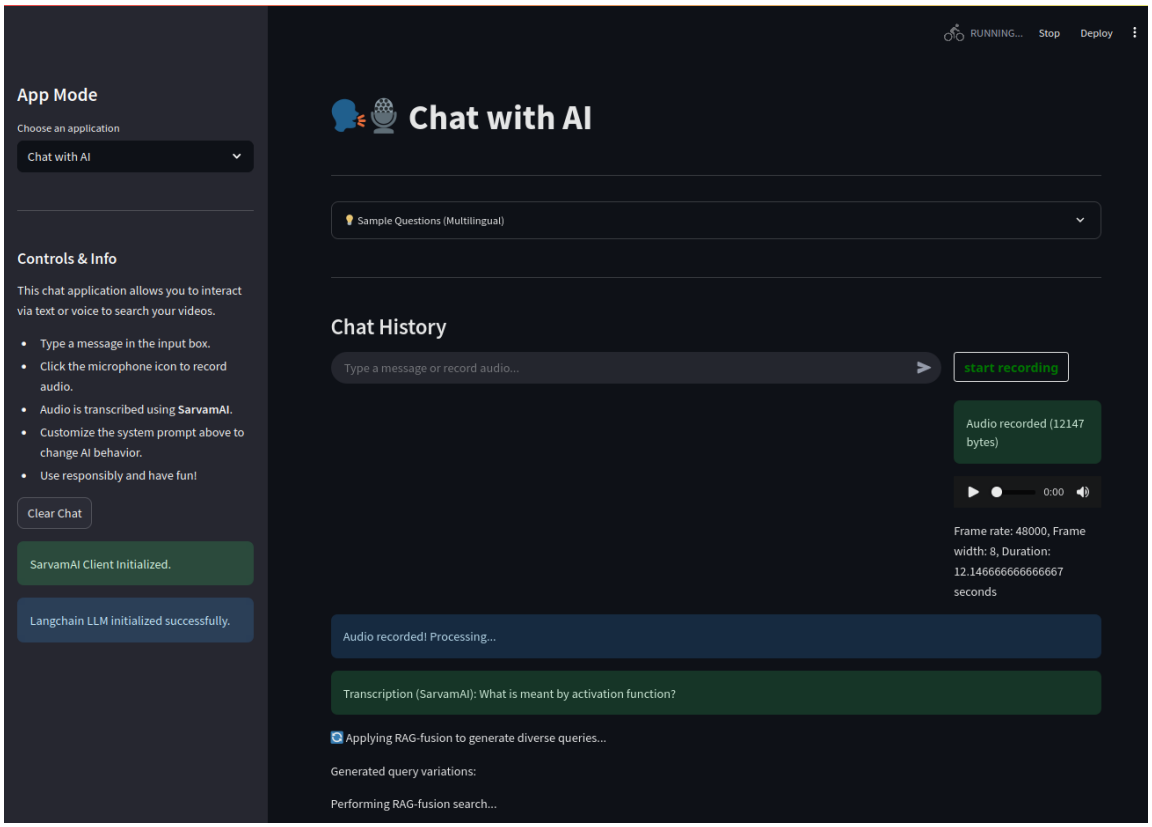

Figure 7. Chat App UI when Voice Input is given

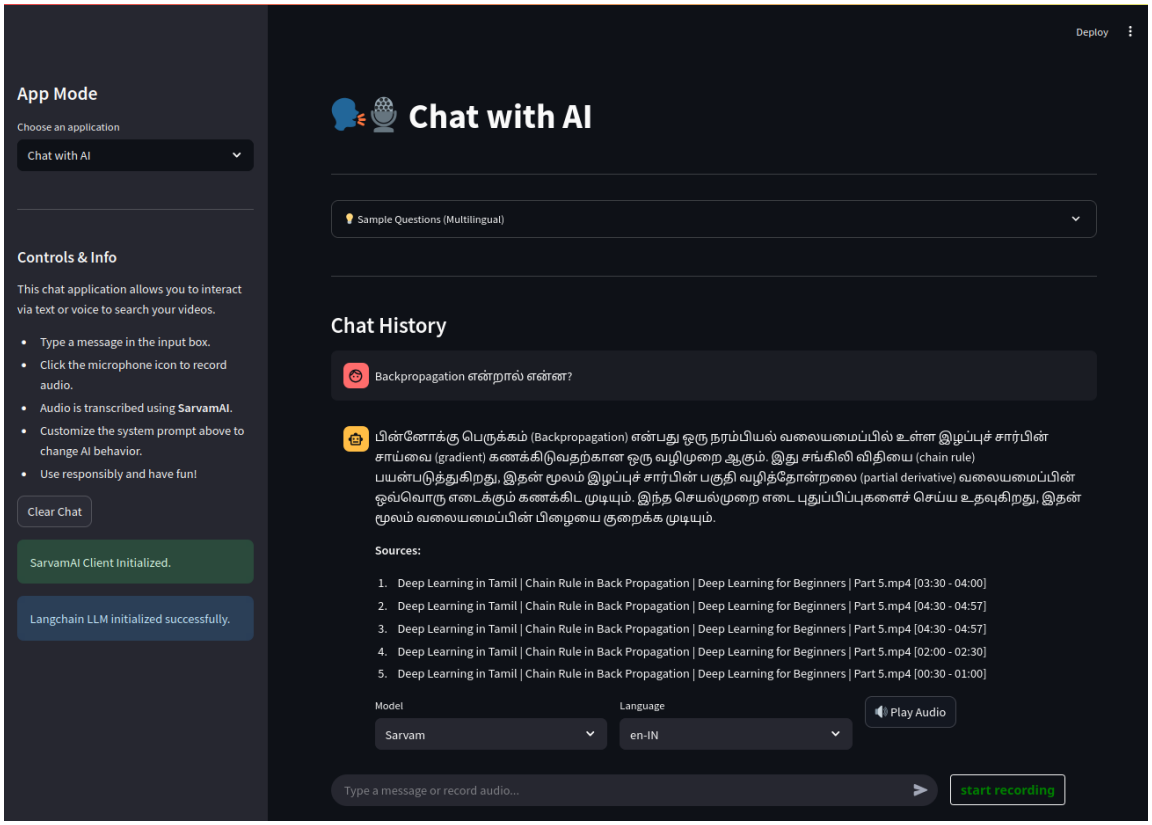

Figure 8. Chat App UI with Indic+English Code-mixed Input and Text +Voice output

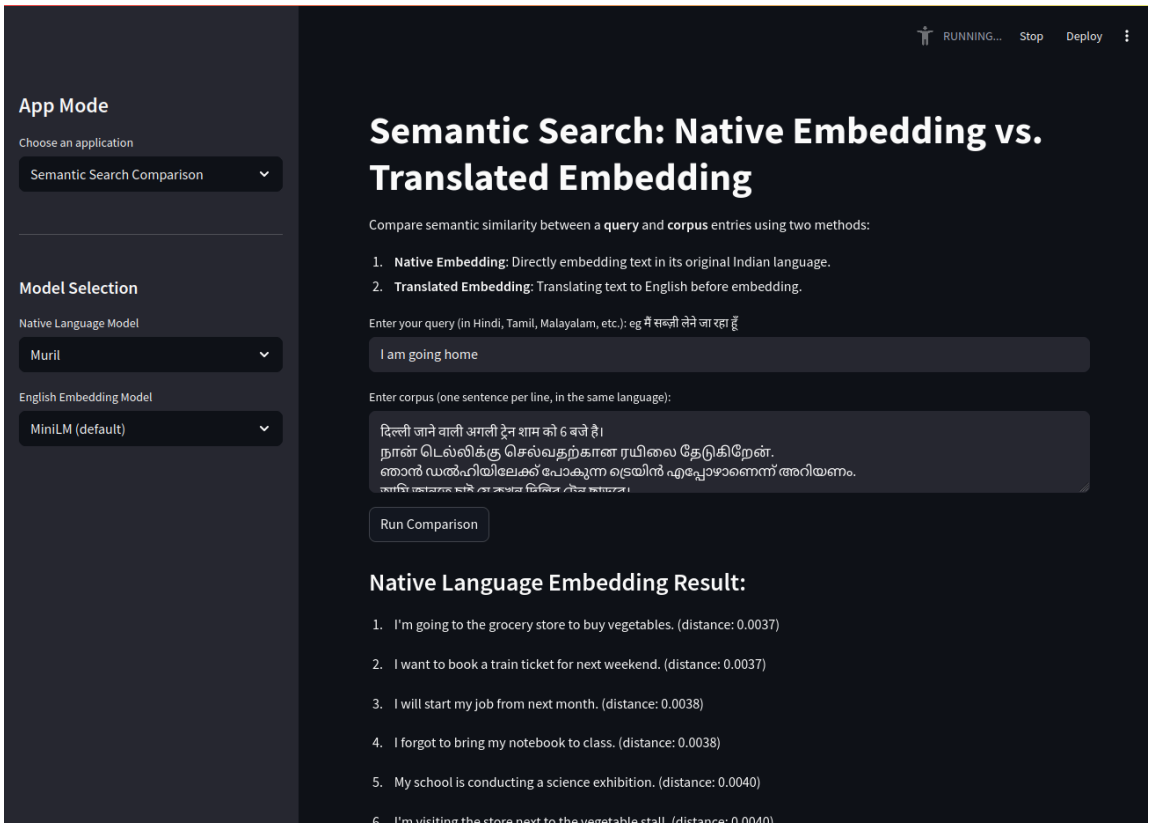

Figure 9. Companion App for Embedding Model Comparison