Major Information Consulting System: A RAG-Based Knowledge Assistant Integrating FastGPT and Gemini AI

Qinghong Cai
University of Illinois Urbana-Champaign
Urbana, United States
qc32@illinois.edu

ABSTRACT

This report presents a software system that applies Retrieval-Augmented Generation (RAG) techniques to deliver accurate and efficient responses to professional academic questions. The system employs modern information retrieval methods by crawling content from online forums (e.g., Reddit), semantically filtering it using large language models (Google Gemini), and embedding the results into a vectorized knowledge base (FastGPT). By integrating web scraping, semantic search, and neural language generation, the system transforms unstructured community knowledge into structured, query-relevant responses. We detail the system architecture, implementation strategies, and provide usage instructions. Evaluation metrics and future development plans are also discussed.

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

Retrieval-Augmented Generation, Information Retrieval, Semantic Embedding, FastGPT, Academic Advising, Large Language Models

1 INTRODUCTION

Students often struggle to find reliable and structured information when exploring or comparing academic majors. Existing sources such as Reddit or Quora may contain valuable insights, but they are typically unstructured, time-consuming to navigate, and lacking in personalization. Many students lack access to timely and expert academic guidance, especially when considering major changes or interdisciplinary transitions.

This project introduces the Major Information Consulting System (MICS), a web-based platform designed to assist students in exploring and comparing academic majors using AI-enhanced information retrieval. MICS targets three primary user groups: high school students deciding on a college major, university students evaluating a switch in discipline, and academic advisors seeking data-driven tools to support consultations.

To address their needs, MICS integrates web scraping, semantic vectorization, and retrieval-based response generation to provide accurate, structured, and personalized academic advice. The system leverages OpenAI embeddings for semantic representation, FastGPT's vector search engine for efficient retrieval, and curated content from forums like Reddit for grounding. In addition to question answering, MICS supports comparative analysis between majors and recommends fields of study based on user interests.

By automating the discovery, filtering, and synthesis of community-driven academic guidance, MICS demonstrates how modern information retrieval techniques can transform unstructured crowd-sourced knowledge into structured, actionable insights for student decision-making.

2 SYSTEM OVERVIEW

The MICS is designed to help students explore and compare academic majors by intelligently retrieving and presenting insights from online forums. The system integrates modern information retrieval techniques with large language model filtering and semantic search to provide accurate, personalized, and structured answers to user queries.

Figure 1 illustrates the overall architecture of MICS. The system is composed of five core modules:

- **Frontend UI (React)**: Provides a user-friendly interface for submitting queries and viewing retrieved answers.
- **Backend Server (Node.js + Express)**: Handles routing, processing logic, and communication between modules.
- **Data Scraper (Reddit API)**: Crawls relevant posts from Reddit based on user questions.
- **Query Filter (Google Gemini API)**: Classifies whether a user's question is academically/professionally relevant before proceeding with retrieval.
- **Knowledge Base (FastGPT + OpenAI Embeddings)**: Converts forum posts into vector representations and stores them for semantic retrieval.

When a user submits a query, it is first passed through the Gemini API to ensure relevance. If the question is judged to be unrelated to academic or professional topics, the system immediately returns a notification on the frontend, prompting the user to revise and resubmit their query. If the question is accepted, the system initiates a search and scraping process to gather relevant discussions on Reddit. These posts are then processed into embeddings using OpenAI's embedding models and stored in FastGPT's vector database. Finally, when a user query is issued, the system performs vector similarity search and returns curated responses via the frontend.

3 IMPLEMENTATION DETAILS

MICS is implemented using a modular architecture composed of multiple components built on modern web and AI technologies. The frontend is developed using React and communicates with the backend via RESTful APIs. The backend is built with Node.js and Express, responsible for coordinating scraping, LLM filtering, vector embedding, and querying.

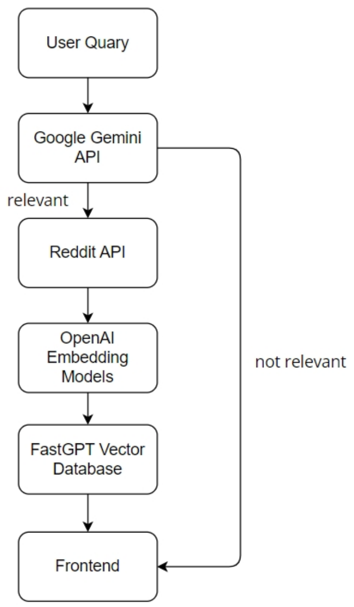


Figure 1: System architecture of the Major Information Consulting System

3.1 Query Filtering with Gemini

Each user query is sent to the Google Gemini API [1] using a structured prompt designed to classify whether the question is related to a professional or academic field. The system uses the Gemini 2.0 Flash model and prompts it to respond strictly with either "true" or "false". This ensures deterministic binary classification without natural language ambiguity.

The response is parsed and normalized to lowercase, and the system explicitly checks whether the answer includes the keyword "true" or "false". If the query is judged irrelevant (i.e., the answer is "false" or ambiguous), the frontend displays a message asking the user to revise their question. This filtering step prevents unnecessary computation and improves the relevance of downstream retrieval and generation.

The filtering logic is implemented as follows:

```
const prompt = `Determine if the following question is
    ↪ related to a professional field. Answer with "
    ↪ true" or "false" only.
Question: ${question}`;
const response = await ai.models.generateContent({
  model: "gemini-2.0-flash",
  contents: prompt,
});
const answer = response.text.trim().toLowerCase();
return answer.includes("true");
```

Listing 1: Gemini-based Query Filtering

3.2 Scraping and Data Collection

Instead of traditional headless browser scraping, our system retrieves Reddit content via its public JSON search API [2]. Given a user query, the system constructs a search URL and fetches the top 3 relevant posts in JSON format. To ensure compliance with Reddit's API policies, a custom User-Agent header is attached to each request.

The response is parsed to extract relevant fields such as post titles, content, and metadata, which are then used for downstream semantic filtering and embedding. If the HTTP request fails or the response is malformed, the system handles the error gracefully and returns an empty result to avoid crashing the pipeline.

The core logic is implemented as follows:

```
const url = `https://www.reddit.com/search.json?q=${
    ↪ encodeURIComponent(query)}&limit=3`;
const response = await fetch(url, {
  headers: {
    'User-Agent': 'Mozilla/5.0 (compatible; RedditSpider
    ↪ /1.0)'
  }
});
const json = await response.json();
const posts = json.data.children.map(child => child.data)
    ↪ ;
```

Listing 2: Reddit JSON API Scraper

3.3 Semantic Embedding and Storage

To support efficient and accurate information retrieval, all collected text data from Reddit is transformed into high-dimensional vector representations using OpenAI's `text-embedding-3-small` model [3]. These embeddings enable semantic search, where user queries are compared against stored entries based on vector similarity rather than exact keyword matching.

The embedded data is stored and indexed using FastGPT's vector knowledge base [4], which leverages PostgreSQL with the PGVector extension and HNSW indexing for high-speed similarity search. Each document is automatically segmented into smaller chunks using FastGPT's built-in text splitting engine in `auto` mode. The data is uploaded under the `qa` training type, allowing the system to generate internal question-answer pairs for more context-aware retrieval.

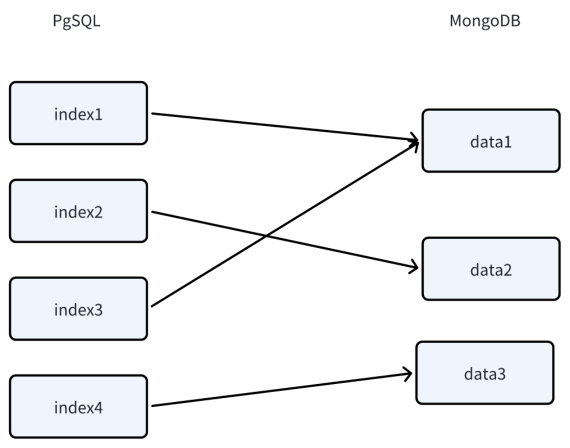


Figure 2: FastGPT vector-to-data mapping structure between PostgreSQL and MongoDB [4]

As shown in Figure 2, FastGPT adopts a decoupled storage architecture. Vector indexes are stored in PostgreSQL, while the original text data is stored in MongoDB. Each piece of content may correspond to multiple vector embeddings, enabling fine-grained and semantically rich retrieval. During a search, FastGPT first locates top-matching vectors, then resolves them back to their original content using shared identifiers between the databases.

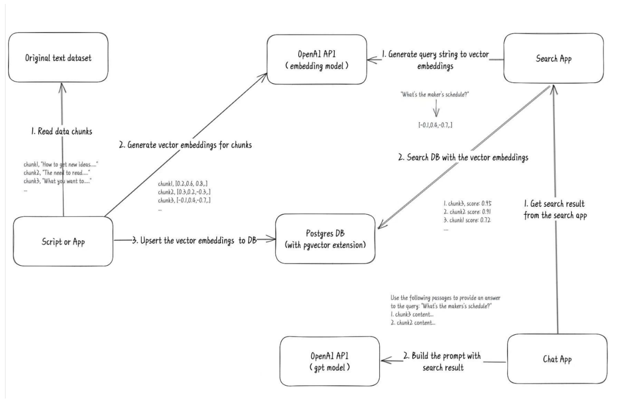

Original text dataset
OpenAI API (embedding model)
1. Generate query string to vector embeddings
Search App
"What's the maker's schedule?"
[-0.1,0.4,-0.7,.]
1. Read data chunks
2. Generate vector embeddings for chunks
2. Search DB with the vector embeddings
chunk1, "How to get new ideas....."
chunk2, "The need to read....."
chunk3, "What you want to....."
chunk1, [0.2,0.6, 0.8,.]
chunk2, [0.3,0.2,-0.3,.]
chunk3, [-0.1,0.4,-0.7,.]
1. chunk3, score: 0.95
2. chunk2 score: 0.91
3. chunk1 score: 0.72
1. Get search result from the search app
Script or App
3. Upsert the vector embeddings to DB
Postgres DB (with pgvector extension)
Use the following passages to provide an answer to the query: "What's the makers's schedule?"
1. chunk3 content...
2. chunk2 content...
OpenAI API (gpt model)
2. Build the prompt with search result
Chat App

Figure 3: FastGPT embedding and retrieval pipeline diagram [5]

Figure 3 illustrates the full retrieval-augmented generation (RAG) workflow, showing how raw data is processed, embedded, stored, and later retrieved to construct a query-aware response.

The ingestion process is handled programmatically via the FastGPT knowledge base API. For each user query and its corresponding crawled content, a payload containing the text, dataset ID, collection name, and chunking parameters is sent to the endpoint `/core/dataset/collection/create/text`. FastGPT then processes and stores the content into its vector database, making it available for downstream semantic retrieval.

A simplified version of the upload logic is shown below:

```javascript
const payload = {
  text: fileData,
  datasetId: config.fastGPT.datasetId,
  name: question,
  trainingType: "qa",
  chunkSettingMode: "auto"
};
const response = await fetch(`${baseURL}/core/dataset/
    ↪ collection/create/text`, {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(payload)
});
```

Listing 3: Uploading Text to FastGPT Knowledge Base

3.4 Answer Retrieval

During the retrieval stage, the system performs semantic similarity matching between the embedded user query and the stored knowledge base vectors. A minimum correlation threshold of 0.8 is applied to filter out loosely related results. To improve retrieval quality, the original query is first optimized using GPT-4o to resolve ambiguity and enhance semantic richness. This is especially helpful in follow-up or short questions.

Once the top-matching chunks are retrieved from FastGPT's vector database, the system formulates a prompt and sends it to FastGPT's chat API for completion. The request is structured with the user query wrapped as a message and sent via a POST request to the `/v1/chat/completions` endpoint.

A simplified version of the completion request is shown below:

```javascript
const body = {
  chatId: chatId,
  stream: false,
  detail: false,
  responseChatItemId: Date.now().toString(),
  variables: {},
  messages: [{ role: "user", content: question }]
};

const response = await fetch(`${baseURL}/v1/chat/
    ↪ completions`, {
  method: 'POST',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(body)
});
```

Listing 4: Submitting Query to FastGPT

The API returns a structured answer grounded in the semantically retrieved content. The system then formats and displays this answer in the frontend, completing the Retrieval-Augmented Generation (RAG) pipeline.

4 USAGE INSTRUCTIONS

The Major Information Consulting System (MICS) is a full-stack web application composed of a Node.js backend and a React frontend. To run the system locally, users must install dependencies and launch both services separately. The source code and installation scripts are available at: https://github.com/QinghongCai0612/Major-Information-Consulting-System.

4.1 Installation

To set up the system locally, first clone the repository using:

```
git clone https://github.com/QinghongCai0612/
Major-Information-Consulting-System.git
cd Major-Information-Consulting-System
```

4.2 Backend Setup

```
cd backend
npm install
npm start
```

This launches the backend server, which handles API logic for Reddit crawling, Gemini-based query filtering, and FastGPT integration. The backend runs on `http://localhost:3000`.

4.3 Frontend Setup

```
cd frontend
npm install
npm start
```

The frontend runs on `http://localhost:3001`, providing an interface where users can submit academic questions and receive curated results.

4.4 Configuration and API Keys

The system is preconfigured with working API credentials in the backend file `backend/config/config.js`, including:

- `google_Gemini_API_key`
- `fastGPT.baseURL`
- `fastGPT.general_authorization`
- `fastGPT.app_authorization`
- `fastGPT.datasetId`

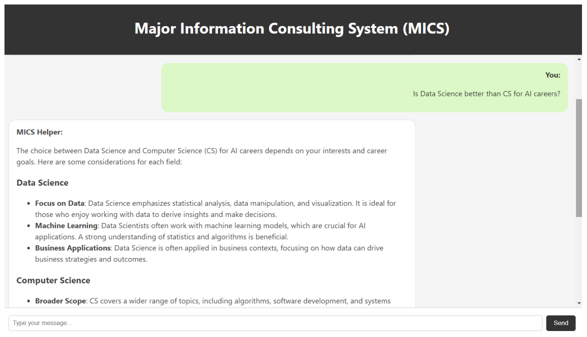
These values allow the system to run out-of-the-box. However, users who wish to use their own Google Gemini API key [1] or FastGPT API keys [6] can replace the values in `config.js`. The official documentation of Gemini and FastGPT provides instructions for obtaining valid API keys.

4.5 Example Usage

Once both frontend and backend are running, users can open `http://localhost:3001` in their browser and begin a session:

1. Enter a question such as: *"Is Data Science better than CS for AI careers?"*
2. The system checks the question's relevance using the Gemini API.
3. If the question is deemed relevant, Reddit posts are crawled, embedded, and stored in FastGPT.
4. A curated response is retrieved via semantic search and displayed in the frontend.

Figure 4 shows an example of a relevant academic question, *"Is Data Science better than CS for AI careers?"*, being successfully processed. The system returns a structured response comparing Data Science and Computer Science, highlighting factors such as focus, applicability, and interdisciplinary skills.



You:

Is Data Science better than CS for AI careers?

MICS Helper:

The choice between Data Science and Computer Science (CS) for AI careers depends on your interests and career goals. Here are some considerations for each field:

Data Science

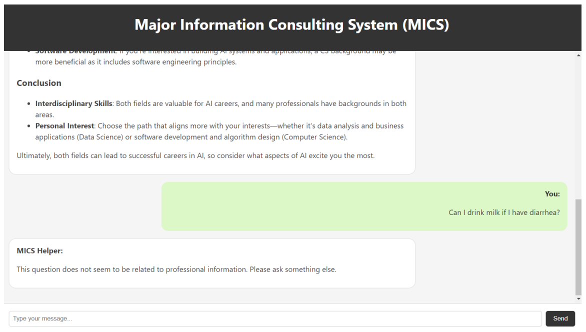
- **Focus on Data**: Data Science emphasizes statistical analysis, data manipulation, and visualization. It is ideal for those who enjoy working with data to derive insights and make decisions.
- **Machine Learning**: Data Scientists often work with machine learning models, which are crucial for AI applications. A strong understanding of statistics and algorithms is beneficial.
- **Business Applications**: Data Science is often applied in business contexts, focusing on how data can drive business strategies and outcomes.

Computer Science

- **Broader Scope**: CS covers a wider range of topics, including algorithms, software development, and systems

Figure 4: Relevant query processed and answered by MICS

Figure 5 illustrates a case where the user submits an unrelated question: *"Can I drink milk if I have diarrhea?"*. The system correctly identifies this query as irrelevant to academic or professional topics using Gemini filtering, and returns a polite rejection message. This mechanism ensures that the retrieval pipeline remains focused on meaningful educational inquiries.



- **Software Development**: If you're interested in building AI systems and applications, a CS background may be more beneficial as it includes software engineering principles.

Conclusion

- **Interdisciplinary Skills**: Both fields are valuable for AI careers, and many professionals have backgrounds in both areas.
- **Personal Interest**: Choose the path that aligns more with your interests—whether it's data analysis and business applications (Data Science) or software development and algorithm design (Computer Science).

Ultimately, both fields can lead to successful careers in AI, so consider what aspects of AI excite you the most.

You:

Can I drink milk if I have diarrhea?

MICS Helper:

This question does not seem to be related to professional information. Please ask something else.

Figure 5: Irrelevant query rejected by Gemini filtering

In some cases, the system may return a message such as `"No content returned"`. This typically indicates that the FastGPT account being used has insufficient knowledge base credits or AI processing credits. If users are running the system with their own FastGPT account, they are advised to monitor their credit usage and purchase additional quota as needed to ensure continued functionality.

5 EVALUATION AND RESULTS

To evaluate the effectiveness of the MICS, we conducted a total of 40 end-to-end dialogue tests. These tests were designed to assess both the relevance filtering mechanism and the overall system responsiveness.

5.1 Functional Accuracy

Out of 40 queries, 30 were academically relevant and 10 were irrelevant. Among the 30 relevant questions, 29 were successfully answered with complete retrieval and generation. One query was mistakenly rejected by the relevance filter, representing a single false negative. All 10 irrelevant queries were correctly blocked by the Gemini API, resulting in a 100% true negative rate.

Table 1: Relevance Filtering and Response Accuracy (N = 40)

Category	Count	Correctly Handled	Accuracy
Relevant Questions	30	29	96.7%
Irrelevant Questions	10	10	100%

5.2 Performance Metrics

For the 29 successfully answered relevant queries, we measured the full end-to-end response time, including Gemini filtering, Reddit scraping, embedding, semantic retrieval, and answer generation. The average response time was **7.794 seconds**.

5.3 Failure Case

The one false negative case involved a professionally relevant question that was overly concise, possibly lacking context for Gemini to classify it correctly. This highlights a limitation in the filtering step's sensitivity to short or ambiguous input.

Table 2: System Response Time

Metric	Value
Total processed queries	29
Average response time	7.794 seconds

Overall, the system demonstrated high accuracy in relevance judgment and robust response quality, with consistent end-to-end performance across realistic usage scenarios.

6 DISCUSSION AND FUTURE WORK

While the current implementation of the MICS demonstrates the feasibility and practicality of integrating LLM-based filtering with vector-based knowledge retrieval, several limitations remain.

6.1 Current Limitations

First, the system relies solely on Reddit as its data source. Although Reddit provides a rich repository of community-driven content, it lacks formal academic structure and coverage diversity. This constraint limits the system's ability to offer broader or more authoritative guidance.

Second, the current version does not support persistent user state or conversation history. Each user query is treated as an isolated interaction, which prevents follow-up or multi-turn academic advising.

Third, the system is dependent on a third-party platform (FastGPT) for vector storage and retrieval. This introduces quota-related limitations, as users with insufficient credits may encounter issues such as `"No content returned"`, affecting reliability.

6.2 Future Improvements

To address these limitations, several improvements are planned. First, the system will be extended to incorporate additional data sources such as Quora or Stack Exchange, enhancing diversity and depth of retrieved content.

Second, we plan to implement user session storage to support history-aware conversations. This will allow users to revisit prior questions and enable more personalized, contextual advising.

Lastly, performance and cost efficiency will be improved by introducing caching mechanisms and integrating reranking strategies after initial vector retrieval. This will reduce redundant computations and enhance the relevance of final responses.

7 CONCLUSION

This report presents the design, implementation, and evaluation of the Major Information Consulting System (MICS), an AI-powered academic assistant that integrates retrieval-augmented generation (RAG) with real-world forum data. The system combines Reddit-based content crawling, Gemini API relevance filtering, OpenAI semantic embedding, and FastGPT vector retrieval to deliver structured, accurate responses to academic and career-related questions.

Through end-to-end testing of 40 user queries, MICS demonstrated strong relevance filtering performance, with an overall accuracy of 96.7% on academic questions and a 100% rejection rate on irrelevant ones. The average response time of 7.794 seconds confirms the system's practical usability for near real-time interactions.

In addition to its technical contributions, MICS introduces an interpretable and extensible pipeline for transforming unstructured community knowledge into structured academic guidance. By decoupling filtering, embedding, and generation components, the system remains modular and adaptable for future extension.

Future improvements will focus on expanding data sources beyond Reddit, implementing persistent user state for multi-turn interactions, and optimizing retrieval with caching and reranking strategies. Overall, MICS provides a promising foundation for building domain-specific, LLM-enhanced academic support tools that bridge informal community insight with structured consulting intelligence.

ACKNOWLEDGEMENTS

We would like to thank the course instructors and teaching assistants of CS510 for their guidance and support throughout the development of this project. We are also grateful to the open-source communities behind FastGPT, Google Gemini, and Reddit, whose tools, APIs, and content formed the foundation of our system. Their contributions to accessible AI infrastructure and publicly available data were instrumental in making this work possible.

REFERENCES

[1] Google DeepMind. Google gemini api documentation. https://ai.google.dev/, 2024. Accessed: 2025-05-07.
[2] Reddit Inc. Reddit api reference. https://www.reddit.com/dev/api, 2023. Accessed: 2025-05-07.
[3] OpenAI. Openai text embedding models. https://platform.openai.com/docs/guides/embeddings, 2023. Accessed: 2025-05-07.
[4] FastGPT Contributors. Fastgpt documentation: Knowledge base vector architecture. https://doc.tryfastgpt.ai/docs/guide/knowledge_base/dataset_engine/, 2025. Accessed: 2025-05-07.
[5] FastGPT Team. Fastgpt faq documentation. https://kjqvjse66l.feishu.cn/docx/HtrgdT0pkonP4kxGx8qcu6XDnGh, 2025. Accessed: 2025-05-08.
[6] FastGPT Contributors. Fastgpt documentation: Api library. https://doc.tryfastgpt.ai/docs/guide/knowledge_base/api_dataset/, 2025. Accessed: 2025-05-08.