Final Assignment: Retrieval-Augmented Generation (RAG) for Video Question Answering

LLMs and RAG Systems

1 Objective

This assignment focuses on designing and evaluating a multimodal Retrieval-Augmented Generation (RAG) system that enables users to query a video and receive relevant video segments in response. You will integrate audio and visual content from a video, apply semantic and lexical retrieval techniques, and build a user-friendly Streamlit interface. Your system should respond to user queries by either:

- Playing the video where the answer can be found, or
- Stating clearly that the answer is not present in the video

The video used in this assignment is:

`https://www.youtube.com/watch?v=dARr3lGKwk8`

2 Multimodal RAG System Overview

You must design a retrieval system that uses both **audio (speech)** and **visual (frames)** information from the video. The goal is to find and return the video segments most relevant to a user's natural language question.

Your system must include:

- **Speech-to-text transcription** using models like OpenAI Whisper (open source) or similar.
- **Text embeddings** generated from transcript segments using open-source models.
- **Image embeddings** generated from sampled keyframes using open-source vision-language models (e.g., CLIP).
- A semantic search engine over these embeddings using vector databases.
- Lexical retrieval baselines using TF-IDF and BM25.

3 Required Components

1. Streamlit Application

Build an interactive UI using Streamlit. It must include:

- A text input for the user to submit a natural language question.
- A returned result that includes:
  - Timestamp(s) in the video where the answer appears.
  - A view of the corresponding video segment.

  - If no relevant result is found, an appropriate message indicating the answer is not present in the video. Also allowing for further questions to the interface. It should be a chatting interface basically.

- The video should be either embedded in the interface.

2. Transcript and Frame Extraction (you don't have to follow exactly)

- Extract the full audio transcript using a speech-to-text model.
- Segment the transcript (e.g., into 10-second or paragraph-based chunks), storing the start time for each.
- Sample keyframes at regular intervals (e.g., every 2–5 seconds or per scene change).
- Associate each keyframe with a timestamp and a corresponding text segment if possible.

3. Embedding Models (Open-Source Only)

- You must use only open-source embedding models.
- For text: Select a model from the MTEB leaderboard.
- For images: Use models like `openai/clip-vit-base-patch32` or any publicly available visual encoder.
- Justify your model choices in the report, considering MTEB benchmarks, resource constraints, and retrieval relevance.

4. Retrieval Techniques

Implement and compare the following methods:

1. **Semantic Retrieval** using:
   - FAISS (in-memory flat index)
   - PostgreSQL with pgvector using:
     - IVFFLAT index
     - HNSW index
2. **Lexical Retrieval** using:
   - TF-IDF
   - BM25

You may optionally experiment with multimodal fusion (e.g., combining text and image similarity scores).

5. Gold Test Set

Manually construct a gold standard test set:

- 10 questions that are directly answerable from the video. For each, specify the ground-truth timestamp where the answer occurs.
- 5 questions that are unanswerable (i.e., they seem plausible but the video does not cover them).

This dataset will be used to evaluate accuracy and rejection performance.

6. Evaluation Criteria

For each retrieval method, evaluate:

- **Accuracy on answerable questions** (correct timestamp in top-1 result).
- **Rejection quality on unanswerable questions** (no false positives).
- **Latency**: Average query time across multiple runs.

Create a comparison table and/or graph in your report. Reflect on:

- Tradeoffs between accuracy and speed.
- Effectiveness of visual vs. textual modalities.
- Failure cases and proposed improvements.

4 Deliverables (All Mandatory)

1. **Streamlit application**:
   - Clean and easy to use.
   - Clearly shows result segment(s) and handles no-answer cases.
2. **Codebase (on git)**:
   - Organized and well-documented.
   - Includes setup instructions and dependencies (e.g., requirements.txt).
   - Gold test set with answers and timestamps.
3. **Video Demonstration (maximum 10 minutes)**:
   - Explain your pipeline clearly: data preparation, embedding, indexing, retrieval.
   - Show the Streamlit UI in action.
   - Demonstrate answering a few questions from the test set.
   - Briefly summarize results and insights.
   - Embedding model justification (refer to MTEB leaderboard).
   - Retrieval method comparison (FAISS, pgvector-IVFFLAT, HNSW, TF-IDF, BM25).
   - Quantitative evaluation results.
   - Observations and analysis.
   - Make the video concise and well-edited. It must stand on its own as a summary of your system.

5 Bonus Challenges (Optional)

- Compare the outputs of different LLMs.
- Try score fusion across modalities (e.g., average text and image embedding similarity).
- Implement UI features like "Explain why this was retrieved" or visual highlights.
- Fine tune a "small" LLM on text extracted from the video and compare the performance of fine-tuning to RAG (that's a big bonus :))

We encourage creativity and thoughtful design. The best submissions will not only work well, but also demonstrate careful reasoning and clear communication.