Project III | AI ChatBot for YouTube Video QA and summary

Nataliya Demyanenko

Table of contents

01

Project Scope

Project Scope

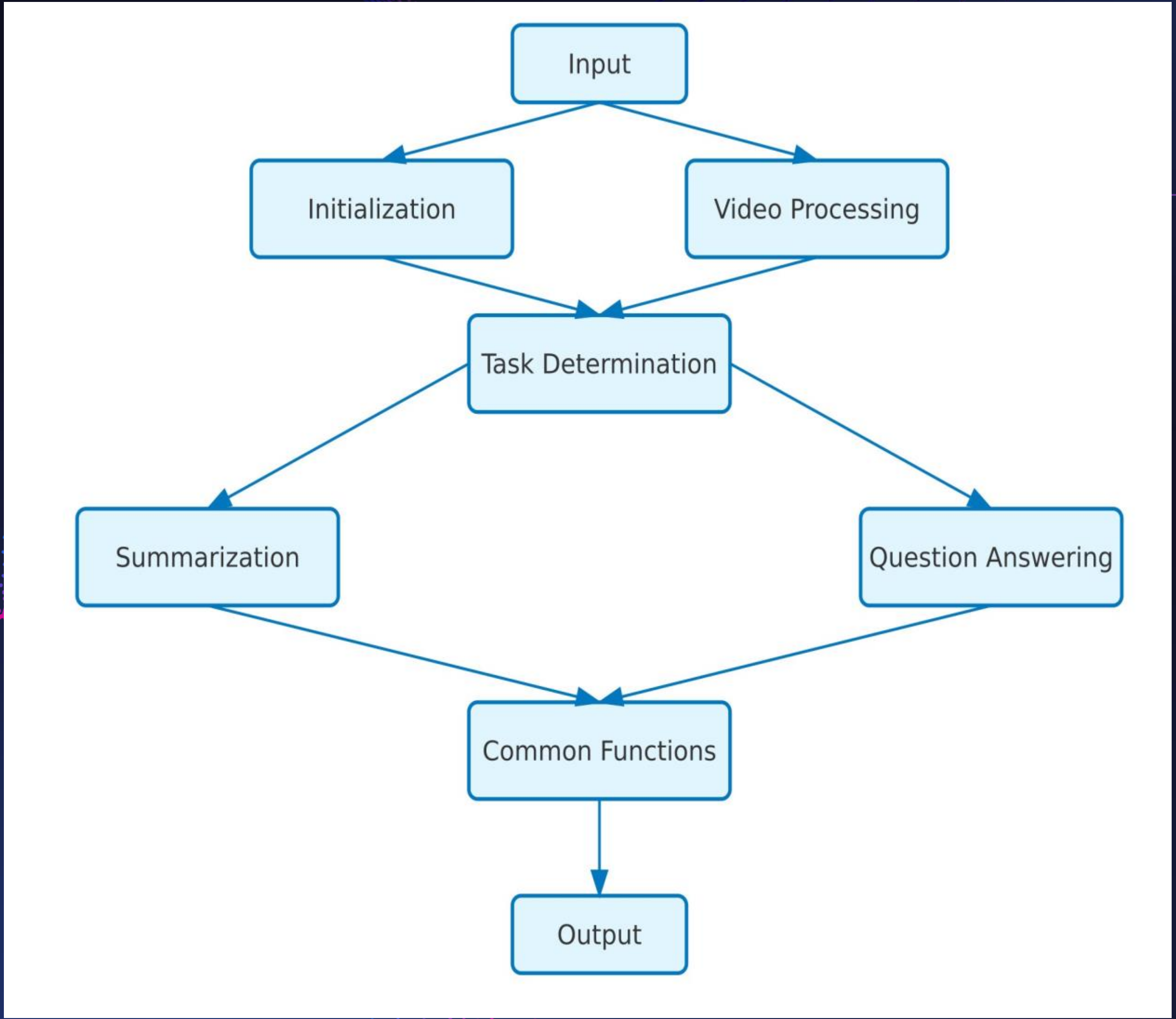
Goal: Build a chatbot for YouTube video summarization and Q&A

Focus areas:

- Exploration of open-source solutions
- Optimization for CPU deployment
- Abstractive question answering
- Real-life processing of user-provided videos

02

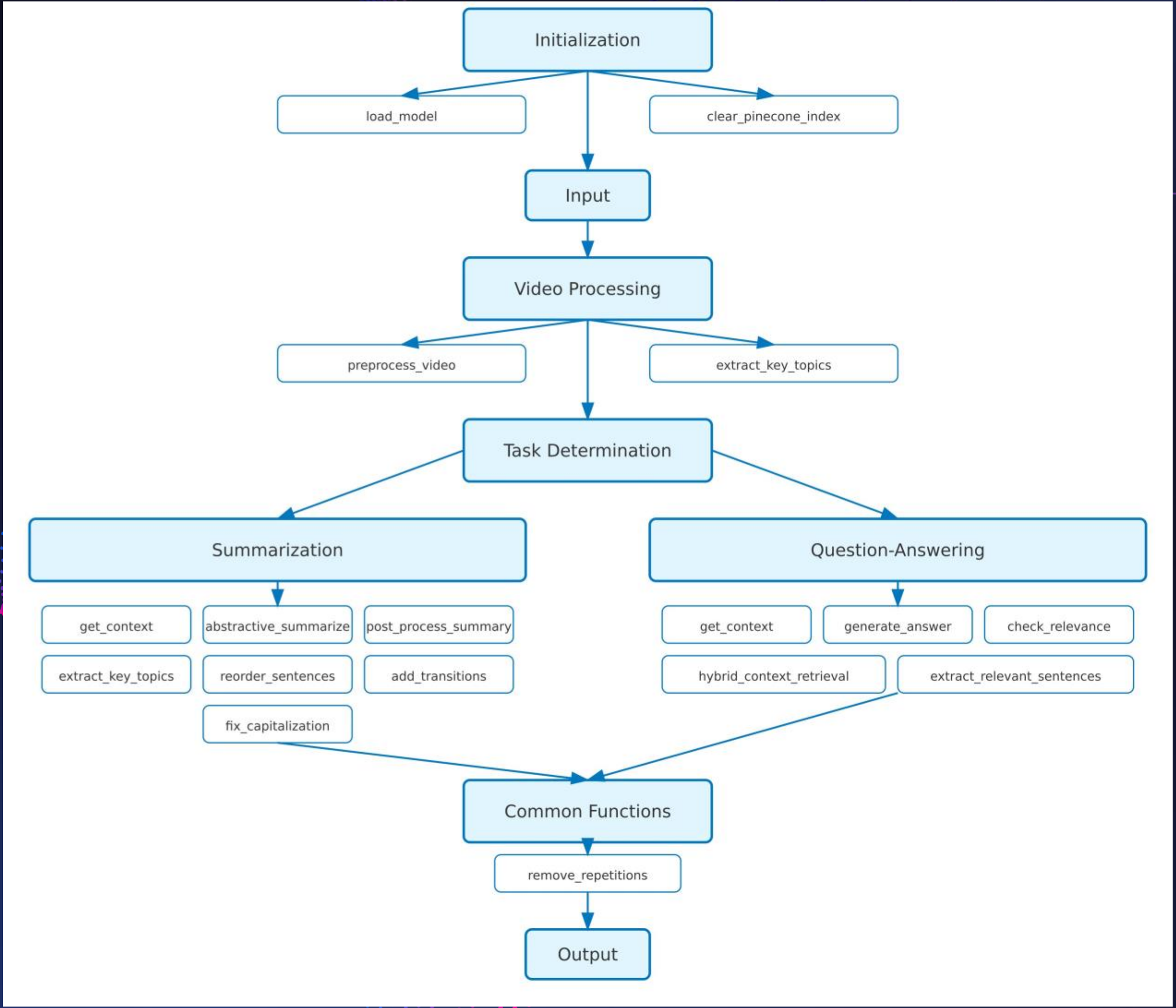
App



General overview of the app

1. **Initialization**: model loading and index clearing.
2. **Video Processing**: Extraction and chunking of video content.
3. **Task Determination**: Q&A or summary
4. **Summarization**: Multi-step process involving context retrieval, abstractive summarization, and post-processing.
5. **Q&A**: Utilizes hybrid context retrieval and relevance checking for accurate answers.
6. **Common Functions**: Post-processing for output quality.

All functions



Video Preprocessing

URL upload

- Extracts video information and transcript
- Splits the content into manageable chunks
- Upserts to Pinecone

```
video_data.append({
                    'chunk_id': chunk_id,
                    'video_id': video_id,
                    'title': info['title'],
                    'channel': info['uploader'],
                    'description': info['description'],
                    'duration': info['duration'],
                    'start_time': combined_start_time,
                    'end_time': combined_start_time + combined_duration,
                    'source': f"https://www.youtube.com/watch?v=
                                {video_id}&t={int(combined_start_time)}",
                    'chunk_text': text,
                    'metadata_context': f"Video Title: {info['title']}\n"
})
```

Task Determination

Function: determine_task()

Purpose: Decide between summarization and Q&A

```
 # List of phrases that indicate a summary request
summary_phrases =
["summarize", "summary", "summarize the video", "what is this video about", "what is the video about", "what is the main topic of the video", "what's the video about", "what's the video about", "give me an overview", "brief overview", "overview", "main points", "key points", "main idea", "what is the video discussing"]
```

Process:

- Checks user input against predefined summarization phrases
- Classifies as 'summarization' or 'question_answering'

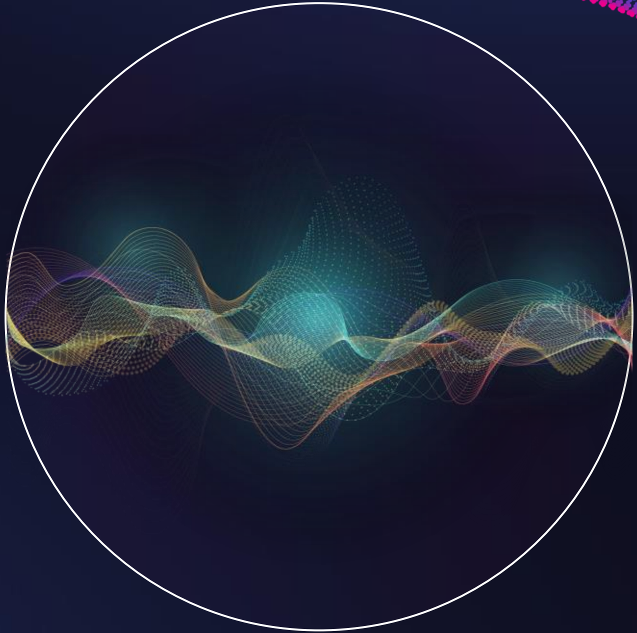
Advantage: Easily expandable for future task types

Model Selection for Q&A

Selected model: VBlagoje/BART-LFQA

Reasons for selection:

- Advanced question-answering capabilities
- State-of-the-art language model (BART architecture)
- Pre-trained on large datasets
- Customization and fine-tuning potential
- Optimized for long-form answers



Abstractive Summarization

Model Used: facebook/bart-large-cnn

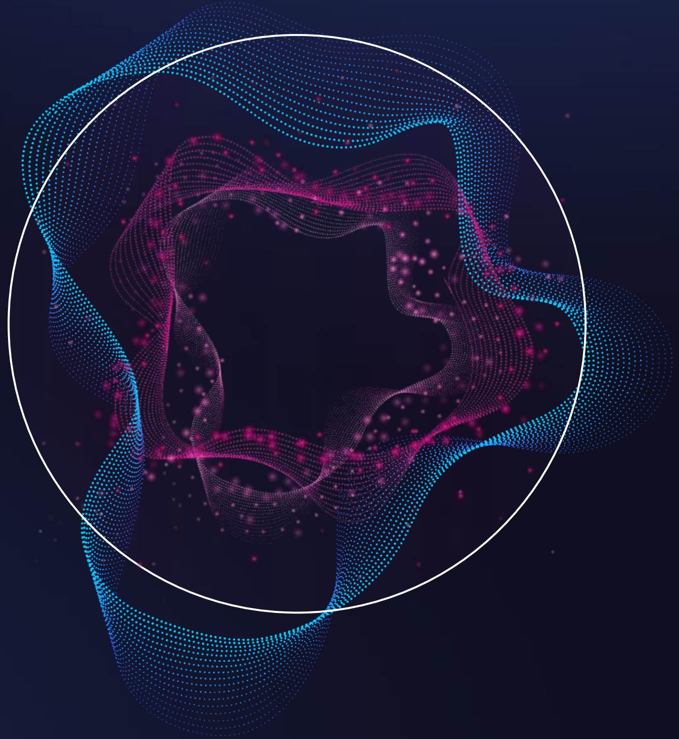
Function: abstractive_summarize()

Process:

- Prepend key topics to guide summary
- Use beam search, length penalties, and sampling

Parameters:

- Beam size, max/min length, length penalty, etc.



Hybrid Context Retrieval

Function: hybrid_context_retrieval()

Combines TF-IDF and semantic similarity for more accurate context retrieval.

- Compute TF-IDF scores
- Generate semantic embeddings
- Combine scores with weighted average

Advantage: Balances keyword matching and semantic understanding

```
def hybrid_context_retrieval(query, contexts, top_k=7, tfidf_weight=0.4):

# TF-IDF
  vectorizer = TfidfVectorizer(stop_words='english')
  tfidf_matrix = vectorizer.fit_transform([query] + contexts)
  tfidf_scores = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:]).flatten()

# Semantic Similarity
   query_embedding = retriever.encode([query])
   context_embeddings = retriever.encode(contexts)
   semantic_scores = cosine_similarity(query_embedding, context_embeddings)[0]

# Combine scores
  combined_scores = 0.5 * tfidf_scores + 0.5 * semantic_scores
  top_indices = combined_scores.argsort()[-top_k:][::-1]
  return [contexts[i] for i in top_indices]
```

Post-Processing and Formatting

Key Functions:
- **post_process_summary()**
- **remove_repetitions()**
- **add_transitions()**
- **fix_capitalization()**

Purpose: Enhance readability and coherence
Techniques:
- Sentence reordering
- Redundancy removal
- Proper capitalization
- Transition phrase insertion

03

Other Steps

Memory implementation

Type: ConversationBufferWindowMemory

Purpose: Maintain context of recent interactions
Implementation:

- Set window size (k=3)
- Integrate with ConversationChain
- Implemented in Jupyter Notebook, not in deployed app

Advantage: Enables more coherent multi-turn conversations

Fine-tuning with PEFT-optimization & Quantization

Due to time constraints, computational limits, and the size of the model being fine-tuned, I had to drastically limit training parameters, which resulted in a poor performance:

	vblagoje/bart_lfqa	Model fine-tuned on SQUAD
Faithfulness	0.653	0.484
Context Recall	0.578	0.578
Relevance	0.110	0.110
ROUGE-1	0.110	0.110
ROUGE-2	0.056	0.056
ROUGE-L	0.110	0.110

I attempted to fine tune the model on two datasets, the Stanford Question Answering Dataset (SQuAD) and ELI5-Category dataset, that both seemed promising fit.

04

Conclusion

Conclusion and Future work

Key Takeaways:

More cautious approach to project scope definition

Future Improvements

- Processing videos without YouTube transcript
- Fine-tuning both Q&A and summarization models
- Advanced sentence reordering in summarization
- Integration of sophisticated NLP techniques

Thanks!