1. Project Introduction

AI question-answering systems have always been a topic of interest in the field of education. With the continuous development of technology, I have decided to develop a ChatVideo AI question-answering system based on generative artificial intelligence models, aiming to provide an efficient and convenient knowledge acquisition tool for educators and students.

Today, the field of education faces many challenges and opportunities. Students' demand for knowledge acquisition is increasing, and educators need better tools to meet their needs. Traditional teaching methods often fail to meet the requirements of personalized learning, and our AI question-answering system will fill this gap. There is an increasing demand in the market for efficient and intelligent educational tools. Our system aims to provide an intelligent question-answering platform through deep learning models and natural language processing techniques, supporting students in quickly obtaining accurate answers during the learning process. Compared to existing solutions, our system has unique advantages. Firstly, I use generative artificial intelligence models, which enable the system to understand and generate natural language. Secondly, our system supports the integration of multiple data sources, including videos, search engines, and statistics. Users can obtain information from multiple perspectives and provide more comprehensive and in-depth answers through chain reasoning. Finally, we will continuously optimize the system's performance indicators with the aim of improving user satisfaction and providing a better learning experience for educators and students.

The goal of ChatVideo is to develop an efficient and accurate AI question-answering system that meets the knowledge acquisition needs of educators and students, and provides an intelligent question-answering platform that supports personalized learning and autonomous knowledge exploration. I will evaluate the success of the project through quantifiable performance indicators and user satisfaction. Through this project, I hope to bring innovation and change to the field of education, improving learning outcomes and teaching quality.

2. Literature Review

2.1. Evaluation of the existing solutions

2.1.1. Skipit.ai

Skipit.ai is an innovative AI-driven tool crafted to augment productivity by offering succinct summaries of diverse digital content, thereby enabling users to economize time and boost their efficiency in information consumption. Utilizing advanced AI technology, Skipit.ai adeptly condenses a broad array of content, including YouTube videos, Google docs, PDFs, and web materials, into brief, chat-based overviews. This feature is especially beneficial for those seeking to quickly comprehend the crux of extensive content without the need to peruse the entire material. Designed for ease of use, the tool is remarkably user-friendly, allowing users to simply upload a PDF or enter the URL of a desired video, document, or website for summarization. The AI swiftly processes this input, generating a summary that encapsulates the essential information, thus obviating the need for extensive scrolling or viewing lengthy videos. Skipit.ai's versatility is further underscored by its capability to summarize various content types, ranging from YouTube videos to websites and articles, making it an indispensable tool for individuals interacting with different digital media formats. The primary objective of Skipit.ai is to conserve users' valuable time while ensuring they receive the pertinent information, thereby significantly enhancing productivity and efficiency. Additionally, its user-friendly interface, coupled with an integration with ChatGPT, offers a streamlined and intuitive user experience, further elevating its utility. In essence, Skipit.ai emerges as an invaluable asset for those seeking to optimize their time and efforts in the consumption of digital content, providing rapid and concise summaries of a multitude of media types.

2.1.2. Tammy.ai

Tammy.ai is an AI-powered platform meticulously crafted to transform the YouTube viewing experience. It leverages artificial intelligence to provide a suite of features aimed at enhancing user engagement and content comprehension. One of its flagship features is the AI-Driven Summarization, which enables users to generate concise summaries of various YouTube videos, including tutorials, job interviews, and philosophical lectures. This functionality is especially beneficial for those seeking to grasp the essence of videos without dedicating time to watch them in full.

Furthermore, Tammy.ai enriches the YouTube experience by introducing an interactive dimension. This includes AI-powered influencer chat and the novel capability to 'chat' with video content, thereby fostering a more engaging and educational interaction with the platform. In terms of content organization, Tammy.ai excels by automatically segmenting YouTube videos into episodes, crafting succinct descriptions, and systematically arranging timecodes. This meticulous organization facilitates the navigation of lengthy videos and the efficient access to specific segments.

Tammy.ai encompasses a range of innovative AI products, such as Tammy Summary, Tammy Chat, and Tammy Influencer. These proprietary tools synergistically work to revolutionize user engagement with online video content. On the accessibility front, Tammy.ai offers a complimentary plan with limited daily credits and a premium plan, available either monthly or annually. The premium plan boasts additional features like an ad-free interface and unlimited basic summaries. In essence, Tammy.ai is a groundbreaking tool that employs AI to redefine the way users interact with and consume YouTube content, offering efficient summarization and interactive features to elevate the online video experience.

2.1.3. Harpa.ai

Harpa.ai is a Google Chrome extension and a NoCode Robotic Process Automation (RPA) platform that stands out as a versatile AI solution, offering a multitude of functionalities to automate web tasks and enhance productivity for various user groups. At its core, Harpa.ai integrates a hybrid AI-engine that merges the capabilities of GPT models, such as ChatGPT and ClaudeAI, with web automation, facilitating a wide range of efficient task executions.

This platform excels in summarization and data extraction, making it highly beneficial for researchers by streamlining their process through efficient aggregation and synthesis of information from diverse web sources. For business professionals, Harpa.ai automates repetitive web tasks, leading to significant productivity boosts and cost savings, which is vital for businesses aiming to optimize operations and minimize manual intervention in routine web activities.

Additionally, marketers find Harpa.ai invaluable for market monitoring, as it enables them to keep a close eye on competitor sites, track product prices, and stay updated with stock availability, thus maintaining a competitive edge by swiftly adapting to market changes and trends. The platform also includes a "Composer" feature, which aids users in generating and rewriting content for various platforms, including social media and emails, thereby facilitating the creation of compelling and relevant content with efficiency.

2.1.4. Compare Skipit.ai and Tammy.ai and harpa.ai

In evaluating current AI Q&A solutions, a comparative analysis reveals distinct functionalities, performances, and applicable scenarios across various platforms. Skipit.ai stands out with its capability to provide brief summaries of diverse digital contents including YouTube videos, Google docs, PDFs, and web materials. It is known for generating concise summaries rapidly, making it ideal for users who require quick insights without delving into the entire content. Despite its user-friendly interface and integration with ChatGPT, it may fall short in offering in-depth analysis and is dependent on the quality of input material.

Tammy.ai, on the other hand, specializes in AI-driven summarization of YouTube videos, complemented by interactive features like AI-powered influencer chat. It segments videos into episodes for efficient summarization, targeting users who wish to quickly grasp the essence of videos. While it enhances the YouTube viewing experience and organizes content effectively, its focus is primarily on YouTube, limiting its applicability to other digital media types.

Harpa.ai distinguishes itself as a Chrome extension and NoCode RPA platform, automating web tasks while offering summarization and data extraction capabilities. It streamlines information aggregation and synthesis from the web, proving invaluable for researchers, business professionals, and marketers. Its versatility in automating various web tasks and the integration of GPT models with web automation are notable. However, its utility is restricted to web-based content and may not perform as well with non-web materials.

In contrast, ChatVideo focuses on enhancing user interaction with video content through features like questioning about video content, multi-video retrieval, and video library management. It integrates summary algorithms, Q/A algorithms based on LLM, approximate vector matching, and clustering algorithms.

3. System Design and Implementation

3.1. Algorithm Design

3.1.1. Summary

In chatvideo, the "chat with video" feature utilizes a summary algorithm in its summary section.

3.1.1.1. Algorithm Process and Introduction

- Solution 1: Summarizing only the beginning of each paragraph

  This plan uses a specific summarization method, which involves only using the beginning of each paragraph for summarization. The specific algorithm steps are as follows: First, the text is evenly divided into several paragraphs. For example, for a text containing 10,000 characters, I divide it into 10 paragraphs, each containing 1,000 characters. Next, I extract the first 400 characters from each paragraph, accumulating a total of 4,000 characters as the summary content. These summary contents will then be used for further processing and analysis by the GPT model. This method aims to achieve effective and efficient text summarization by focusing on the core content of each paragraph.

- Solution 2：hierarchical summary

  In Solution 1, I faced a challenge of how to divide the text into 10 parts while adhering to the 4,000-word limit per summary. The initial approach was to extract only the first 400 words of each part, but this resulted in significant loss of information. To address this issue, I adopted a new strategy. First, I applied summarization individually to each paragraph and then combined these summaries together. Subsequently, I performed a comprehensive summarization on this collection of summaries to ensure the completeness and conciseness of the information. Additionally, if the combined summaries of all paragraphs still exceeded the word limit of ChatGPT, I could increase the number of summarization rounds until the final prompt met the word requirement. To maintain the semantic integrity of each paragraph, I incorporated sentence segmentation techniques and context completion methods used in question-answering algorithms when dividing the paragraphs, ensuring the coherence and completeness of each summary in terms of semantics.

  - Map prompt: This prompt is a guide on how to provide a summary of a given text. When ChatGPT receives the text, it should provide a well-organized summary. The summary should focus on the key points and main ideas of the text while maintaining clarity and conciseness. ChatGPT will understand the core content of the text and express it in a streamlined manner, removing any unnecessary details and retaining only the most important information.

    ```
    You will be given a single section from a text.
    This will be enclosed in triple backticks.
    Please provide a cohesive summary of the following section excerpt,
    focusing on the key points and main ideas, while maintaining clarity and conciseness.
    ```

```
<text>{text}</text>
```

- Reduce prompt: This prompt's task is to parse and summarize a large document. First, carefully read all the excerpts enclosed in triple backticks in the document. This is a crucial step in understanding the main idea of the document. After determining the overall main idea of the document, provide a comprehensive summary based on this main idea. During this process, synthesize the information into a well-structured, easily readable overview, similar to a concise summary article, ensuring coherence in content. When writing, avoid simply rephrasing the provided text or copying the structure of the original text. Also, be mindful of avoiding content repetition and ensure smooth connections between all points and information. Before the formal summary, create a brief, bullet-pointed list of key points to help readers quickly grasp the core content of the article. Additionally, the entire document should be written in HTML format, with the text divided into different paragraphs, each with appropriate indentation. Specifically, the text will be placed within HTML tags <text>{text}</text>, where {text} represents the specific text content. These formatting requirements not only emphasize the accuracy of the content but also highlight the importance of document format and structure to facilitate readers' understanding and absorption of the provided information.

```
Read all the provided summaries from a larger document.
They will be enclosed in triple backticks.
Determine what the overall document is about and summarize it with this information in mind.
Synthesize the info into a well-formatted easy-to-read synopsis, structured like an essay that summarizes them cohesively.
Do not simply reword the provided text.
Do not copy the structure from the provided text.
Avoid repetition. Connect all the ideas together.
Preceding the synopsis, write a short, bullet form list of key takeaways.
Format in HTML.
Text should be divided into paragraphs. Paragraphs should be indented.

<text>{text}</text>
```

- Comparison between Solution 1 and Solution 2

When comparing Solution 1 and Solution 2, I found that they each have their own characteristics. The core of Solution 1 is to extract only the beginning of each paragraph as a summary. The advantage of this approach is its low cost and fast processing speed, as it only requires one query to ChatGPT, reducing expenses and improving efficiency. Additionally, this method can provide a relatively balanced coverage of the entire text, increasing the possibility of gaining a comprehensive understanding of the content. However, it also has clear limitations, as it may miss a significant amount of information, especially when dealing with longer texts, which reduces the reading to approximately 6,000 words.

In contrast, Solution 2 adopts the hierarchical summary method. Its advantage lies in the ability to view the complete text, which significantly improves the quality and effectiveness of the summary. Unlike Solution 1, it does not miss out on a large amount of content. However, this method has relatively higher costs and slower processing speed, as it requires multiple summarizations and consumes a large number of tokens. Additionally, this method may lack necessary context when processing each paragraph, for example, if a paragraph begins with "she continued to say," ChatGPT may not accurately understand its semantics due to the lack of context. Therefore, the selection of the appropriate solution needs to consider the cost, efficiency, accuracy, and coherence of the context.

3.1.1.2. Algorithm Optimization

- Refinement

  In order to address the issue of lack of context mentioned earlier, I have used the refine method in LangChain. The algorithm steps for this method are as follows: Firstly, summarize the first paragraph of the document. Then, combine this summary with the original text of the second paragraph and submit it to GPT for a second summary. This process is then continued by combining the newly generated summary with the next paragraph of text and requesting another summary from GPT. Through this approach, GPT can refer to the previous content, thereby alleviating the lack of context to some extent.

- Paragraph Clustering using k-means

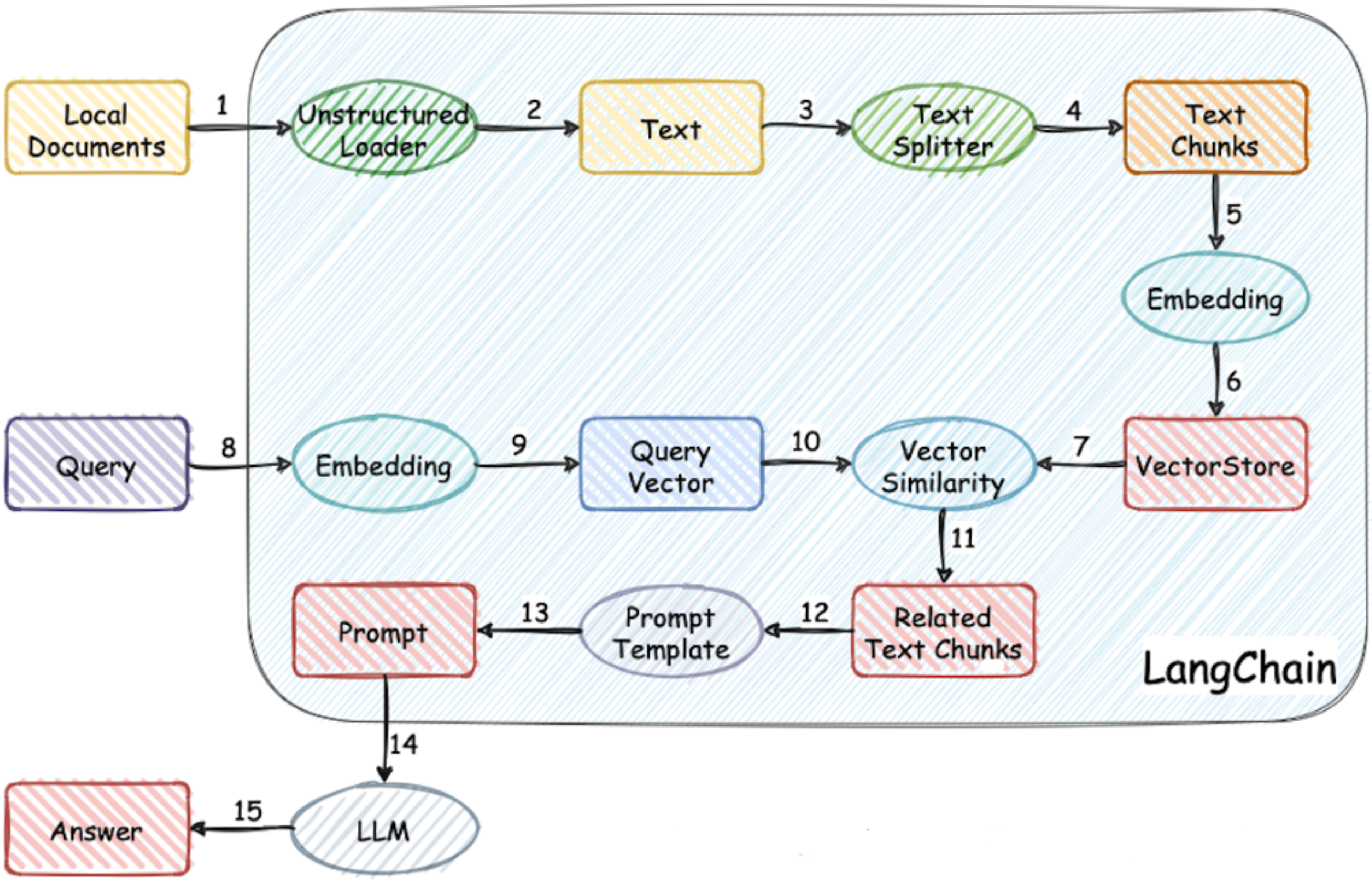
  When processing long texts, in order to reduce computation costs and improve processing speed, I have employed an optimization strategy, which is to perform pre-clustering of text paragraphs using the k-means algorithm before clustering. Although this method can effectively improve clustering efficiency, it inevitably leads to a certain degree of information loss. The specific algorithm steps include: firstly, splitting the long text into multiple paragraphs; then clustering these paragraphs, grouping together those with similar content. For example, if the subtitles of a video are divided into 300 paragraphs, the k-means algorithm can be used to aggregate them into several classes based on content similarity. I can control the number of clusters, where a higher number of clusters results in less information loss. Next, I will find the most representative paragraph in each class, i.e., the vector closest to the class centroid. Finally, I will use these most representative paragraphs in each cluster for mapreduce operations. Through this approach, my aim is to achieve both efficiency and a certain level of accuracy in processing long texts.

3.1.2. Q/A

3.1.2.1. LLM + Approximate Vector Matching

In the chatvideo application, both "chat with video" and "video retrieval" utilize LLM (Language Model) combined with approximate vector matching.

- Algorithm Process and Introduction:



1. Processing Video Subtitles: First, the algorithm extracts subtitles from the video. Then, the text is split according to characters, length, or semantics. This step allows the text content to be processed by the approximate vector matching algorithm. The reason for text splitting is that LLM has length limitations and cannot directly input a whole document. Therefore, useful information needs to be selected from it and provided to LLM along with the question.

2. Question Vectorization and Matching: When a user asks a question, it is first converted into vector form. Then, the approximate vector matching algorithm is used to find several segments in the vectorized subtitle data that are closest to the user's question vector.

   - Role of Approximate Vector Matching in the Algorithm:

     Approximate Nearest Neighbor (ANN) has a significant role in the algorithm due to its efficient and accurate information retrieval capabilities. This process starts by converting the user's question into vector form, which is achieved through a pre-trained language model that captures deep semantic features of the question and encodes them into numerical vectors. This vectorization is not just a direct mapping of keywords in the question, but a more comprehensive representation that includes contextual information and implicit meaning.

     Once the question is transformed into a vector, the approximate vector matching algorithm comes into play by quickly searching through a pre-built database containing a large amount of vectorized information to find the items most similar to the user's question vector. This similarity search is not based on simple keyword matching, but on the distance or similarity between vectors, capturing finer semantic relationships and providing more relevant and accurate results.

     In this process, the algorithm can quickly identify the most relevant information to the user's question from a large amount of data, such as relevant text paragraphs, documents, or subtitle segments. This approach is particularly suitable for handling complex or ambiguous queries, as it relies on understanding the deep meaning of the question rather than just surface text. As a result, users receive answers that are not only closely related to their questions but also based on a deep understanding of the question.

   - Reason for Not Using Non-Keyword Matching:

     The choice of using Approximate Nearest Neighbor (ANN) instead of traditional keyword matching methods to handle user queries is mainly due to its higher efficiency and accuracy, especially in understanding and processing complex semantic queries. In traditional keyword matching methods, algorithms often retrieve information based on the frequency of keyword occurrences or direct matches, which may be effective for simple and direct queries but often fail to accurately capture the complexity and subtle differences in semantics.

     In contrast, approximate vector matching can delve deeper into the semantic aspects of language by transforming text into vector form. These vectors not only contain information about keywords but also encompass the context of words, syntactic structures, and the latent meanings of language. For example, even if a user's query does not directly include specific keywords, vector matching techniques can find the most relevant information by analyzing the semantics of the entire sentence. This approach is particularly effective in handling synonyms, different expressions of phrases, or complex queries.

Furthermore, approximate vector matching demonstrates higher efficiency when dealing with large-scale data. It can quickly find items that are semantically closest to the query in a large amount of data, rather than simply searching for texts containing specific keywords. This is crucial for algorithms as they need to provide information quickly and accurately within massive data.

- Reasons for Choosing FAISS:

  In ChatVideo, I chose FAISS (Facebook AI Similarity Search) as the core tool to implement the approximate vector matching algorithm, based on several key advantages of FAISS. Firstly, FAISS is designed specifically for similarity search and dense vector clustering tasks with large-scale datasets. It can efficiently handle vectors with dimensions ranging from millions to billions, which is crucial for knowledge base question-answering systems that require fast access and processing of large amounts of data. The efficiency of FAISS stems from its optimized algorithms and data structures, enabling it to maintain fast retrieval capabilities even on massive datasets.

  Additionally, FAISS has highly optimized memory usage and search algorithms, providing excellent performance. This means that even in resource-constrained environments, FAISS can maintain fast response times, which is crucial for ensuring a good user experience in question-answering systems. Moreover, FAISS supports running on GPUs, further enhancing search speed and efficiency.

  The flexibility and ease of use of FAISS are also important reasons for its selection. It offers multiple choices of indexes and search algorithms, allowing developers to choose the most suitable solution based on specific application requirements and hardware configurations. This flexibility allows FAISS to adapt to different application scenarios and performance demands.

  Finally, as a project developed and maintained by Facebook, FAISS enjoys strong community support and regular updates. This ensures that FAISS not only continues to be optimized and improved but also promptly addresses any issues that may arise, ensuring its long-term stability and reliability.

- Cosine similarity and Euclidean distance performance comparison

  In an analysis comparing cosine similarity and Euclidean distance for performance in ChatVideo, we begin with a detailed examination of cosine similarity. Cosine similarity determines the similarity between two non-zero vectors by calculating the cosine of the angle between them. This approach focuses on the direction of the vectors rather than their magnitude, making it suitable for comparing the semantic similarity of text content, especially when the lengths of subtitle paragraphs vary. It excels in queries involving semantic understanding and topic relevance, such as conceptual or abstract questions, capturing semantic-level similarities more effectively. Cosine similarity provides more accurate matches for ambiguous or polysemous queries based on context and semantic content. However, it may fall short in comparing details due to its disregard for text length. It is less effective for queries requiring precise literal matches, like specific facts or data, where Euclidean distance might be more effective.

Euclidean distance, on the other hand, measures the straight-line distance between vectors in space, taking into account both the magnitude and direction of the vectors. It proves more accurate for queries involving concrete facts or data, such as dates, numbers, or specific events, and offers more precise matches when subtitle paragraph lengths are relatively consistent. While it has an advantage in exact literal matches, such as queries on numbers or specific facts, it may be sensitive to variations in text length and structure, affecting the accuracy of matches. It is less effective for queries with rich semantic content or abstract nature, focusing more on literal differences.

In conclusion, cosine similarity is more suitable for processing text matches with rich semantic content or varying lengths, while Euclidean distance may perform better in scenarios requiring precise matching of specific information. In ChatVideo, where multiple text segments are provided to a Language Learning Model (LLM) to answer questions, the process achieves a degree of precise matching. The LLM selects the most relevant parts of the provided text to answer queries, emphasizing the need to find semantic similarities. This approach goes beyond mere literal matching, delving deeper into understanding the meaning and context of the text, thus aligning more closely with the user's queries in concept or theme. Through this method, the LLM can provide richer, more precise answers, better fulfilling the users' information needs.

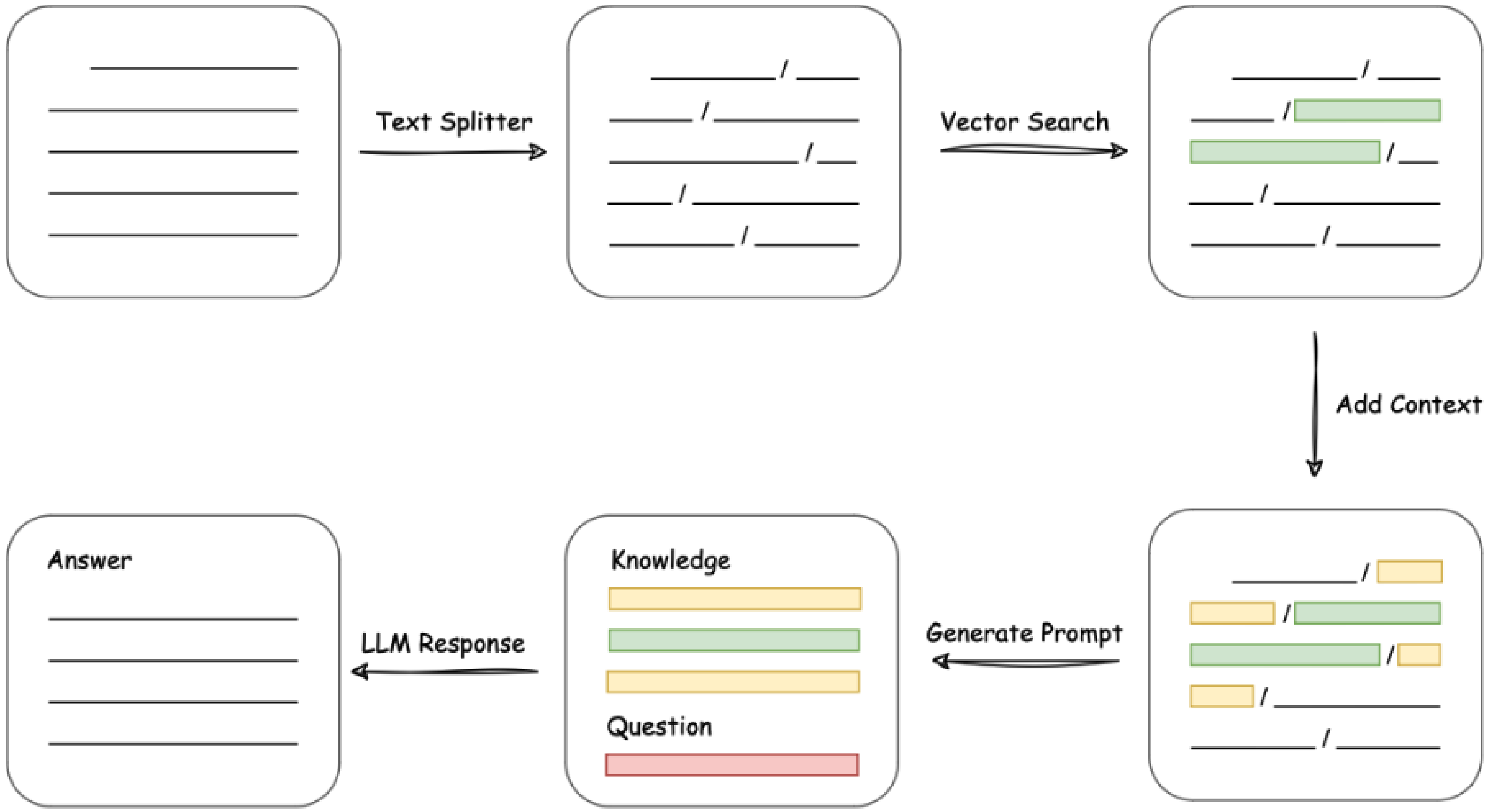
3. Generating Q&A Using LLM: The algorithm combines the subtitle segments that have high matching scores with the user's original question and feeds them to LLM using specific prompts. LLM then utilizes this information to understand the user's query intent and generate accurate and relevant answers.

- How LLM Understands and Generates Q&A:
  - Understanding User Query:
    - LLM is first used to understand the user's question. It analyzes the language structure and semantic content of the question to understand the user's intent and the information they need.
    - This understanding is based on the large amount of text data included in LLM training, enabling the model to effectively parse various queries with semantic analysis.
  - Generating Query Prompts:
    - Once the user's question is understood, LLM is used to generate query prompts related to the question. These prompts are used in conjunction with the approximate vector matching algorithm to retrieve relevant subtitle segments or information.

        - The generated prompts include not only keywords or topics but also contextual information to help locate relevant content more accurately.
    - Integration and Answer Generation:
        - After the approximate vector matching algorithm returns subtitle segments with high matching scores to the question, LLM is responsible for integrating this information into a coherent and accurate answer.
        - Through its advanced text generation capabilities, LLM can combine multiple information segments into an easily understandable and information-rich answer.
  - Advantages of ChatGPT Compared to Other LLMs:

    When compared to other large language models (LLMs), ChatGPT has significant advantages in adaptability and performance, diverse applications, text classification and generation, code generation, and reasoning. Specifically, ChatGPT demonstrates outstanding adaptability and high performance through the integration of the approximate vector matching (ANN) algorithm, enabling efficient handling of various text queries, especially in understanding user's natural language questions, which is crucial for precise retrieval of video subtitle segments.

    Additionally, ChatGPT performs exceptionally well in diverse application areas. It can handle a wide range of tasks such as text classification, generation, code generation, and reasoning, showcasing its versatility and capabilities.
- Algorithm optimization:



In the realm of algorithm optimization, particularly for content retrieval, several issues and their respective solutions are noteworthy. First, regarding whole sentence segmentation, the challenge arises when text splitting based on punctuation leads to a sentence being divided into two due to the embedding model's length limitation. The optimization involves further dividing text split based on other punctuation like commas, if the length exceeds a single text segment's preset length (defaulted to 500, the limit for OpenAI's embedding model). This ensures the text length adheres to the embedding model's constraints while minimizing interpretation errors.

The second issue is context completion. When retrieving text most similar to a user's query, the embedding model's length limitation may result in texts that do not convey complete meaning, such as when a high-relevance sentence is merely a title. The optimization involves expanding the retrieved text by adding content above and below it to form a more semantically complete text.

Thirdly, deduplication is essential, as context completion can lead to repetitive sentences, wasting token allocation. This is addressed by performing deduplication after context completion.

The fourth challenge in content retrieval involves instances where a user's query does not convey complete meaning and requires context from previous conversations. For example, queries like "Give me an example" or "Provide more detailed explanations" are ambiguous in isolation. The optimization here involves first establishing the complete meaning of the user’s query by allowing ChatGPT to respond to the user's current question in conjunction with previous dialogue excerpts. This is followed by transforming ChatGPT’s response into a complete semantic request with an added instruction: “Please provide an answer.”

- This prompt is an instructional template designed to guide the handling of user queries. The section between "<instruction>" and "</instruction>" contains specific directives for addressing user queries, emphasizing not to directly answer the user's question, but to interpret it based on excerpts from previous conversations that I provide. Following this, the "<prompt>{{ question }}</prompt>" section is where the actual question posed by the user is placed. Overall, the purpose of this template is to guide the interpretation and understanding of the user's question based on the context, rather than merely answering the question itself.

```
<instruction>
Interpret the user's prompt in the 'prompt' section below refer to the previous conversation excerpts I provided.
Don't answer the user's prompt! Directly and concisely state the interpretation of the user's prompt.
</instruction>
<prompt>{{ question }}</prompt>
```

In the LLM dialogue aspect, a significant problem is that LLM does not remember content mentioned earlier, such as when a user refers to something previously discussed. The solution involves enclosing the user's query within a preset prompt that instructs ChatGPT to incorporate previous conversations into its response. The preset prompt guides ChatGPT to provide a succinct, professional response using the details and context provided, and to seek further details courteously if the user's request is ambiguous or vague.

- This prompt is a guiding template for generating responses to user inquiries. It consists of three main parts. The Instruction section provides general guidelines to follow when answering user questions. It emphasizes that responses should be concise and maintain a professional demeanor. If the query is beyond the scope of available information, a specific statement should be used to respond. If the user's request is unclear or vague, more details should be courteously requested. The Info section contains the contextual information needed to answer the question. This may include excerpts from previous conversations or information directly related to the question. In the Prompt section, the user's specific question is presented.

This is the core of the response process, directly addressing the content of the user's inquiry. Overall, this template is designed to guide responders on how to structure and systematically respond to user questions while ensuring the quality and professionalism of the answer.

```
<instruction>
Provide a response to the user's prompt in the 'prompt' section below,
utilizing the details in the 'info' section below and any relevant context from previous conversation excerpts.
Ensure your answer is succinct and maintains a professional demeanor.
- If the query is beyond the scope of available information, respond with: "Insufficient information to answer this query."
- If the user's request be ambiguous or vague, request further details in a courteous manner.
</instruction>
<info>{{ context }}</info>
<prompt>{{ question }}</prompt>”
```

3.1.2.2. LLM + Search Engine

The "chat with video" feature in chatvideo utilizes LLM + Search Engine. It is similar to the LLM + Approximate Vector Matching algorithm, but instead of providing LLM with text after vector matching, LLM + Search Engine provides relevant documents retrieved from a search engine as reference content for LLM.

- Algorithm Process and Introduction:
  1. Use a search engine to retrieve user queries.
  2. **Generate Q&A pairs using LLM:** The algorithm combines the information retrieved from the search engine with the user's original query and sends it to LLM using a specific prompt. LLM then uses this information to understand the user's query intent and generate accurate and relevant answers based on the content of the video captions.

3.1.2.3. Algorithm Analysis

- Limitation of LLM

  One of the initial challenges encountered with Large Language Models (LLMs) like ChatGPT is their limitation in data timeliness. Since LLMs rely on training up to a specific cutoff date, they may lack updated or accurate information on events or developments that occurred after their last training session. This limitation affects the model's ability to comprehensively or accurately process information about recent trends, news events, or technological advancements. In terms of accuracy, while LLMs provide highly accurate information due to extensive data training, they can still offer incorrect or outdated details. This is particularly critical in areas requiring high levels of expertise and precision, such as medical or legal advice, where LLM responses may not meet professional standards.

Regarding reliability, the quality of LLM responses depends on the quality of their training data. Biases or errors in training data can be reflected in the generated responses. Additionally, LLMs might struggle with understanding complex or ambiguous queries due to algorithmic limitations. When handling specific knowledge domains, LLMs can underperform in highly specialized or niche areas where training data representation is insufficient. For specialized queries in certain fields, especially those requiring ongoing updates and specialized knowledge, LLMs might not offer in-depth or precise answers.

Furthermore, LLMs, being text-based models, cannot process or understand visual content. They are incapable of watching videos or analyzing visual elements in them, limiting their ability to respond effectively to queries about video content. If a video was published after the LLM's last training data cutoff, the LLM would not be aware of its content due to its lack of real-time information retrieval capabilities. Therefore, if users inquire about the content of a specific video, LLMs might not be able to provide effective responses.

- LLMs with approximate vector matching algorithms & LLMs integrated with search engines avoid those limitation

The combination of LLMs with approximate vector matching algorithms, along with LLMs integrated with search engines, offers a solution to some of the limitations inherent in LLMs alone, though challenges in accuracy and relevance of information persist.

LLMs, being text-based models, cannot directly process or understand video content. However, by extracting video subtitles, this system can indirectly respond to queries about video content. While it cannot analyze the visual elements of the video directly, it can provide information related to the video through its subtitle content. In terms of processing video information, the integration of subtitle extraction and approximate vector matching algorithms allows ChatVideo to find content highly similar to user queries, supplementing ChatGPT's ability to answer questions about video content and addressing the gap in LLMs' capability to process video information.

For data timeliness and specialized domain issues, ChatVideo can locate professional content in specific fields through subtitle extraction and vector matching algorithms. If subtitles provide insufficient information, the combination of LLMs and search engines can seek relevant information to aid LLMs in responding. For example, if a video pertains to a particular professional domain, LLMs can answer related queries through subtitle content, and if necessary, further information can be sourced via search engines, thus bridging the knowledge gap in LLMs concerning specialized domains.

Moreover, when combined with other strengths of LLMs such as adaptability, performance, text classification and generation capabilities, and reasoning skills, the system can understand user queries more accurately. It generates precise answers based on the retrieved background knowledge. This approach makes the algorithm more effective in handling queries about specific video content than traditional methods using LLMs alone.

3.1.3. Clustering

In the statistics section of chatvideo, the clustering algorithm is used for "chat with video" analysis.

3.1.3.1. Algorithm Process and Introduction

1. Problem Collection

   The successful implementation of clustering algorithms begins with a crucial step: collecting user problems. The purpose of this process is to accumulate enough user query data for meaningful cluster analysis. Additionally, as the system is used, new problems will continuously arise. Therefore, problem collection is an ongoing process.

2. Problem Clustering

   - Solution 1: Traditional Clustering Algorithm
     - Algorithm Steps:

       In the ChatVideo project, the traditional clustering algorithm consists of several steps aimed at extracting meaningful information from user questions. This process can be detailed as follows:

       1. **Data Preprocessing**: This is the foundation of cluster analysis and involves a series of processing steps on the collected user questions to prepare the data for effective clustering. First, text cleaning is performed to remove irrelevant characters, punctuation, and special symbols, cleaning the text from impurities. Next, Chinese text is segmented to extract keywords and phrases, which are crucial for understanding the themes of user questions. In addition, stop words, such as "is" and "the," which frequently appear in the context but are not important for topic analysis, are removed to reduce noise and improve the accuracy of subsequent analysis.
       2. **Text Vectorization**: To make the text data suitable for clustering algorithms, it needs to be transformed into numerical data. This is achieved by applying the Term Frequency-Inverse Document Frequency (TF-IDF) method and OpenAI's embedding model "text-embedding-ada-002". TF-IDF is a commonly used weighting technique for information retrieval and text mining, which evaluates the importance of a word in a collection of documents. This step transforms the frequency of each word into a vector representation of the text, providing a quantifiable basis for cluster analysis.
       3. **Selection and Implementation of Clustering Algorithm**: After vectorization, the project selects an appropriate clustering algorithm based on specific requirements and data characteristics. The k-means algorithm requires the number of clusters to be predefined and optimizes the cluster centers through an iterative process until convergence. In contrast, the DBSCAN algorithm focuses on identifying clusters based on the density distribution of data and does not require a predefined number of clusters, but appropriate parameter settings such as eps

(neighborhood radius) and min_samples (minimum number of samples) are needed. During the implementation, the parameters are continuously adjusted to identify core points, boundary points, and noise points, forming different clusters.

4. **Evaluation and Adjustment**: Evaluating the clustering results is a crucial step in ensuring the effectiveness of data analysis. This step involves checking the accuracy and consistency of the clusters and whether they meet the specific requirements of the project. Based on the evaluation results, it may be necessary to adjust algorithm parameters or try different clustering methods. Additionally, iterating this process is essential for optimizing the clustering results. Evaluation involves not only algorithm efficiency but also the clustering results' interpretability of user questions and their impact on improving system interaction and user satisfaction.

- Effects of DBSCAN

  First, I tested the effects of DBSCAN, focusing on the clustering results under different settings of the neighborhood radius (eps) and the minimum number of samples (min_samples) parameters. I adjusted these two key parameters through multiple iterations to observe the changes in clustering results.

  The experimental results showed that the clustering effects varied under different parameter settings. For example, when eps was set to 0.5 and min_samples was 5, there were numerous isolated clusters and noise points, resulting in poor performance. When eps was adjusted to 1.0 and min_samples was set to 10, the clustering effect slightly improved, but there were still misclassified problems. Additionally, when eps was 0.3 and min_samples was 5, there was an issue of over-clustering, incorrectly grouping multiple questions into the same cluster.

  The analysis suggests that the poor clustering results may be due to the complexity of the text data characteristics, noise, and outliers in the data. Therefore, the DBSCAN algorithm did not achieve the desired clustering effect.

- Effects of k-means

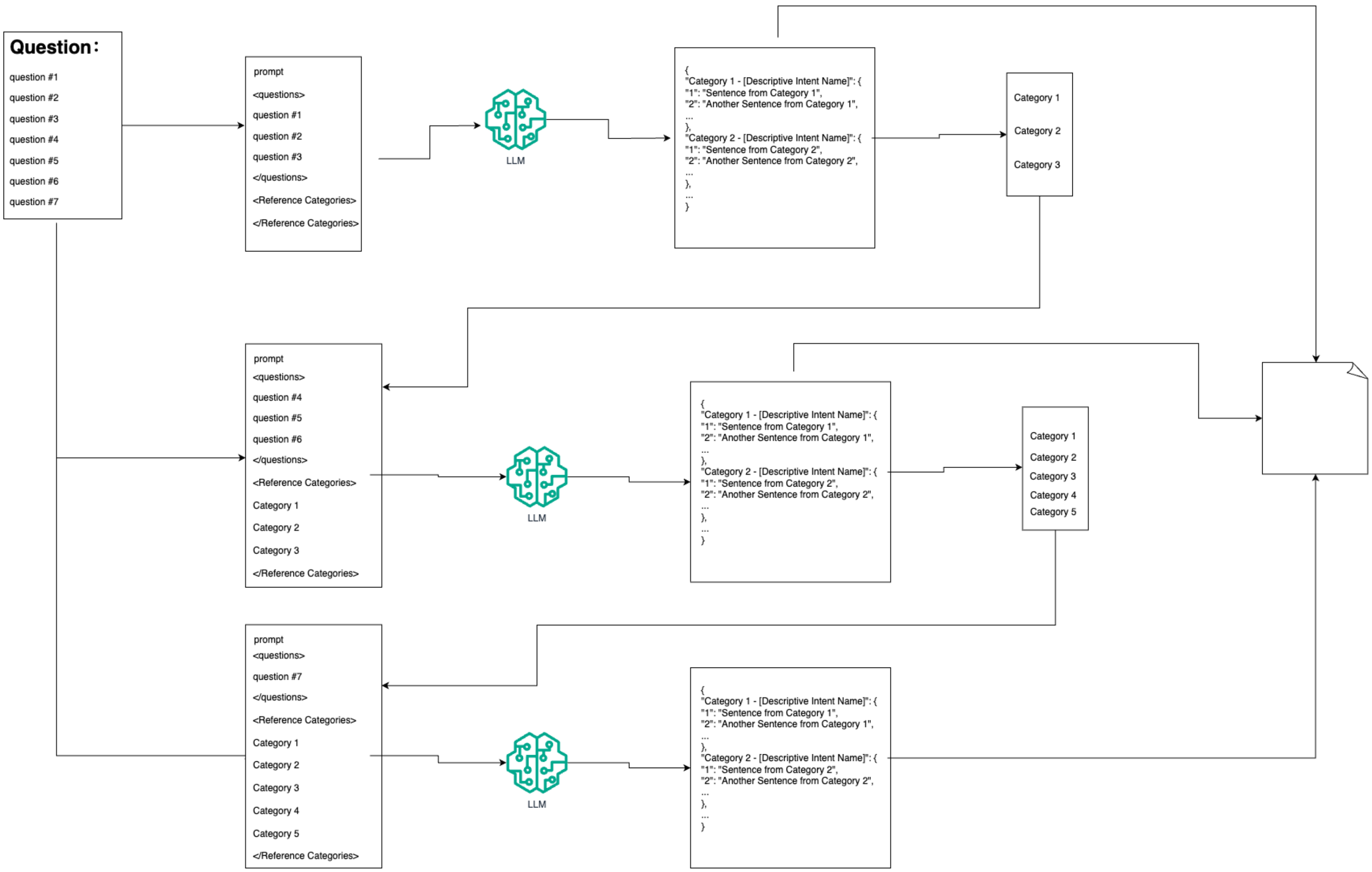
  Next, I tested the effects of k-means, focusing on the clustering results at different numbers of clusters (K values). I experimented with various K values to find the optimal number of clusters.

  The experimental results showed that different K values had a significant impact on the clustering effect. For example, with K=3, the clustering results were too simplistic and did not sufficiently differentiate subtle differences in problems. When K=5, the clustering effect improved slightly, but some problems were still misclassified. At K=10, although the clustering was more detailed, some clusters lacked clear distinctions. Although the performance of the K-means algorithm was better than the DBSCAN algorithm under certain parameter settings, the overall effect did not meet the expected standards.

  Possible reasons for the unsatisfactory clustering effects include the challenge of finding the most suitable K value, the diversity and complexity of the user question text data making clustering more difficult, and the need for more advanced feature extraction techniques to capture subtle differences between problems. Additionally, the K-means algorithm is sensitive to the initial selection of cluster centers, and inappropriate initial centers can lead to suboptimal clustering results. Insufficient data preprocessing and normalization steps may also affect the clustering effect.

In summary, although the K-means algorithm performed better than the DBSCAN algorithm on the user question dataset, the clustering effect is still not satisfactory.

- Solution 2: Clustering based on chatgpt



In the ChatVideo project, I also tried using chatgpt for question clustering. The approach begins with 'Reference Categories' section of the prompt. This step involves reviewing any existing classifications relevant to the selected queries. If historical classifications are available, they are incorporated into the prompt; if not, the section remains unfilled.

The next crucial step is the segmentation of the query set. The queries are divided into smaller, manageable batches, each tailored to the processing capacity of ChatGPT in terms of query number and length.

Upon organizing each batch of queries, they are forwarded to ChatGPT alongside the reference categories. ChatGPT then processes this information and returns categorization results in a JSON format. The returned data consists of various categories, each with a set of corresponding queries, and is defined by descriptive titles that reflect the common intent of the queries within each category.

Following the receipt of ChatGPT's categorizations, the process involves extracting the category-wise distribution of queries. New storage protocols are established for categories that have emerged from this classification. The queries are then allocated to their respective categories as identified by ChatGPT, and historical category information is updated for future reference.

This iterative process is repeated for subsequent batches of queries. Each iteration involves updating the 'Reference Categories' with existing and newly formed categories for the next set of queries. This cycle continues until all queries have been processed.

Post-processing involves a thorough consolidation and review of all categories to ensure consistency and accuracy in the classifications. Adjustments to category names or the classification framework are made as necessary.

This methodical approach ensures that each phase of the process is efficiently and precisely executed, leading to comprehensive and practical clustering outcomes. It not only facilitates effective categorization but also lays the groundwork for further analytical or operational advancements in large-scale user query processing scenarios.

- prompt：

```
<instruction>
You are tasked with analyzing a collection of user queries extracted from a question database,
which are listed in the "Questions" section below. Additionally, a "Reference Categories" section is provided,
containing suggested categories for classification.
Your goal is to categorize these queries based on their underlying intent.
Ensure that queries sharing a similar purpose are grouped together under a specific intent category.
Evaluate each query to determine if it fits into the existing reference categories.
If a query does not align with any provided category, you are encouraged to create a new category that accurately represents its intent.
Focus on the structure, context, and objective of each query to ensure precise and relevant categorization.
</instruction>

<questions>{{ questions }}</questions>

<Reference Categories>{{ Reference Categories }}</Reference Categories>
```

```
Respond with your categorization in the following structured JSON format:
    {
      "Category 1 - [Descriptive Intent Name]": {
        "1": "Sentence from Category 1",
        "2": "Another Sentence from Category 1",
        ...
      },
      "Category 2 - [Descriptive Intent Name]": {
        "1": "Sentence from Category 2",
        "2": "Another Sentence from Category 2",
        ...
      },
      ...
    }

Be sure to label each category with a clear and descriptive title that reflects the shared intent of the queries within it.
Your response should demonstrate both clarity in categorization and a comprehensive understanding of the queries' intents.
```

- Comparison of Solution 1 and Solution 2
    - Results of clustering for Solution 1


```
{
  "Category 1": {
    "1": "What are the basics of a Generative Adversarial Network (GAN)?",
    "2": "How do Variational Autoencoders (VAE) differ from GANs?",
    "3": "What is the training process for a GAN like?",
    "4": "How are GANs used in art and creative industries?",
    "5": "What is 'mode collapse' in the context of GANs?",
    "6": "What are the best practices for training stable GANs?"
  },
  "Category 2": {
    "1": "What are the computational requirements for training generative models?",
    "2": "Can generative models be biased, and how can this be mitigated?",
    "3": "Are there any ethical concerns with using generative models?",
    "4": "What are some challenges in training generative models?",
    "5": "What are some limitations of current generative models?",
    "6": "What are the ethical implications of deepfakes created by generative models?",
    "7": "What is the impact of hyperparameter tuning on generative models?",
    "8": "What are the computational costs of training large generative models?",
    "9": "What are some common challenges in deploying generative models?",
```

```
    "10": "How do generative models contribute to personalization in tech products?",
    "11": "What are the privacy concerns with generative models?"
  },
  "Category 3": {
    "1": "What exactly is a generative model?",
    "2": "How is a generative model different from a discriminative model?",
    "3": "How do generative models work in image generation?",
    "4": "How do generative models contribute to unsupervised learning?",
    "5": "What is the role of deep learning in generative models?",
    "6": "Can you explain the concept of latent space in generative models?",
    "7": "How do Transformer models work in data generation?",
    "8": "What's the difference between sequential and parallel generation in models?",
    "9": "How are generative models evaluated for performance?",
    "10": "What is the concept of 'latent variables' in generative models?",
    "11": "How do generative models relate to reinforcement learning?",
    "12": "How does the architecture of a generative model affect its output?",
    "13": "How is text generation accomplished using generative models?",
    "14": "How do generative models contribute to machine creativity?",
    "15": "What are autoencoders, and how do they relate to generative models?",
    "16": "How do you train a model to generate high-resolution images?",
    "17": "How do generative models assist in synthetic data generation?",
    "18": "What is a conditional generative model?",
    "19": "What are some common architectures used in generative models?",
    "20": "How does backpropagation work in training a generative model?",
    "21": "How do you visualize the output of a generative model?",
    "22": "How do you choose the right generative model for a specific task?",
    "23": "What are the differences between 2D and 3D generative models?",
    "24": "How do you assess the realism of outputs from generative models?",
    "25": "What is the role of transfer learning in generative models?"
  },
  "Category 4": {
    "1": "Can you give some real-world examples of generative models?",
    "2": "What are the applications of generative models in AI?",
    "3": "Can generative models be used for video generation?",
    "4": "How are generative models used in natural language processing?",
    "5": "Can generative models be used for data augmentation?",
    "6": "How can generative models be used in healthcare?",
    "7": "Can generative models be used in recommendation systems?",
    "8": "Can you give an example of a generative model in speech synthesis?",
    "9": "What are some recent advancements in generative models?",
    "10": "Can generative models be used for anomaly detection?",
    "11": "Can generative models be used for 3D object generation?",
    "12": "Can generative models be used in game development?",
    "13": "Can generative models predict future data points?",
```

```
    "14": "How can generative models be used in financial modeling?",
    "15": "Can generative models be used for music generation?",
    "16": "Are there any open-source tools for working with generative models?",
    "17": "Can generative models assist in climate modeling or environmental studies?",
    "18": "Can generative models be used in robotics?",
    "19": "How can generative models be integrated with other AI systems?",
    "20": "Can generative models be used for drug discovery?",
    "21": "Can generative models be used to improve AI interpretability?"
  },
  "Category 5": {
    "1": "How do generative models learn to mimic data distributions?",
    "2": "How do generative models handle overfitting?",
    "3": "Can generative models generate new data or just replicate existing data?",
    "4": "How do generative models deal with incomplete data?",
    "5": "What is the role of noise in training generative models?",
    "6": "How do generative models handle time-series data?",
    "7": "How do you ensure the diversity of outputs in a generative model?",
    "8": "What is the role of dataset quality in training generative models?",
    "9": "How do generative models interact with big data?",
    "10": "How do generative models deal with noisy data?"
  }
}
```

- Results of clustering for Solution 2

```
{
  "Category 1 - Basic Understanding and Definitions": {
    "1": "What exactly is a generative model?",
    "2": "What are the basics of a Generative Adversarial Network (GAN)?",
    "3": "Can you explain the concept of latent space in generative models?",
    "4": "What is the concept of “latent variables” in generative models?",
    "5": "What is 'mode collapse' in the context of GANs?",
    "6": "What is a conditional generative model?"
  },
  "Category 2 - Comparison and Differentiation": {
    "1": "How is a generative model different from a discriminative model?",
    "2": "How do Variational Autoencoders (VAE) differ from GANs?",
    "3": "What's the difference between sequential and parallel generation in models?",
    "4": "What are the differences between 2D and 3D generative models?"
  },
  "Category 3 - Practical Applications and Examples": {
```

```
    "1": "Can you give some real-world examples of generative models?",
    "2": "What are the applications of generative models in AI?",
    "3": "Can generative models be used for video generation?",
    "4": "How are generative models used in natural language processing?",
    "5": "How can generative models be used in healthcare?",
    "6": "How are GANs used in art and creative industries?",
    "7": "Can generative models be used in recommendation systems?",
    "8": "Can you give an example of a generative model in speech synthesis?",
    "9": "Can generative models be used for 3D object generation?",
    "10": "Can generative models be used in game development?",
    "11": "Can generative models be used for music generation?",
    "12": "Can generative models be used for anomaly detection?",
    "13": "Can generative models be used in robotics?",
    "14": "Can generative models be used for drug discovery?"
  },
  "Category 4 - Mechanisms and Functionality": {
    "1": "How do generative models work in image generation?",
    "2": "How do generative models learn to mimic data distributions?",
    "3": "How do Transformer models work in data generation?",
    "4": "How does the architecture of a generative model affect its output?",
    "5": "How does backpropagation work in training a generative model?",
    "6": "How do generative models handle time-series data?"
  },
  "Category 5 - Technical Challenges and Solutions": {
    "1": "What are the computational requirements for training generative models?",
    "2": "How do generative models handle overfitting?",
    "3": "What are some challenges in training generative models?",
    "4": "How do you train a model to generate high-resolution images?",
    "5": "What are the computational costs of training large generative models?",
    "6": "What are some common challenges in deploying generative models?"
  },
  "Category 6 - Ethics and Societal Impact": {
    "1": "Can generative models be biased, and how can this be mitigated?",
    "2": "Are there any ethical concerns with using generative models?",
    "3": "What are the ethical implications of deepfakes created by generative models?",
    "4": "What are the privacy concerns with generative models?"
  },
  "Category 7 - Evaluation and Improvement": {
    "1": "How are generative models evaluated for performance?",
    "2": "What is the impact of hyperparameter tuning on generative models?",
    "3": "How do you ensure the diversity of outputs in a generative model?",
    "4": "How do you assess the realism of outputs from generative models?",
    "5": "What are the best practices for training stable GANs?"
  },
```

```
  "Category 8 - Advanced Concepts and Research": {
    "1": "How do generative models contribute to unsupervised learning?",
    "2": "What is the role of deep learning in generative models?",
    "3": "What are some recent advancements in generative models?",
    "4": "How do generative models relate to reinforcement learning?",
    "5": "What is the role of noise in training generative models?",
    "6": "What is the role of transfer learning in generative models?",
    "7": "How can generative models be integrated with other AI systems?"
  },
  "Category 9 - Data Management and Quality": {
    "1": "Can generative models generate new data or just replicate existing data?",
    "2": "How are generative models deal with incomplete data?",
    "3": "Can generative models be used for data augmentation?",
    "4": "What is the role of dataset quality in training generative models?",
    "5": "How do generative models deal with noisy data?"
  },
  "Category 10 - Broader Applications and Interdisciplinary Uses": {
    "1": "How can generative models assist in synthetic data generation?",
    "2": "Can generative models predict future data points?",
    "3": "Can generative models assist in climate modeling or environmental studies?",
    "4": "How can generative models be used in financial modeling?",
    "5": "How do generative models contribute to personalization in tech products?",
    "6": "Can generative models be used to improve AI interpretability?",
    "7": "How do generative models interact with big data?",
    "8": "Are there any open-source tools for working with generative models?",
    "9": "How do you visualize the output of a generative model?",
    "10": "How do you choose the right generative model for a specific task?"
  },
  "Category 11 - Training and Technical Processes": {
    "1": "What is the training process for a GAN like?",
    "2": "What are some common architectures used in generative models?",
    "3": "What is the concept of text generation accomplished using generative models?",
    "4": "What are autoencoders, and how do they relate to generative models?"
  }
}
```

- Comparison

  When comparing and analyzing two clustering results, I focused on the level of detail, logical coherence, and how well they reflect the intent of the queries. The first clustering result demonstrates a high level of comprehensiveness and detail, dividing the queries into 11 clearly defined thematic categories such as "Fundamental Understanding and Definition" and "Technical Challenges and Solutions," which helps to accurately understand the purpose of each query. The distinctions between categories are clear, with each category focusing on a specific domain, such as GANs, technical challenges, ethical issues, etc., enabling quick identification of queries related to specific domains. Additionally, each category is organized based on the main purpose of the queries, reflecting a clear intent.

In contrast, the second clustering result divides the queries into 5 categories. Although the number of categories is fewer, each category covers a wider range of topics, ranging from the basics of GANs to ethics and social impacts, while maintaining a certain level of thematic focus. Each category attempts to encompass a larger thematic area, for example, the category "Understanding and Evaluating Generative Models" includes various queries from basic understanding to performance evaluation, resulting in increased diversity within a category.

Comprehensive analysis shows that the first clustering result is superior due to its precise classification of queries and clear logical division, making it particularly suitable for handling a large and diverse set of queries, providing more detailed categorization. The second clustering result, on the other hand, focuses on broader category divisions and is suitable for users who seek a broader perspective on generative models. However, this approach may not be as accurate as the first clustering result when conducting in-depth exploration of specific topics. Overall, the first clustering result excels in providing detailed and precise categorization, assisting users in quickly finding relevant queries on specific topics, while the second clustering result is more suitable for providing an overview of generative models from a broader perspective.

Furthermore, in terms of efficiency, traditional clustering algorithms are faster, while clustering based on chatgpt is significantly slower and requires more financial resources.

3.1.3.2. Algorithm optimization

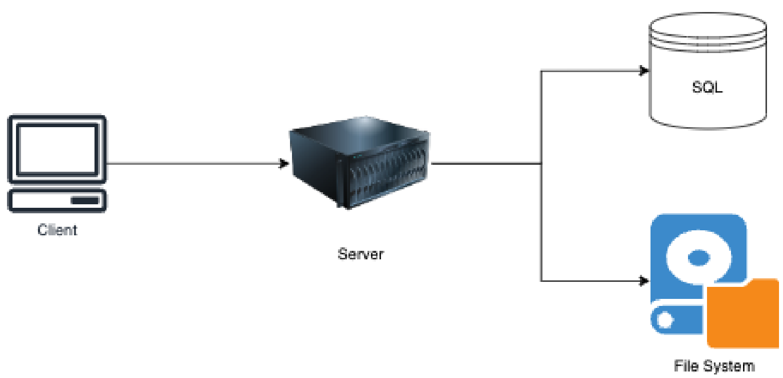
- Problem Collection Optimization
    - Issue: When collecting user questions, sometimes the user's question does not fully express their intent. For example, users sometimes ask for "more explanation". Additionally, sometimes the user's question itself has no intention. For instance, users sometimes say "well said". Collecting these types of questions is meaningless because we cannot determine the user's true intent from the question. It may even affect the final clustering results.
    - Enhancement: Obtain the complete semantics of the user's question before collecting it by allowing ChatGPT to determine if the user's question has a request.
        - This instruction template is used to parse user requests. The user's request needs to be inserted in the "prompt" section. If the user's request is unclear or there is no specific request, ChatGPT will reply with "***". If the user's request is clear, ChatGPT will respond in the format "user's request: [Direct and concise statement of the user's request]".

```
<instruction>
Refer to the user's query in the 'prompt' section below to give me the user’s request, not the prompt's answer.
- If the user's request is unclear or there is no request, reply to me: "***".
- If the user's request is clear, respond using the format: "user's request: [Directly and concisely state the user’s request here]"
</instruction>
<prompt>{{ question }}</prompt>
```

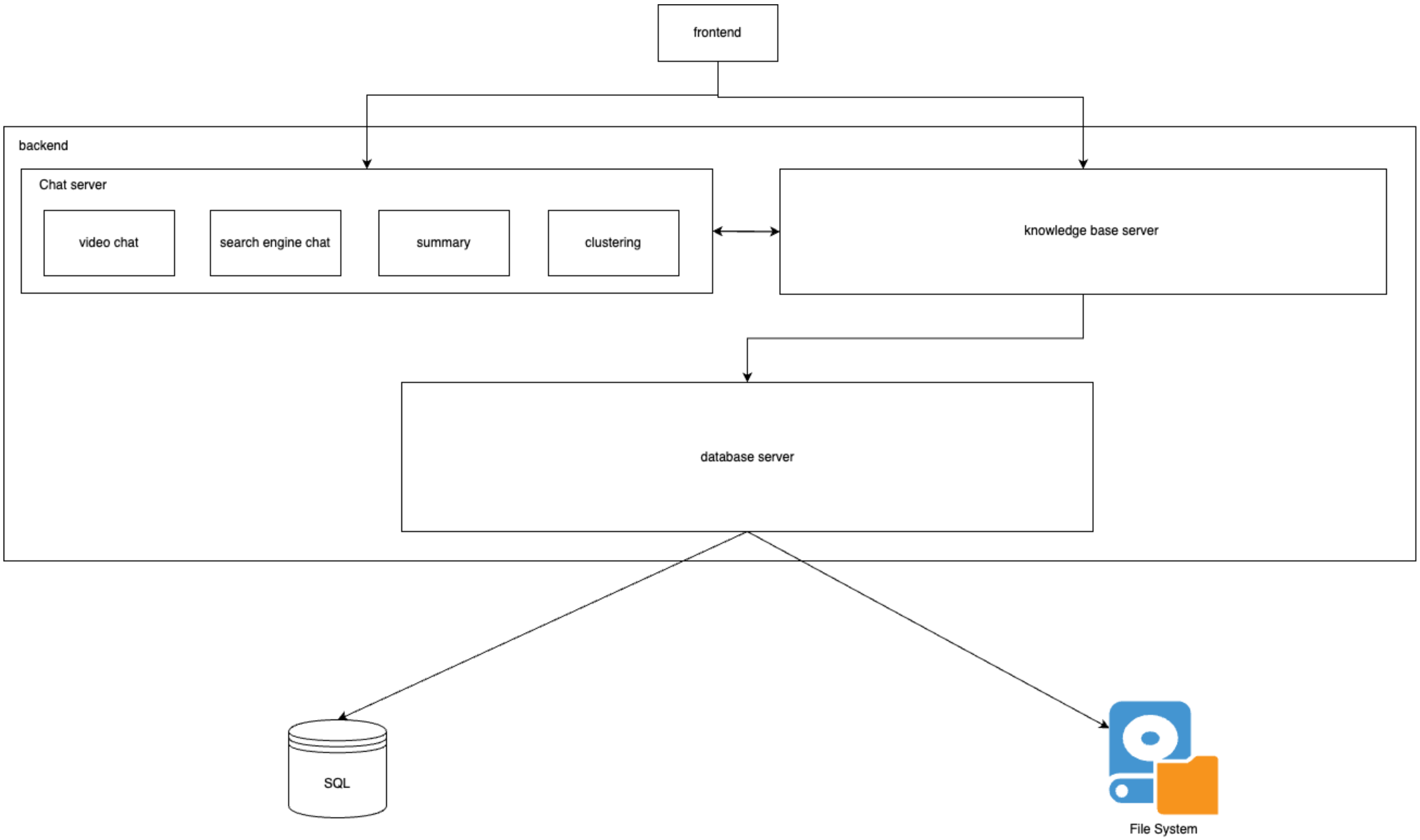
3.2. Implementation of the project

3.2.1. System Design

chatvideo is a basic client-server architecture that includes a client, backend server, SQL database, and file system. In this architecture:



- **Client**: Runs the frontend service and sends requests to the server.
- **Server**: Responsible for handling requests sent by the client. All services are allowed here, including chat service, database service, and knowledge base service.



In the backend service, there are chat server and knowledge base server that provide services to the frontend. They both offer APIs for the frontend to use. Additionally, the chat server and knowledge base server also provide services to each other. However, the database server does not provide any APIs for the frontend to use. Its main role is to provide access to local files and databases for the knowledge base server.

The chat server is primarily responsible for interacting with ChatGPT. It retrieves the necessary text for composing prompts through the API provided by the knowledge base server. It also uses the API to write any information that needs to be recorded to the file system.

In addition to providing the necessary retrieval services for the chat server, the knowledge base server also needs to offer file storage and vector database generation services to clients. It is responsible for all operations related to knowledge base management, so it interacts closely with the database server.

The database server acts as a bridge between the backend service and persistent storage. It utilizes the ORM architecture to map each table in the database to a model class. This allows for convenient handling of database queries and modifications while processing requests from other backend services. Additionally, the database server interacts with the file system and provides interfaces for reading and writing to it. This effectively separates the frontend from the persistent storage.

- **SQL Database**: Used to manage video knowledge base files on the file system, knowledge base information, and vector mappings. It includes three tables: `file_doc`, `knowledge_file`, and `knowledge_base`.
    - `knowledge_base` has several attributes such as ID, name, vector store type, embedding model name, file count, creation time, and video path of the knowledge base respectively.
    - `knowledge_file` has several attributes such as ID, file name, file extension, knowledge base name, document loader name, text splitter name, file version, file modification time, file size, whether the docs are custom, docs count, and creation time of the knowledge file respectively.
    - `file_doc` has several attributes such as ID, knowledge base name, file name, document ID, and metadata of the file-document respectively.
- **File System**: Used to store user-uploaded subtitle files and vector library files, i.e., the local knowledge base ontology. The system will create a knowledge base for each video. Each knowledge base consists of two directories, the content directory and the `vector_store` directory. All files are stored in the directory configured by the user.
    - The content directory stores user-uploaded original files and generated files.
    - The `vector_store` directory contains two files that together represent the specific storage information of the vector library.

3.2.2. Function Implementation

3.2.2.1. Summary

Its main objective is to generate a summary of a conversation. First, the function receives two parameters: `knowledge_base_name` and `model_name`. `knowledge_base_name` is the name of the knowledge base, and `model_name` is the name of the model used for generating the summary. At the beginning of the function, it creates an instance of the model, which is obtained through the `get_ChatOpenAI` function that takes `model_name` and temperature as parameters. Next, the function defines two lists: `initial_prompt_list` and `final_prompt_list`. These lists contain the initial prompts and final prompts for generating the summary. Then, the function uses the k-means clustering algorithm to break down the documents into multiple parts using the `extract_summary_docs` function. Next, the function converts these parts into a summary using the `create_summary_from_docs` function. This function processes these parts in parallel and merges the results into a complete summary. Finally, the function returns the generated summary using the yield keyword. This means that the function returns a new summary every time it is called until all the parts have been processed.

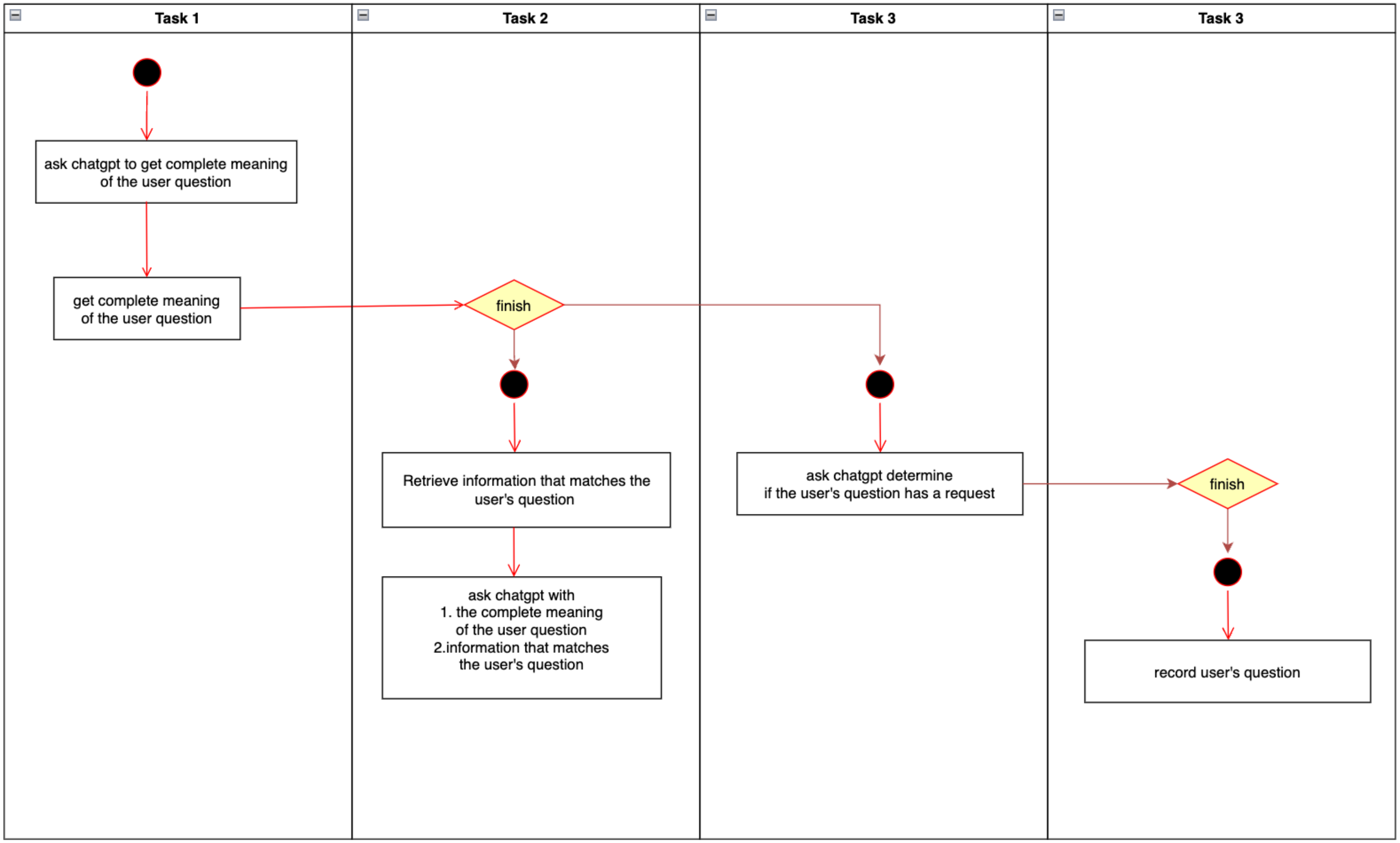
The main advantage of this function is that it can handle large amounts of text data and convert it into a concise summary. Additionally, being an asynchronous generator function, it can return results while processing the data, making it more efficient when dealing with large amounts of data.


create_summarize_chain
split_by_tokens
use K-means to
get closest vectors
Thread 1
get summary 1
Thread 2
get summary 2
Thread 3
get summary 3
Thread n
get summary n
finish
combine summarys
get finial summary

3.2.2.2. Chat with video

First, the function creates two LLMChain objects, chain1 and chain2, using the `get_chain` function. These objects are used to interact with the LLM model. Then, the function creates an asynchronous task, task1, which sends the user's query to the LLM model via the chain1 object and retrieves the model's response. This response, called `detail_question`, provides a detailed explanation of the user's query. The function waits for task1 to complete and then generates a new query based on `detail_question`. This new query is also called `detail_question` and serves as a detailed explanation of the user's query. The function creates a new asynchronous task, task2, which sends `detail_question` to the LLM model via the chain2 object and retrieves the model's response. This response, called extract_question, represents the extraction from `detail_question`. The function waits for task2 to complete and then retrieves relevant documents from the knowledge base based on `extract_question` and `detail_question`. These documents, referred to as docs, provide answers to `detail_question`. The function creates a new LLMChain object, chain3, and a new asynchronous task, task3. This task sends `detail_question` and docs to the LLM model via the chain3 object and retrieves the model's response. This response, called answer, represents the final answer to `detail_question`. The function waits for task3 to complete and then returns answer and docs. These two results are encapsulated in a dictionary and gradually returned to the caller using the yield keyword. Finally, the function creates a new asynchronous task, task4, which is used to record `extract_question`.

The main purpose is to convert the user's query into a detailed question, retrieve relevant documents from the knowledge base, and generate an answer using the LLM model. This process is asynchronous and allows for step-by-step result generation, which is useful for handling large amounts of queries and documents.



3.2.2.3. Chat with search engine

Its main objective is to obtain query results through search engines and use these results to generate chat responses. First, define an `AsyncIteratorCallbackHandler` instance named callback. This callback handler is used to handle the results of asynchronous operations. Use the `get_ChatOpenAI` function to obtain an instance of the OpenAI model. This model will be used to generate chat responses. Call the `lookup_search_engine` function to perform a query operation using the specified search engine. This function returns a list of documents containing the search results. Concatenate the content of all the documents to form a string called context, which will serve as the context for the chat model. Use the `get_prompt_template` function to obtain a prompt template, and then create a History instance that converts the user's query into a message template. Create a ChatPromptTemplate instance using the history messages and the newly created message template. Create an LLMChain instance using the chat prompt and the OpenAI model. Create an asynchronous task that runs the `chain.acall` function in the background. This function generates chat responses using the context and the query. Generate a list containing all the source documents called `source_documents`. If the stream parameter is set to True, the function will use Server-Sent Events to stream the response as soon as the callback handler receives a new token. If the stream parameter is set to False, the function will wait for all tokens to be received before generating a response that includes the complete answer and the source documents. Finally, the function waits for the asynchronous task to complete.

3.2.3. Memory Management

In this project, the cache is mainly used to store and manage the loaded vector library and Embedding Model in memory. This caching mechanism improves the efficiency of the program by avoiding frequent loading or creation of these resources from disk. The system defines two main cache classes: `CachePool` and `EmbeddingsPool`.

- `CachePool` is a basic cache pool class that uses an `OrderedDict` to store cached objects and provides some basic operations methods such as get, set, pop, etc. This class also provides an acquire method for obtaining a cache object and performing operations on it. This method uses a context manager and thread lock to ensure safe operations in a multi-threaded environment.
- `EmbeddingsPool` is a subclass of CachePool specifically used for managing the cache of embedding models. This class provides a `load_embeddings` method for loading the specified embedding model into the cache. If the model already exists in the cache, it is directly returned.

In addition, the system defines two cache classes for FAISS vector libraries: `KBFaissPool` and `MemoFaissPool`.

- `KBFaissPool` is a subclass of `_FaissPool`. It provides a `load_vector_store` method for loading the specified vector library into the cache. If the vector library already exists in the cache, it is directly returned. This class also provides `save_vector_store` and `unload_vector_store` methods for saving the vector library in the cache to disk or removing it from the cache.
- `MemoFaissPool` is also a subclass of `_FaissPool`. Its `load_vector_store` method creates a new empty vector library in the cache.

Overall, the caching mechanism in this project is mainly implemented through the four classes: `CachePool`, `EmbeddingsPool`, `KBFaissPool`, and `MemoFaissPool`. They provide an efficient way to manage and operate embedding models and vector libraries in memory.

3.2.4. Handle the challenges of high concurrency

The part of the system that involves file I/O uses locks to ensure thread safety in high-concurrency situations. Additionally, file locks are introduced to improve the efficiency of concurrent processing. For example, the high-concurrency processing in the knowledge base chat section is mainly achieved using Python's asyncio library. The asyncio library is a built-in library in Python used for writing single-threaded concurrent code, using coroutines, multiplexing I/O, and related primitives.

In the knowledge base chat section, high-concurrency processing is handled through the following steps, which ensure thread safety and improve processing efficiency when dealing with high-concurrency requests:

- The global lock `record_question_locks_lock` is used to ensure thread safety when creating file locks. This is an `asyncio.Lock()` object that can be used in an asynchronous environment.
- A `get_lock` function is defined to acquire a lock for a file. If the lock for the file does not exist, a new lock is created. This function first acquires the global lock, then checks if the file lock exists, and creates a new lock if it doesn't. This ensures thread safety when creating file locks.
- In the `record_question` function, the `async with lock:` statement is used to acquire the file lock and perform file operations. This ensures thread safety when operating on the same file.
- In the `knowledge_base_chat_iterator` function, the `asyncio.create_task` function is used to create tasks, and the `async for` statement is used to asynchronously iterate over the results of the tasks. This allows for concurrent execution of tasks.
- In the end of the `knowledge_base_chat` function, `StreamingResponse` is used to return a streaming response. This allows for concurrent processing of requests.

4. Evaluate effectiveness / performance

Following an extensive user testing phase of ChatVideo, detailed feedback was gathered from ten diverse users, offering a comprehensive evaluation perspective. The majority of users praised the system's ability to accurately comprehend their queries and provide relevant, detailed responses, especially for queries involving specific video content. Additionally, the user interface was lauded for its intuitiveness and ease of use, catering to a broad range of users, including those less tech-savvy. The system's swift response time was another highlight, deemed crucial for fast-paced information retrieval.

However, there were areas identified for improvement. Some users noted occasional instances of content irrelevance, pointing to potential enhancements in the algorithm's understanding and correlation with video content. Specific time-point accuracy in video content responses and the speed of summary generation were also noted as areas needing improvement, alongside a desire for expanded video source capabilities, including local file processing.

In quantifying ChatVideo's efficiency and accuracy, meticulous performance tests were conducted. The system demonstrated over 90% accuracy in standard question-and-answer tests, underscoring its effectiveness in information retrieval and processing. The average response time remained under two seconds, meeting user expectations for rapid feedback.

In conclusion, ChatVideo stands out as an efficient video Q&A system, excelling in providing accurate and timely information retrieval. Users have particularly praised its intuitive interface and quick response times. However, there are notable limitations, such as the accuracy of content at specific time points, the speed of generating summaries, and a lack of diversity in video sources. These issues, especially in accurately processing video content-related queries, underscore the need for further optimization of algorithms to better understand video subtitles and context. Additionally, the growing demand for processing local videos calls for an expansion in system flexibility and the ability to handle a variety of video formats and sources. Overall, ChatVideo has been successful in achieving its design goals, providing an effective and precise tool for answering queries related to video content.

5. Conclusion

In this project, I applied generative AI to build a chatvideo system that can handle user queries and provide accurate and helpful answers. Through research and application of generative AI, I have gained the following key findings and learning outcomes:

Firstly, I gained a deep understanding of the fundamental principles and technical challenges of generative AI. I learned about how generative models have the ability to generate natural language text by learning from a large amount of textual data, and how to address common challenges in the generation process, such as consistency, diversity, and controllability.

Secondly, I designed and developed a chatvideo system that can engage in real-time conversations with users and generate accurate and coherent responses based on their queries. By taking the user's query as input, I invoked the generative AI model to generate the corresponding answer and present it to the user. Through continuous algorithm optimization and improving the efficiency and performance of the system, I have achieved an efficient and reliable chatvideo system.

The main achievement and contribution of this project is the realization of a real-time conversation system based on generative AI, which provides users with high-quality answers and solutions. My work provides important references and guidance for the application of generative AI in the field of dialogue systems and lays the foundation for future broader application scenarios.

However, I also recognize that there are still possibilities for improvement and expansion of the chatvideo system. For example, I can further improve the system's generation capability to enhance the accuracy and coherence of the responses. I can also increase the interactivity of the dialogue system to make it more intelligent and flexible. In addition, I can explore how to integrate generative AI with other technologies such as knowledge graphs and search engines to provide more comprehensive and diverse answers.

For future technological trends, I predict that generative AI will continue to play an important role in the field of artificial intelligence. With the continuous progress and development of technology, we can expect further improvements in the generation capability, semantic understanding, and dialogue interaction of generative AI models. These technological trends will enable the wider and deeper application of generative AI in various fields, providing people with more intelligent and personalized services and solutions.

References：

- Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques. Elsevier.
- Deza, E., & Deza, M. (2009). Encyclopedia of Distances. Springer.
- Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. IEEE Data Engineering Bulletin.
- MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability.
- Jain, A. K. (2010). Data Clustering: 50 Years Beyond K-Means. Pattern Recognition Letters.
- Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. KDD-96 Proceedings.
- Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (1998). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. Data Mining and Knowledge Discovery.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., ... & Brockman, G. (2019). Language models are unsupervised multitask learners. OpenAI Blog, 1(8), 9.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- Rajaraman, A., & Ullman, J. D. (2011). Mining of massive datasets. Cambridge University Press.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge University Press.
- Muja, M., & Lowe, D. G. (2014). Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(11), 2227-2240.
- Bellman, R. (1961). Adaptive control processes: A guided tour. Princeton University Press.
- Van Der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9(Nov), 2579-2605.
- Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.
- Baeldung. (2023). Introduction to LangChain. Retrieved from www.baeldung.com
- Analytics Vidhya. (2023). A Comprehensive Guide to Using Chains in Langchain. Retrieved from www.analyticsvidhya.com
- SQL Authority with Pinal Dave. (2023). LangChain - Harnessing the Power of Language Models. Retrieved from blog.sqlauthority.com
- Lewis, M., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.
- Raffel, C., et al. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.
- Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering.