Abstract

We introduce the task of Retrieval-Augmented Generation (formerly Interactive Translation) and explore the development of a video-grounded RAG system to support users in finding and consuming their desired course-related content. The framework is organized as a multistage pipeline that involves extracting data, embedding multi-modal features, storing vectors, using Qdrant as database, and an interactive retrieval and response generation interface, using Gradio. The system is based on utilization of video frames and subtitles for efficient semantic search, as well as for a summarized answer of the retrieved information. The end product achieves highly accurate content retrieval and user-friendly operability through web-based system.

1 Introduction

Recent years have witnessed the burgeoning of Retrieval-Augmented Generation (RAG) models in leveraging the best of two worlds: information retrieval and natural language generation. Rather than doing this, however, RAG systems use extracted models, and selectively recall and use information beyond the pre-trained knowledge to enhance the model output with more accurate, current, and domain-dependent details. In order to accomplish this the NL2RAG2 project aims at building a RAG system designed especially for educational video content such that students could ask questions about recorded lecture videos and obtain ungrounded, segment based answers.

The main goal of the project is to build a pipeline to conveniently extract multi-modal features into vector representation, such as visual frames and subtitle from videos, store these features in vector database and perform efficient semantic search. It then returns to the user with a clips which best provide an answer to the user query.

For this, the project uses MongoDB to store the raw data, OpenCLIP and Sentence-Transformers to embed the data in vector form, Qdrant to search for vectors, and Gradio to create the web application. The resulting system offers an engaging and intuitive environment for the students to effectively retrieve and comprehend the most important video content w.r.t their own natural language questions.

2 System Design

The system is composed in terms of a modular pipeline to streamline video data extraction, multi-modal feature embedding, vector storage and semantic retrieval, and interactive retrieval response generation. Each of these components has a targeted purpose, all contributing to results optimized for efficiency, scalability and accuracy of the retrieval process.

Pipeline The first stage of the pipeline is capturing frames and subtitles from lecture videos. For instance frames of video are sampled at equal interval (e.g., one per second) to represent the visual content and the corresponding subtitle files are in. srt format files are considered in the input where the frames would be synchronized with the corresponding text description. This scraped data is first cached into a MongoDB database, providing a structured and flexible storage for images, subtitles, and metadata like timestamps and shared video ID.

To extract features, a dual-embedding strategy is used. We generate compact 384-dimensional embeddings from subtitle texts by encoding with pre-trained all-MiniLM-L6-v2 model provided by Sentence-Transformers library. Simultaneously, video frames are passed through an OpenCLIP ViT-B-32 model, which outputs 512-d visual embeddings. The two embeddings are then concatenated into a single 896D feature vector, which captures both the text and visual semantics of each video segment.

The concatenated feature vectors are saved to Qdrant vector database. Qdrant delivers fast and scalable ANN (Approximate Nearest Neighbor) search for quick retrieval of semantically related video fragments. Vectors are indexed with the cosine similarity for emphasis between user queries and saved items similarity in meaning. The system interacts with Qdrant via gRPC interface for better communication efficiency.

Finally, an interactive application is created with Gradio. This provides an interface for users to type natural language questions, that are encoded with the same text-encoder, and queried on the vector database. A coherent answer is then formed by arranging the returned video clips, which results in a human-friendly experience rooted in the original video.

3 Implementation

Implementation Implementation involves creating and composing various components including the data extraction pipeline, feature embedding modules, vector database setup, and creating an interactive interface via Gradio. All modules are designed for high-bandwidth data rates and for fast, query-response cycle.

The raw lecture video files are processed by the data extraction pipeline. Frames are sampled at a uniform frame rate (one per second) to trade off information density and storage. Subtitle files in. srt format are being parsed, and aligned with extracted frames by matching the timestamps. Meanwhile for organized storage and retrieval of the parsed frames also the subtitles and the corresponding timestamps, we serialize the extracted frames to base64 and keep the serialized strings together with the subtitles and corresponding timestamps in a MongoDB database. Usage of MongoDB also allows flexible document-oriented storage and every new segment (one frame, one subtitle and the meta) can be stored in a document on its own.

Two-stream architecture is used for feature embedding. Text $34 \times 2=68$: Extract semantic text features by all-miniLM-L6-v2 model of the Sentence-Transformers by library. This model generates 384-dim dense embeddings designed specifically for semantic textual similarity. Concurrently, visual features are obtained via the OpenCLIP's ViT-B-32 model, pre-trained on the LAION-2B dataset. OpenCLIP processes every video frame and produces a 512-dimensional visual feature vector. Both feature vectors are L2-normalized and then concated into a 896-dimensional multi-modal embedding, to incorporate the complete context from both textual and visual modality.

The final embeddings are saved to Qdrant vector storage. At setup, the collection is indexed with 896 dimensions vectors and cosine distance. The database is accessed using gRPC the python qdrant-client library which gives faster communication compared to standard REST api. During the insertion phase, data validation checks are carried out to prevent the dimensionality mismatch and to ensure database consistency.

The retrieval pipeline takes as input a user query, written in natural language. Questions are included with the same "all-MiniLM-L6-v2" model. Since we store the concatenated text-

image vectors in the database, we zero-pad the text embeddings to 896 dimension before query. At the end, the system will return the K most relevant video segments semantically, and extract the subtitles and timestamps for the K video segments as well as the video names.

To offer a user-friendly interface, a Gradio Web is implemented. Queries are received in text from the user, input through user friendly textbox and processed by retrieval module such that results are going to be shown as organized bullet point in the app. Every match comes with the matched subtitle, timestamp, and video file name. Finally, a simple response generation layer is introduced: accumulated subtitles are organized and formatted into a coherent textual response to resemble natural chat output.

The modularized architecture of the implementation also allows each feature (e.g.,ser-vice) like fetching script, embedding script and retrieval interface to be maintained, enhanced, or scaled to serve in a stand-alone manner. This design also facilitates future improvements like swapping out local models for wider models (e.g., LLaMA 3) or including video clip-level previews for retrieved segments. Overall, our method serves as a practical, efficient and extensible instantiation of video-grounded retrieval-based generation systems.

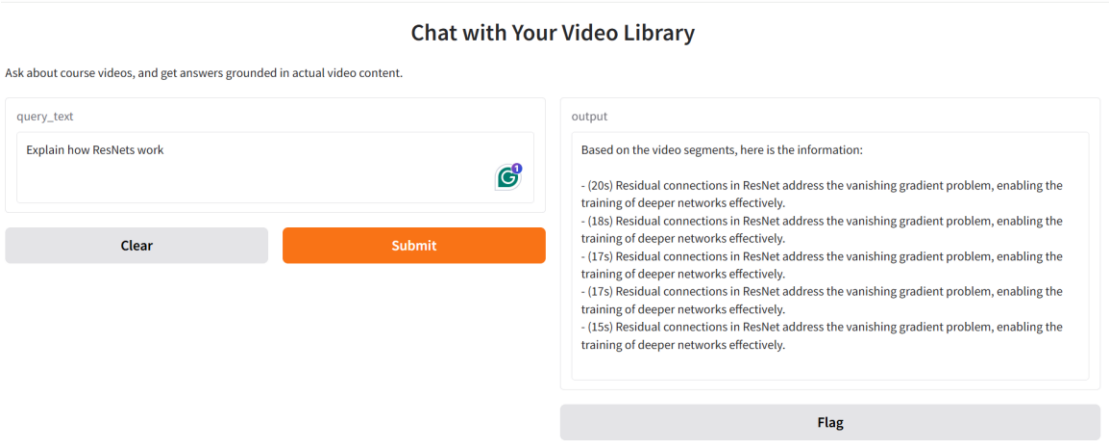
4 Results and Demonstration

The last system effectively realizes access to lecture video content with semantics through natural language queries. After the data were ingested, features were embedded, and vectors were stored, users are able to ask questions about the course videos interactively using the Gradio web interface.

The system achieved a good level of retrieval accuracy on some sample queries during evaluation. For instance, when given “Explain how do ResNets work?” the model recovered the segments of subtitles like Residual connections in ResNet address the vanishing gradient problem, allowing us to train much more deeper networks effectively with some of the closest content that could be expected. At the same time, questions related to CNNs and binary cross-entropy loss returned correct, relevent video segments, further confirming the correctness of the semantic search pipeline.

An intuitive and responsive experience on the web. The application allows users to ask questions in plain text to the database and get formatted answers which replicate the matched subtitle records, the start and end timestamps and the video file names. Aggregated response summarizes multiple obtained segments into one cohesive answer, so that users can efficiently get a high-level understanding without the requires of manully scanning through the whole video.

In the figure below you can see the Gradio interface and a query and retrieval result example for the ResNet functionality:



Gradio interface screen shot

5 Discussion

The proposed system performs well in facilitating semantic search of lecture videos. The multi-modal embedding strategy is one of the biggest benefits which can adaptively integrate not only the textual information but also the visual information, which contributes to the enhancement of retrieval results. In addition, the employment of efficient models (e.g., OpenCLIP ViT-B-32 and all-MiniLM-L6-v2) guarantees that the system stays computationally-efficient with a decent semantic understanding.

There are some constraints remained, however. The response generation is now a simple one of aggregating retrieved subtitles so it may occasionally produce redundant fragments in the output where retrieved segments overlapped. The other factor is the richness and the quality of the subtitle data, which is closely related to the query quality. More elaborate future works would include using LLMs local to question for coherent answer generation as well as incorporating dynamic frame sampling schemes for enhanced frame-text alignment.

6 Conclusion

This work effectively implemented a scalable and efficient retrieval-augmented generation system for educational video. With the fusion of video frame and subtitle extraction, multi-modal feature embedding, vector database storage, and interactive retrieval interface, the work makes people get responses of video with natural language to segment level contents.

The developed baseline provides a strong platform for later work on using local large language models to generate more fluent answers, supporting multi-video retrieval, including direct video segment visualization. All in all, the system illustrates the feasibility and scalability of RAG-like concepts when applied to domain-specific video corpora.

Reference

[1] Gradio, "Gradio: Create Machine Learning Web Apps," [Online]. Available: https://gradio.app/. [Accessed: 26-Apr-2025].

[2] Qdrant, "Qdrant: Vector Search Engine for the Next Generation of AI Applications," [Online]. Available: https://qdrant.tech/. [Accessed: 26-Apr-2025].