Vietnam General Confederation of Labor

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

HOANG DINH QUY VU - 521H0517
TRAN QUOC AN - 521H0385

Building a virtual assistant to support admissions information using RAG techniques

INFORMATION TECHNOLOGY PROJECT

COMPUTER SCIENCE

Instructor

Assoc. Prof. Dr. Le Anh Cuong

HO CHI MINH CITY, 2025

Vietnam General Confederation of Labor

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

HOANG DINH QUY VU - 521H0517
TRAN QUOC AN - 521H0385

Building a virtual assistant to support admissions information using RAG techniques

INFORMATION TECHNOLOGY PROJECT

COMPUTER SCIENCE

Instructor

Assoc. Prof. Dr. Le Anh Cuong

HO CHI MINH CITY, 2025

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to **Assoc. Prof. Dr Le Anh Cuong** for the valuable knowledge he has shared and his dedication to teaching the Information Technology Project. We truly appreciate his professionalism and passion for teaching, and we are grateful for the time and effort he has invested in guiding us throughout our learning journey.

His in-depth and detailed instruction has greatly enhanced our understanding of this development framework and its practical applications. Thanks to his teachings, we have gained the skills to effectively process data, yielding positive results in real-world applications.

We are also thankful for his continuous care and enthusiastic support during the learning process. He has always been willing to answer our questions and help us overcome any challenges we encountered while mastering the material.

We deeply admire his professionalism and passion for teaching. His dedication and enthusiasm have not only given us a strong foundation but also inspired us to pursue our passions and dreams.

With heartfelt gratitude, we wish **Assoc. Prof. Dr Le Anh Cuong** good health, happiness, and even greater success in imparting knowledge and supporting students. We hope his contributions continue to inspire and bring achievements to him and the entire Faculty of Information Technology.

Ho Chi Minh City, February 11, 2025

Author

(Sign and print full name)

[illegible] An

Hoàng Đình Quý Vũ Trần Quốc An

PROJECT COMPLETED

AT TON DUC THANG UNIVERSITY

I hereby certify that this is my own research work and was scientifically guided by Associate Professor Dr. Le Anh Cuong. The research content and results in this topic are honest and have not been published in any form before. The data in the tables for analysis, comments and evaluation were collected by the author himself from various sources clearly stated in the reference section.

In addition, the Project also uses some comments, evaluations as well as data from other authors and other organizations, all of which are cited and noted.

If any fraud is detected, I will take full responsibility for the content of my Project. Ton Duc Thang University is not related to any copyright and copyright violations caused by me during the implementation process (if any).

Ho Chi Minh City, February 11, 2025

Author

(Sign and print full name)

[illegible]

Hoàng Đình Quý Vũ

An

Trần Quốc An

ABSTRACT

University admissions in Vietnam include complicated and constantly changing information, making it difficult for students to obtain accurate assistance. To solve this, we suggest building an AI-powered chatbot that uses Retrieval-Augmented Generation (RAG) to improve information retrieval and response generation. Our research looks on how Advanced RAG, such as Self-RAG and Corrective-RAG, might increase inquiry understanding and response accuracy.

To improve data retrieval, we use a Query-to-SQL technique that provides structured access to admission score databases while overcoming the restrictions of traditional table chunking. Furthermore, we use online search to retrieve external, real-time information, which improves the chatbot's responses.

We use Recursive Chunking and Semantic Chunking to balance granularity and relevance in retrieved documents. The system is running on Hugging Face Spaces with Streamlit, which provides a user-friendly interface for interactive student interaction.

Our findings show that combining Advanced RAG methods, Query-to-SQL, and web search improves response accuracy, coherence, and adaptability. This strategy simplifies information access, lowers confusion, and enables students to make informed academic decisions.

TABLE OF CONTENTS

LIST OF IMAGES

LIST OF TABLES

LIST OF ABBREVIATIONS

RAG	Retrieval-Augmented Generation
CRAG	Corrective Retrieval-Augmented Generation
LLM	Large Language Model
NLP	Natural Language Processing
AI	Artificial Intelligence

CHAPTER 1. INTRODUCTION

1.1 Reason for choosing this topic

The university admission process in Vietnam involves a vast amount of information from multiple sources, including university websites, social media, and online forums. However, this information is often scattered, inconsistent, and frequently updated, making it difficult for students to find accurate and reliable guidance. Many students struggle to filter relevant details, leading to confusion and uncertainty in making important academic decisions.

Many students struggle with unclear or misleading information even after spending a lot of time looking for answers. Universities may also change their admission requirements once a year, which makes it necessary for students to continuously confirm information, adding to their anxiety and uncertainty. Even though some organizations provide online assistance, sometimes the responses are slow, general, or unable to address particular issues.

Given these challenges, an AI-powered chatbot using Retrieval-Augmented Generation (RAG) is a promising solution. By integrating real-time information retrieval with generative AI, the chatbot can provide students with accurate, context-aware, and up-to-date admission guidance. This approach helps streamline information access, reduce confusion, and support students in making well-informed academic decisions.

1.2 Goal of this topic

By integrating multiple information sources, the chatbot will help reduce misinformation, minimize the time students spend searching for details, and enhance the overall efficiency of the admission consultation process.

Therefore, the goal of this project is to create a chatbot that can provide trustworthy and customized university admission advice in Vietnam. The system will help students make informed decisions about their academic future, improve accessibility, and lessen information overload.

CHAPTER 2. BACKGROUND

2.1 LARGE LANGUAGE MODELS (LLM)

2.1.1 Overview

Large Language Models (LLMs) are advanced AI systems capable of processing and generating text with coherent communication. They exhibit strong generalization across various tasks and can achieve near-human performance in many applications.

The development of LLMs stems from the evolution of natural language processing (NLP), progressing from statistical methods to neural language models and, eventually, to pre-trained language models (PLMs). Traditional language models (LMs) are trained in a supervised manner for specific tasks, whereas PLMs utilize self-supervised learning on vast text corpora to acquire generalized representations applicable to multiple NLP tasks.

By fine-tuning for specific applications, PLMs outperform conventional language models. As larger PLMs demonstrate increased performance benefits, the transition to LLMs has involved up model parameters (ranging from scaling tens to hundreds of billions) and expanding training datasets (spanning gigabytes to terabytes). With appropriate prompts, LLMs can effectively generate accurate responses to various queries.

Fine-tuning them with task instructions data and aligning with human preferences enhances generalization to unseen tasks, improving zero-shot performance significantly and reducing misaligned behavior.

LLMs appear to have emergent abilities, such as reasoning, planning, decision-making, in-context learning, and answering in zero-shot settings. These abilities are acquired by them due to their gigantic scale even when the pre-trained LLMs are not trained specifically to possess these attributes.

LLMs can be widely adopted in diverse settings, including multi-modal, robotics, tool manipulation, question answering, and autonomous agents.

LLMs solve diverse tasks with human-level performance at the cost of slow training and inference, extensive hardware requirements, and higher running costs.

2.2 Retrieval-Augmented Generation (RAG)

2.2.1 Overview

Retrieval-Augmented Generation (RAG) is a technique that enhances Large Language Models (LLMs) by incorporating knowledge from external databases. It addresses limitations of LLMs such as:

Hallucinations

Outdated knowledge

Non-transparent reasoning

Retrieval-Augmented Generation (RAG) enhances language models by retrieving relevant document chunks from an external knowledge base using semantic similarity techniques. By incorporating external knowledge, RAG helps mitigate the risk of generating inaccurate information. This approach seamlessly integrates an LLM's inherent knowledge with large, dynamic external databases.

In a typical RAG workflow, a user submits a query to an LLM. Since the model's knowledge is limited to its pre-training data, it may not have access to the latest updates or recent developments. RAG addresses this limitation by retrieving relevant information from external sources, merging the retrieved content with the user's query to create a more informative prompt. This enriched input enables the LLM to generate more accurate and well-informed responses.

2.2.2 Process

The traditional process of RAG that includes indexing, retrieval and generation which is also characterized as a "Retrieve-Read" framework

a. Indexing

Indexing begins with cleaning and extracting raw data from various formats such as PDF, HTML, Word, and Markdown. This data is then standardized into plain text.

To address the context limitations of language models, the text is divided into smaller, manageable segments. These segments are transformed into vector representations using an embedding model and stored in a vector database, a key step for facilitating efficient similarity searches during retrieval.

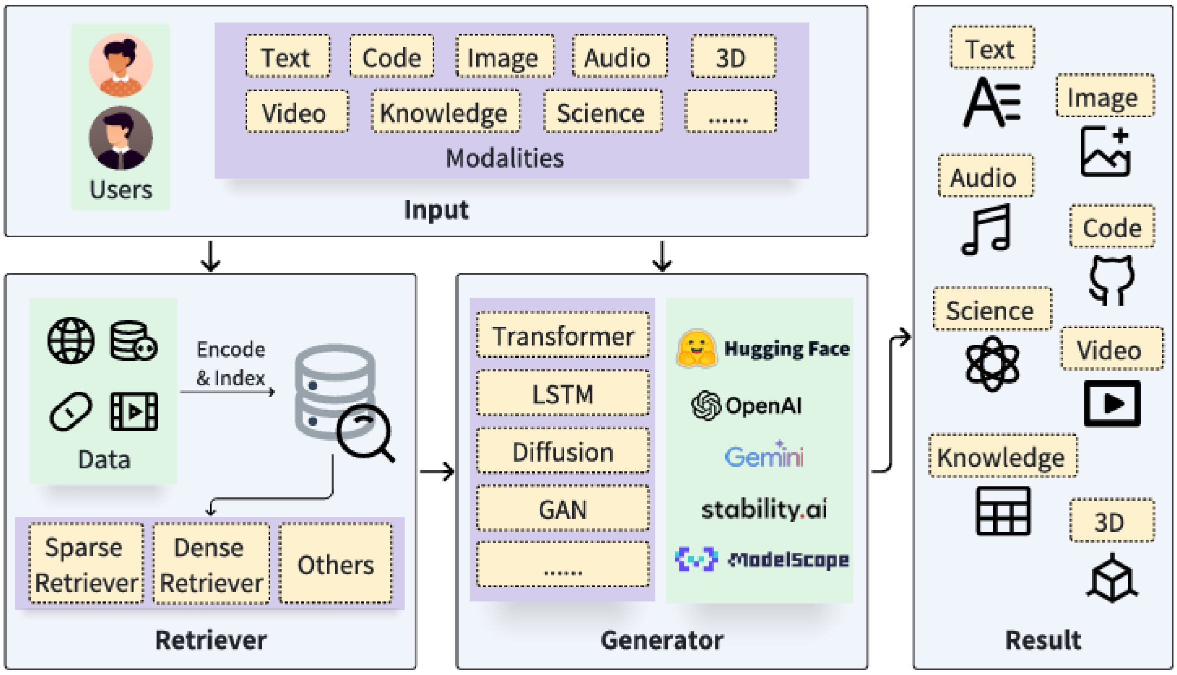


Figure 1. Generic RAG Architecture

b. Retrieval

When a user submits a query, the RAG system converts it into a vector representation using the same encoding model from the indexing phase. It then calculates similarity scores between the query vector and the stored vectors in the indexed corpus. The system ranks and retrieves the top K most relevant chunks based on similarity, which are then incorporated as additional context in the prompt.

c. **Generation**

The user's query, along with the retrieved documents, is structured into a coherent prompt. A large language model then generates a response, either by leveraging its internal parametric knowledge or by strictly relying on the retrieved information. In ongoing conversations, previous dialogue history can be included in the prompt, enabling seamless multi-turn interactions.

2.2.3 Advantages

a. Improved Accuracy & Reliability

RAG reduces hallucination by grounding responses in external knowledge sources

It ensures that generated content is based on factual and up-to-date information

b. Access to Up-to-Date Information

Unlike traditional language models that rely only on pre-trained knowledge, RAG can retrieve the latest data from external sources, making it ideal for dynamic domains like news, finance, and research.

c. Better Handling of Long-Tail Queries

It enhances responses to niche or domain-specific queries by retrieving specialized information from relevant sources.

d. Reduced Memory & Training Requirements

Instead of retraining large language models with new data, RAG can dynamically fetch updated knowledge, making it more efficient and scalable.

e. Domain Adaptability

Easily integrates with custom knowledge bases, making it useful for industries like healthcare, law, and technical support.

2.2.4 Challenges

a. Retrieval challenges

   The retrieval phase often struggles with balancing precision and recall, which can result in selecting irrelevant or misaligned chunks while overlooking crucial information.

b. Generation difficulties

   The model may generate hallucinated content that is not grounded in the retrieved information.

   Issues such as irrelevance, toxicity, and bias can negatively impact the quality and trustworthiness of responses.

c. Augmentation Hurdles

   Integrating retrieved information into different tasks can be difficult, sometimes leading to incoherent or fragmented outputs.

   Redundancy may occur when similar content is retrieved from multiple sources, resulting in repetitive responses.

   Determining the importance and relevance of various passages while maintaining stylistic and tonal consistency adds complexity.

d. Context limitations

   A single retrieval attempt based on the initial query may not always provide enough context for accurate responses.

   There is a risk that the generation model may overly rely on retrieved content, leading to responses that merely repeat the extracted information rather than offering meaningful synthesis or insights.

2.3 Promp Engineering

2.3.1 Overview

Prompt engineering involves crafting effective prompts to maximize the performance of large language models (LLMs) for specific tasks. Since LLMs are highly sensitive to prompt design, identifying optimal prompts often demands significant manual effort. Non-experts in AI may find it challenging to clearly convey their intended task to an LLM, leading to an ad-hoc rather than systematic approach to prompt engineering. The primary objective is to determine the optimal prompt ***p**** that yields the best results on the given dataset ***D*** when applied to specific LLM-based task model.

More specifically, we assume all datasets can be formatted as textual input-output pairs, i.e. ***D = {(x,y)}***

The prompt engineering problem can be described as:

$$p^* = \arg\max_{p} \sum_{(x,y)\in D_{dev}} f(\mathcal{M}_{task}(x;p), y)$$

Figure 2. Formula of prompt engineering

where ***Mtask(x;p)*** is the output generated by the task model when conditioning on the prompt p, and f is a per-example evaluation function. For example, if the evaluation metric is exact match, ***f(Mtask(x;p),y) = 1[Mtask(x;p) = y].***

2.3.1 Types of Prompt Engineering

a. **Zero-shot Prompting**

   Asking the model to perform a task without providing any examples
   Use case: When the model has sufficient pre-trained knowledge to handle the request

b. **Few-shot Prompting**

   Providing a few examples in the prompt to guide the model's response

Use case: When the model need guidance on formatting, tone or logic

c. **Chain-of-Thought Prompting**

Encouraging the model to break down its reasoning step by step

Use case: Useful for complex reasoning tasks such as math problems, logical analysis, or multi-step decision-making

d. **Self-Consistency Prompting**

Generating multiple responses and selecting the most consistent answer

Use case: Enhance reliability in reasoning-based tasks

CHAPTER 3. METHOD

3.1 Corrective Retrieval-Augmented Generation (CRAG)

3.1.1 Overview

Corrective Retrieval Augmented Generation (CRAG) is a method designed to improve the robustness of generation in scenarios where the retriever returns inaccurate results. CRAG aims to self-correct the results of the retriever and improve the utilization of documents for augmenting generation. It is a plug-and-play method that can be seamlessly coupled with various RAG-based approaches.

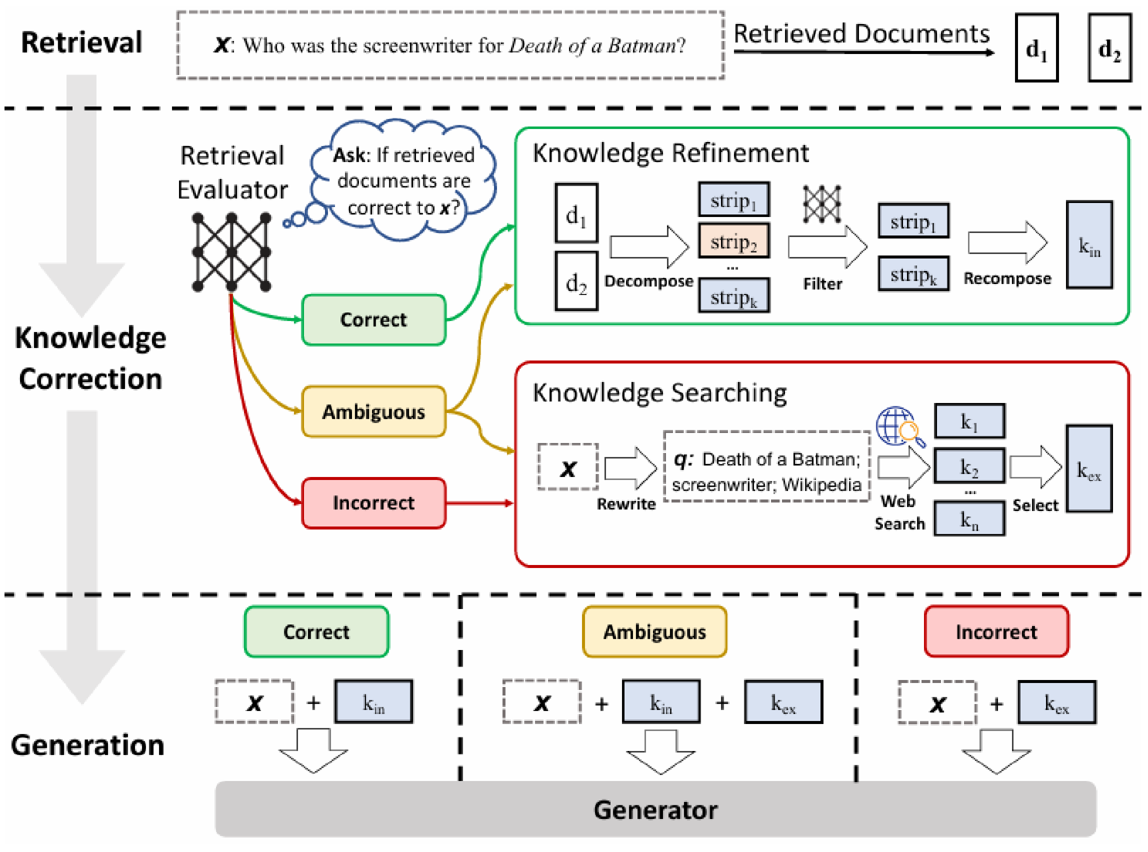


Figure 3. An overview of CRAG

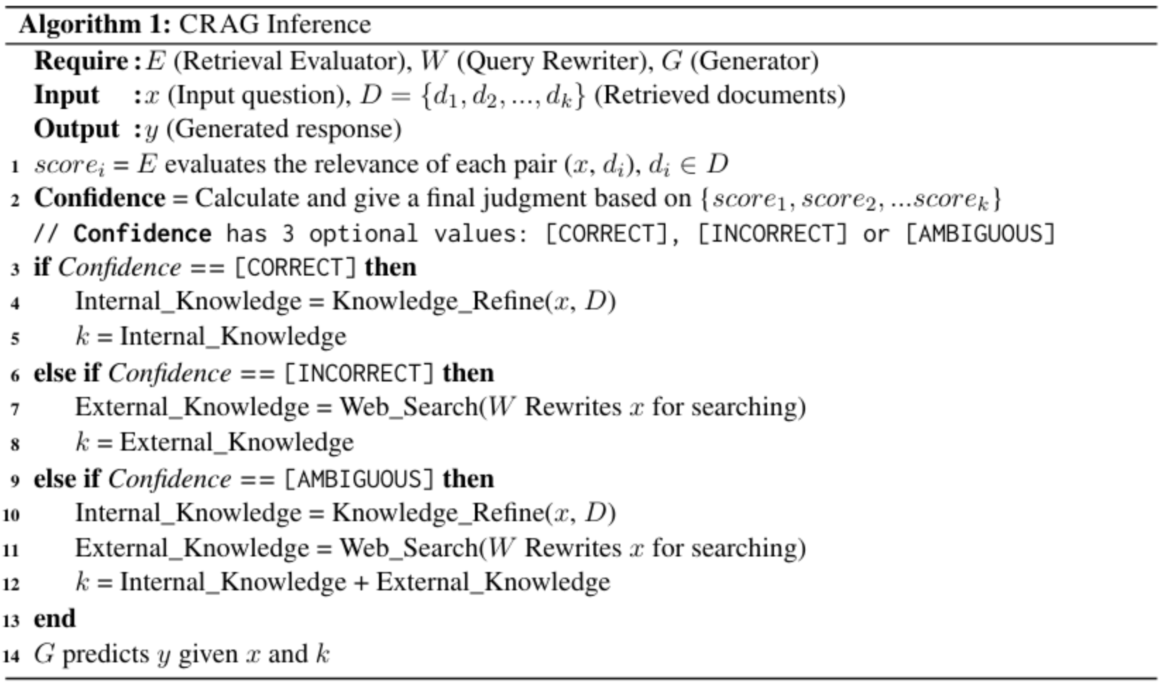
Key aspects of CRAG include:

Retrieval Evaluator: A lightweight retrieval evaluator is designed to assess the overall quality of retrieved documents for a query. It returns a confidence degree based on which different knowledge retrieval actions can be triggered.

Knowledge Retrieval Actions: Based on the confidence score, three types of actions are designed: Correct, Incorrect, and Ambiguous.

Web Searches: Large-scale web searches serve as a supplementary method to improve retrieval accuracy when the initial retriever provides incorrect results.

Decompose-then-Recompose Algorithm: This approach processes retrieved documents by isolating essential information while filtering out irrelevant content. It refines the retrieved data, optimizing the extraction of key insights while minimizing unnecessary elements, ultimately enhancing the quality and effectiveness of the retrieved information.



```
Algorithm 1: CRAG Inference
Require : $E$ (Retrieval Evaluator), $W$ (Query Rewriter), $G$ (Generator)
Input   : $x$ (Input question), $D = \{d_1, d_2, ..., d_k\}$ (Retrieved documents)
Output  : $y$ (Generated response)
$score_i$ = $E$ evaluates the relevance of each pair ($x$, $d_i$), $d_i \in D$
Confidence = Calculate and give a final judgment based on $\{score_1, score_2, ...score_k\}$
// Confidence has 3 optional values: [CORRECT], [INCORRECT] or [AMBIGUOUS]
if Confidence == [CORRECT] then
    Internal_Knowledge = Knowledge_Refine($x$, $D$)
    $k$ = Internal_Knowledge
else if Confidence == [INCORRECT] then
    External_Knowledge = Web_Search($W$ Rewrites $x$ for searching)
    $k$ = External_Knowledge
else if Confidence == [AMBIGUOUS] then
    Internal_Knowledge = Knowledge_Refine($x$, $D$)
    External_Knowledge = Web_Search($W$ Rewrites $x$ for searching)
    $k$ = Internal_Knowledge + External_Knowledge
end
$G$ predicts $y$ given $x$ and $k$
```

Figure 4. CRAG Inference Algorithm: A decision-making framework for evaluating retrieved knowledge and refining responses in a retrieval-augmented generation (RAG) system

3.1.2 Why CRAG?

a. Improved Retrieval Accuracy

Naïve RAG: Retrieves chunks based solely on initial query embedding similarity, often leading to misalignment or missing important context.

Corrective RAG: Introduces **query expansion, re-ranking, or feedback loops** to refine retrieval, ensuring more relevant and complete information is retrieved.

b. Reduction in hallucination

Naïve RAG: The model may hallucinate by generating responses that are not fully supported by the retrieved context.

Corrective RAG: Implements **validation mechanisms** (e.g., confidence scoring, fact verification) to ensure the generation phase stays grounded in retrieved evidence.

c. Adaptive Retrieval with multi-turn feedback

Naïve RAG: Performs a single retrieval step based on the initial query.

Corrective RAG: Uses an **iterative refinement approach** where the model can modify the query or re-retrieve relevant documents based on previous responses and user interactions.

d. Handling noise or redundant data

Naïve RAG: Struggles with selecting the most relevant chunks when multiple documents contain overlapping information.

Corrective RAG: Applies **de-duplication, weighting mechanisms, or fusion strategies** to consolidate information and remove redundant content.

e. Contextual re-ranking of retrieved chunks

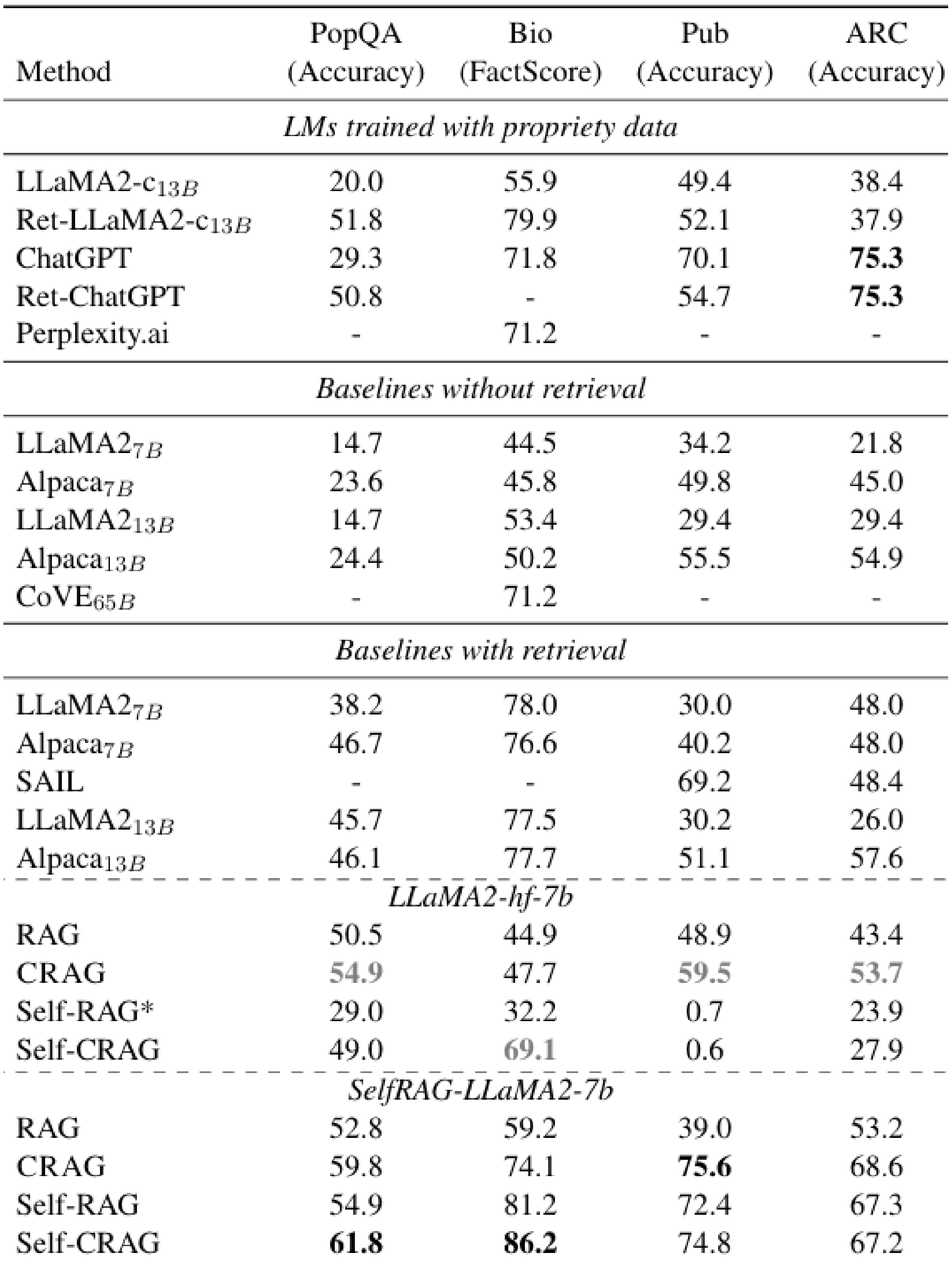
Naïve RAG: Uses raw similarity scores to rank retrieved documents.

Corrective RAG: Incorporates **re-ranking models** (e.g., BERT-based rankers) to prioritize documents based on contextual relevance rather than just embedding similarity.

f. Improved response coherence and consistency

Naïve RAG: May generate disjointed or inconsistent responses if retrieved documents contain conflicting information.

Corrective RAG: Uses post-retrieval filtering or response calibration mechanisms to ensure that the output is logically consistent.



Method	PopQA (Accuracy)	Bio (FactScore)	Pub (Accuracy)	ARC (Accuracy)
LMs trained with propriety data				
LLaMA2-c$_{13B}$	20.0	55.9	49.4	38.4
Ret-LLaMA2-c$_{13B}$	51.8	79.9	52.1	37.9
ChatGPT	29.3	71.8	70.1	**75.3**
Ret-ChatGPT	50.8	-	54.7	**75.3**
Perplexity.ai	-	71.2	-	-
Baselines without retrieval				
LLaMA2$_{7B}$	14.7	44.5	34.2	21.8
Alpaca$_{7B}$	23.6	45.8	49.8	45.0
LLaMA2$_{13B}$	14.7	53.4	29.4	29.4
Alpaca$_{13B}$	24.4	50.2	55.5	54.9
CoVE$_{65B}$	-	71.2	-	-
Baselines with retrieval				
LLaMA2$_{7B}$	38.2	78.0	30.0	48.0
Alpaca$_{7B}$	46.7	76.6	40.2	48.0
SAIL	-	-	69.2	48.4
LLaMA2$_{13B}$	45.7	77.5	30.2	26.0
Alpaca$_{13B}$	46.1	77.7	51.1	57.6
LLaMA2-hf-7b				
RAG	50.5	44.9	48.9	43.4
CRAG	**54.9**	47.7	**59.5**	**53.7**
Self-RAG*	29.0	32.2	0.7	23.9
Self-CRAG	49.0	**69.1**	0.6	27.9
SelfRAG-LLaMA2-7b				
RAG	52.8	59.2	39.0	53.2
CRAG	59.8	74.1	**75.6**	68.6
Self-RAG	54.9	81.2	72.4	67.3
Self-CRAG	**61.8**	**86.2**	74.8	67.2

Figure 5. Performance comparison of different language models on various benchmarks with and without retrieval-based approaches, highlighting the effectiveness of RAG and CRAG methods

3.2 MODELS

In this project we use Gemini-1.5-pro to generate contents, transform, route and construct user's query. Using Gemma2-9b-it to answer the question based on the knowledges that are retrieved from CRAG

3.2.1 Gemini-1.5-pro

Gemini 1.5 Pro is an advanced generative AI model designed to excel in a wide range of applications, including text creation, image recognition, and coding support.

Developed by Google, this large language model (LLM) is multimodal, meaning it can seamlessly process and generate content in multiple formats, such as text, images, audio, and video.

With an extensive context window and a knowledge cutoff of November 2023, Gemini 1.5 Pro is well-equipped to provide accurate and up-to-date responses across various domains. Let's explore its key features in more detail.

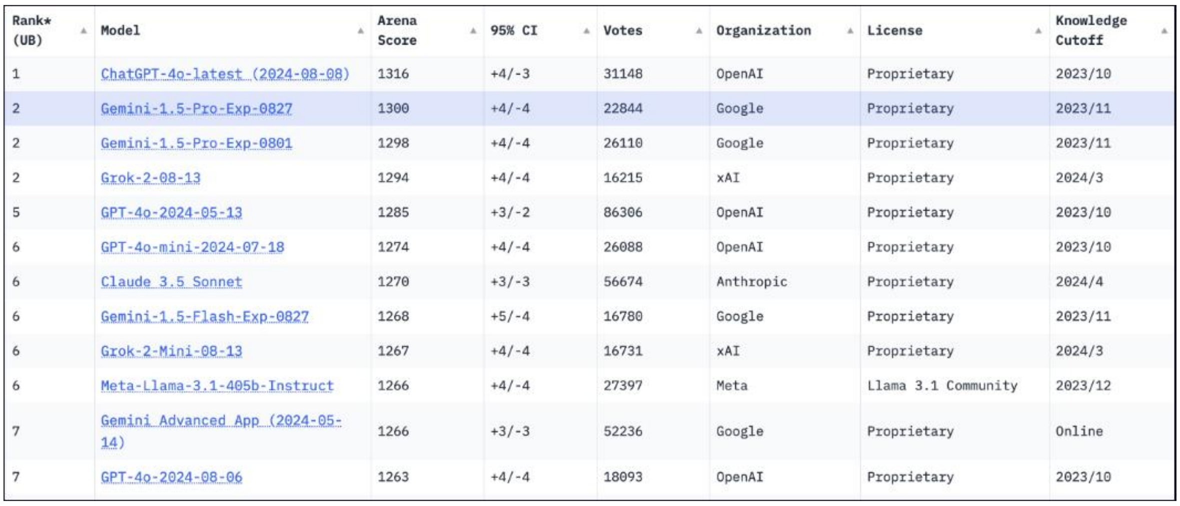
Why gemini-1.5-pro?

Model Size: Gemini 1.5 Pro boasts **over 200 billion parameters**, enabling it to understand intricate language patterns and produce high-quality, nuanced outputs. This large parameter count significantly enhances its ability to generate coherent and contextually relevant content.

Context Window: The model supports a **128,000-token context window**, ensuring **coherence in extended conversations** and enabling effective handling of long-form content.

Knowledge Cutoff: With a **knowledge cutoff date of November 2023**, Gemini 1.5 Pro remains **relevant and informed**, incorporating recent advancements across various industries. This makes it a valuable tool for users who require the latest data and insights.

Multimodal Capabilities: The model can **process and generate content in multiple formats**, including text, images, audio, and video.



Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	ChatGPT-4o-latest (2024-08-08)	1316	+4/-3	31148	OpenAI	Proprietary	2023/10
2	Gemini-1.5-Pro-Exp-0827	1300	+4/-4	22844	Google	Proprietary	2023/11
2	Gemini-1.5-Pro-Exp-0801	1298	+4/-4	26110	Google	Proprietary	2023/11
2	Grok-2-08-13	1294	+4/-4	16215	xAI	Proprietary	2024/3
5	GPT-4o-2024-05-13	1285	+3/-2	86306	OpenAI	Proprietary	2023/10
6	GPT-4o-mini-2024-07-18	1274	+4/-4	26088	OpenAI	Proprietary	2023/10
6	Claude 3.5 Sonnet	1270	+3/-3	56674	Anthropic	Proprietary	2024/4
6	Gemini-1.5-Flash-Exp-0827	1268	+5/-4	16780	Google	Proprietary	2023/11
6	Grok-2-Mini-08-13	1267	+4/-4	16731	xAI	Proprietary	2024/3
6	Meta-Llama-3.1-405b-Instruct	1266	+4/-4	27397	Meta	Llama 3.1 Community	2023/12
7	Gemini Advanced App (2024-05-14)	1266	+3/-3	52236	Google	Proprietary	Online
7	GPT-4o-2024-08-06	1263	+4/-4	18093	OpenAI	Proprietary	2023/10

Figure 6. Leaderboard of AI models ranked by Arena Score, comparing performance, votes, organization, license type, and knowledge cutoff dates

3.2.2 Gemma-2-9b-it

Gemma is a **lightweight, state-of-the-art family of open AI models**, developed using the same research and technology behind the **Gemini** models.

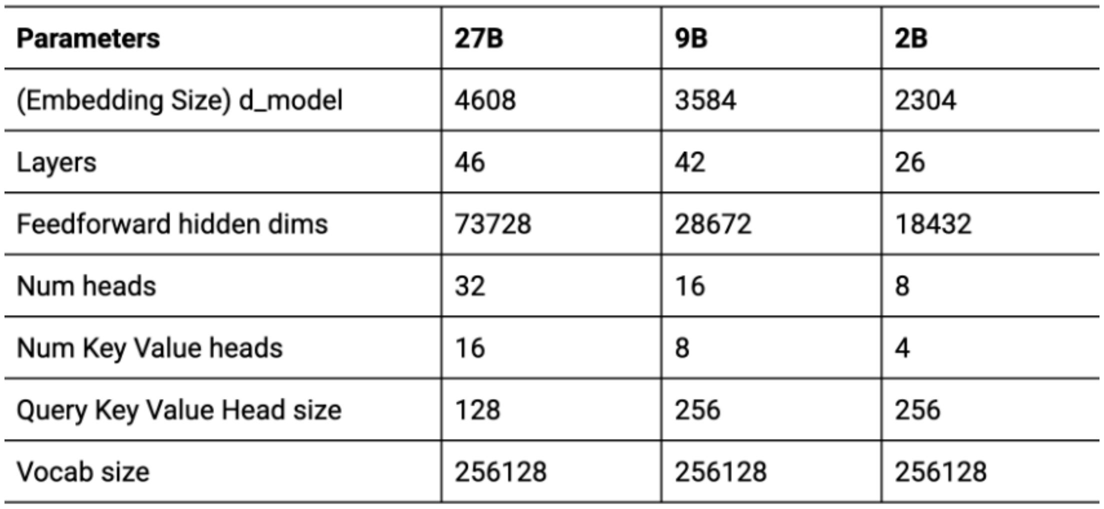
Different versions of Gemma are optimized for **various use cases and modalities**, including:

- **Single-modality models** (processing text input and generating text output).
- **Specialized coding models** tailored for programming-related tasks.
- **Multimodal models** that can process both text and images as input while generating text output.
- **Models of varying sizes** to accommodate different hardware, inference speed, and computational constraints.
- **Innovative architectures** that introduce novel approaches to model design.

Why Gemma-2-9b-it?

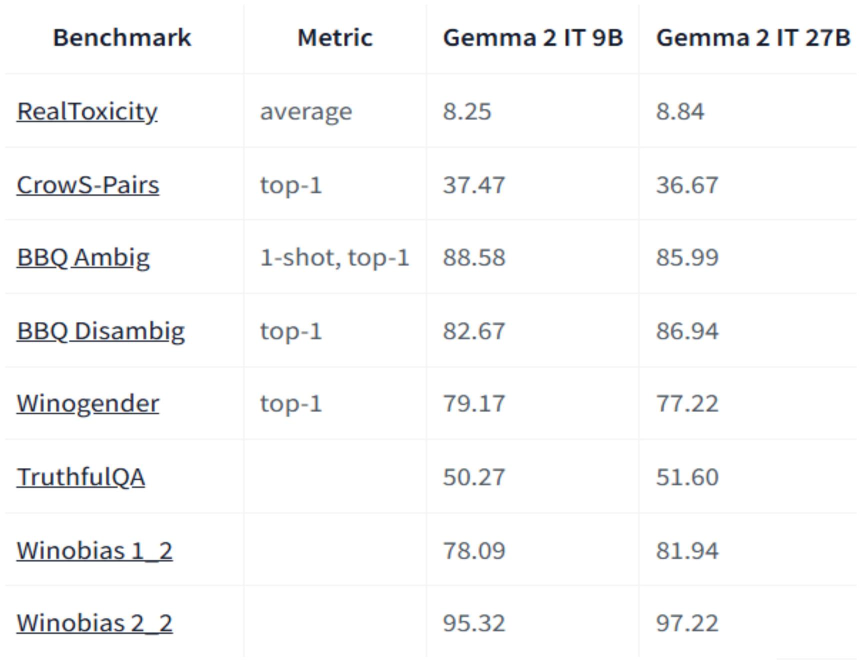
In this project, we use CRAG to retrieve the documents from vector database. In addition, CRAG also supports additional documents if the retrieved documents from vector database that are irrelevant to the user query. Thus, we need a model with a large number of tokens to be able to contain all retrieved documents.

Gemma-2-9b-it architecture



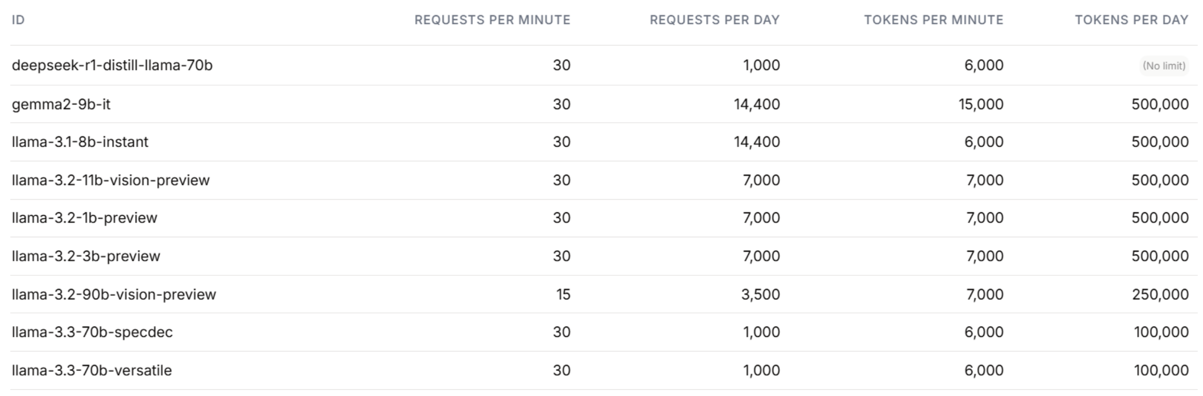
Parameters	27B	9B	2B
(Embedding Size) d_model	4608	3584	2304
Layers	46	42	26
Feedforward hidden dims	73728	28672	18432
Num heads	32	16	8
Num Key Value heads	16	8	4
Query Key Value Head size	128	256	256
Vocab size	256128	256128	256128

Figure 7. Gemma architecture



Benchmark	Metric	Gemma 2 IT 9B	Gemma 2 IT 27B
RealToxicity	average	8.25	8.84
CrowS-Pairs	top-1	37.47	36.67
BBQ Ambig	1-shot, top-1	88.58	85.99
BBQ Disambig	top-1	82.67	86.94
Winogender	top-1	79.17	77.22
TruthfulQA		50.27	51.60
Winobias 1_2		78.09	81.94
Winobias 2_2		95.32	97.22

Figure 8. Benchmark performance comparison between Gemma 2 IT 9B and Gemma 2 IT 27B



ID	REQUESTS PER MINUTE	REQUESTS PER DAY	TOKENS PER MINUTE	TOKENS PER DAY
deepseek-r1-distill-llama-70b	30	1,000	6,000	(No limit)
gemma2-9b-it	30	14,400	15,000	500,000
llama-3.1-8b-instant	30	14,400	6,000	500,000
llama-3.2-11b-vision-preview	30	7,000	7,000	500,000
llama-3.2-1b-preview	30	7,000	7,000	500,000
llama-3.2-3b-preview	30	7,000	7,000	500,000
llama-3.2-90b-vision-preview	15	3,500	7,000	250,000
llama-3.3-70b-specdec	30	1,000	6,000	100,000
llama-3.3-70b-versatile	30	1,000	6,000	100,000

Figure 9. Comparison of request and token limits for various AI models

3.3 Vector Database

Vector databases are a modern approach to managing abstract data representations generated by advanced machine learning models, such as deep learning architectures. These representations, known as vectors or embeddings, serve as compressed versions of the data used to train models for various tasks, including sentiment analysis, speech recognition, and object detection.

Qdrant is a **vector similarity search engine** designed for **production-ready** use, offering a user-friendly API for storing, searching, and managing vectors (referred to as points) along with an **additional payload**. These payloads serve as extra metadata that help refine searches and enhance the information provided to users.

Figure 10. Qrant Vector Database

A vector in this context is a mathematical representation of an object or data point, where elements of the vector implicitly or explicitly correspond to specific features or attributes of the object.

Vector databases use specialized indexing techniques such as **Hierarchical Navigable Small World (HNSW)** for Approximate Nearest Neighbors and **Product Quantization** to enable **fast similarity and semantic searches**. These methods allow users to efficiently find vectors that are **closest to a query vector** based on various distance metrics.

Key distance metrics in Qdrant:

- **Cosine Similarity** – Measures how aligned two vectors are. Values range from **-1 (completely dissimilar) to 1 (identical)**. Often used in text similarity comparisons.
- **Dot Product** – Similar to cosine similarity but considers vector magnitude, making it useful when vector values represent frequency-based data.
- **Euclidean Distance** – Computes the straight-line distance between two points in space, commonly used in machine learning for similarity comparisons.

3.4 Embedding model

An **embedding model** is a machine learning model that transforms raw data (such as text, images, or audio) into **dense numerical vector representations**. These representations, known as **embeddings**, capture the semantic meaning of the data and enable efficient similarity searches in a **vector store**.

In this project, we use ***vietnames-document-embedding***, is the Document Embedding Model for Vietnamese language with context length up to 8096 tokens. This model is a specialized long text-embedding trained specifically for the Vietnamese language, which is built upon gte-multilingual and trained using the Multi-Negative Ranking Loss, Matryoshka2dLoss and SimilarityLoss.

This model is suitable for this project because it has large context lengths that can store semantic chunks, it is essential for searching by using distance formular to find the relevant chunks

CHAPTER 4. SYSTEM

The system is designed to intelligently route queries through the most appropriate retrieval and response generation pipeline, ensuring that users receive accurate, contextually relevant, and up-to-date information. By incorporating a multi-source approach—RAG retrieval, SQL database querying, and web search retrieval—the system can dynamically adapt to different types of user inquiries.

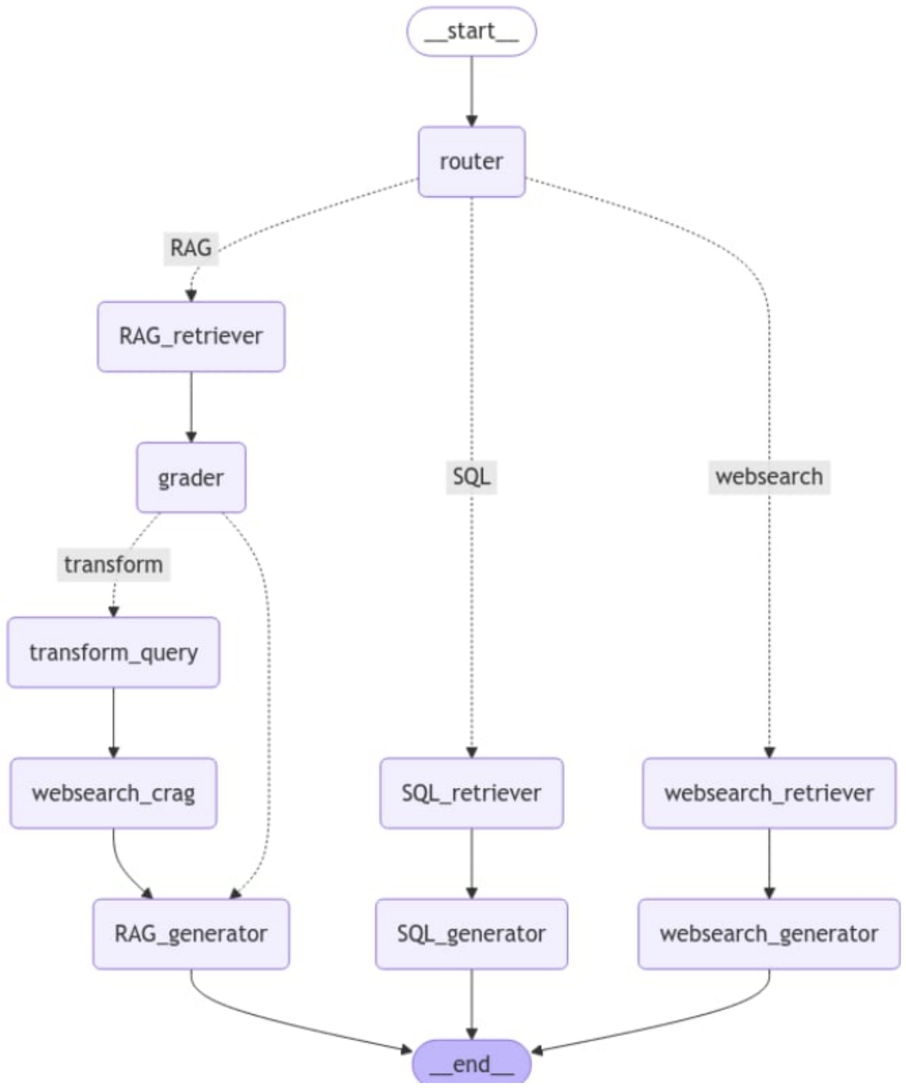


Figure 11. Flow chart of the system

A key innovation in our system is the Corrective RAG mechanism, which enhances traditional RAG-based retrieval by incorporating query transformation and grading. This ensures that retrieved documents are highly relevant, and in cases where gaps exist, additional corrections or web-based retrievals are performed to improve response accuracy. The system architecture consists of a router-based decision mechanism, which classifies incoming queries into three main categories:

RAG-based retrieval for the queries that are related to university information, admission information by year of each university, information about faculties or programs at university, …

SQL-based retrieval for structed data queries is the question related to benchmarks

Web search retrieval for external information gathering

4.1 Query Analysis Block

Query analysis is the first block of the system where the query is received from the user, it is responsible for deciding which type of query the query belongs to (RAG, SQL or websearching), from there the query direction goes to the right query blocks.

In query analysis block, gemini-1.5-pro is the model is responsible for classifying the query, returning the result to the router from which it can make a classification decision.

Few-shot is the type of prompt engineering used in this block, by giving examples so that the model can recognize each type of query and from there make decisions

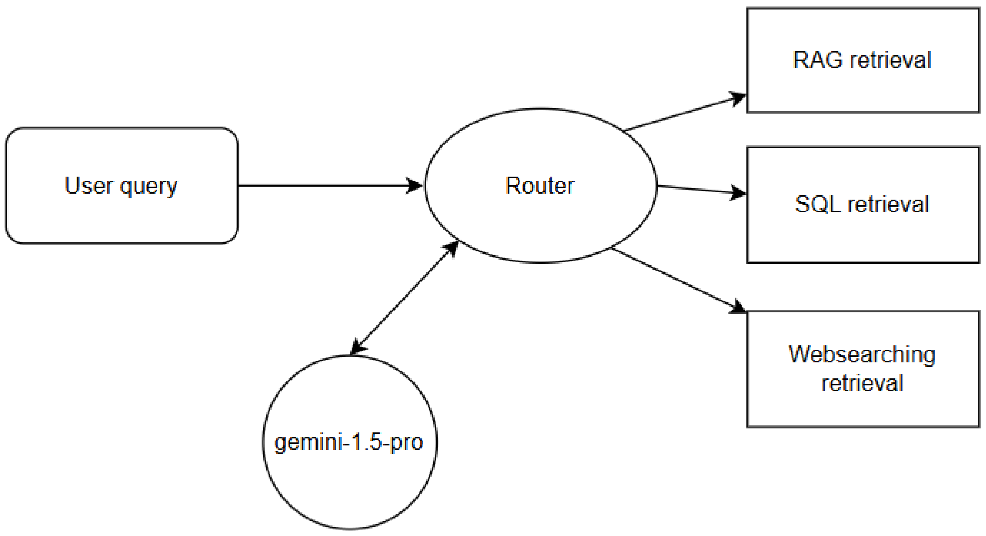


Figure 12. Architecture of Query analysis block

4.2 RAG Retrieval Block

The RAG retrieval block in the system is responsible for retrieving the relevant documents from a knowledge base that related to university information, admission information by year of each university, information about faculties or majors at university and generating response based on them. In this block, Corrective RAG is used instead of naïve RAG because the retrieved documents have pass to a node that evaluates whether the document is relevant to the user's query, otherwise it is

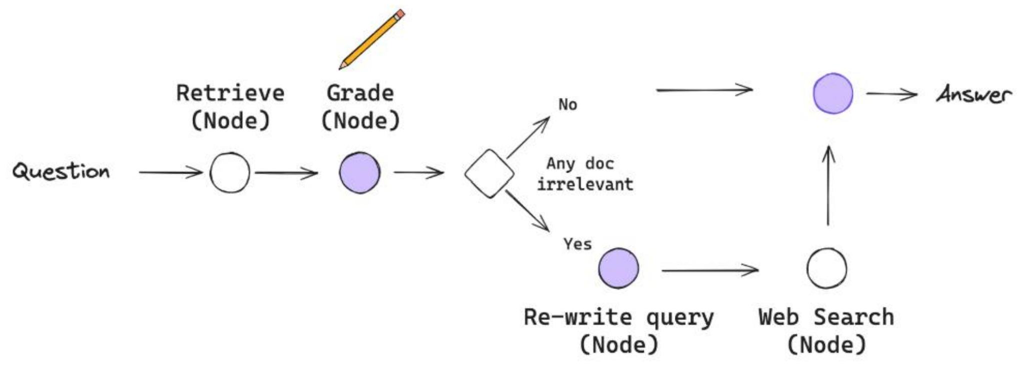


Figure 13. Architecture of RAG retrieval block

supplemented by searching the web to add more documents to enhance the accuracy of the answer

Components of the RAG Retrieval Block:

RAG retriever: this module retrieves relevant documents from a predefined knowledge based on the user query

Grader: the retrieved documents are evaluated by gemini-1.5-pro, this model received a query – document pair to grade based on the relevance between the document and the query, on a 5-point scale, any document that get at least 3 points is considered relevant.

Transforming query (optional correction step): If any retrieved document is grade to be irrelevant, the query is passed to the transforming node, where the original query is examined and evaluated if it is a complex query. If so, it is broken down into several smaller queries. If not, the query is rewritten to match more relevant documents.

Web searching node: this node will call the web searching tool function to receive the transformed query, the result returned is the documents related to the above query from the web searching results

Generating node: this is the final step of the RAG retrieval block; it involves generating a response based on the retrieved document. In this node, gemma-2-9b-it is responsible for generating response.

4.3 SQL Retrieval Block

The SQL retrieval block in the system is responsible for retrieving structured data from a database. This block will handle benchmark scores over the years for each major and each university

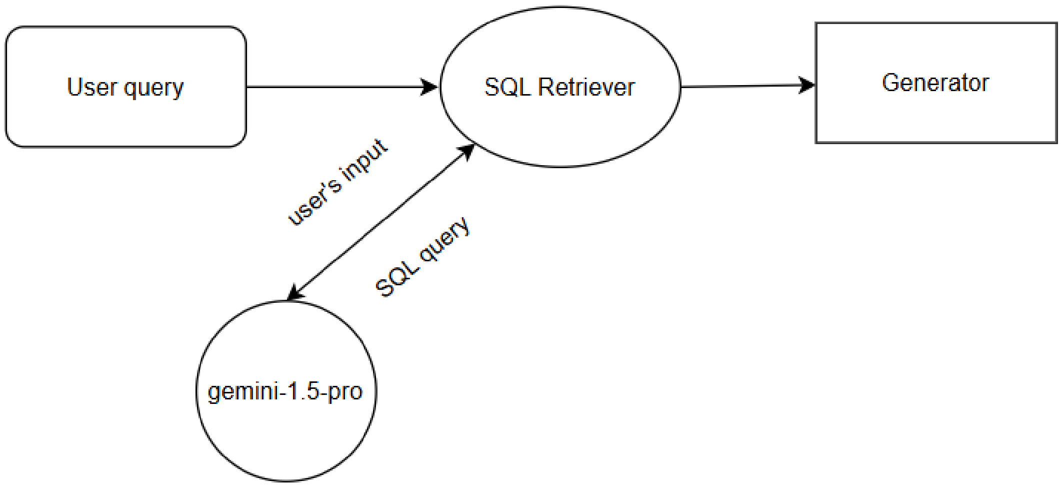


Figure 14. Architecture of SQL retrieval block

SQL retrieval block components:

- SQL retriever: this module queries the database based on a SQL query that is transformed by user's input. Gemini-1.5-pro is responsible for transforming from user input to SQL query.

- Generating node: this is the final step, the data retrieved from the database will be generated into a single document and passed into the gemma-2-9b-it along with the query from the user to generate response

4.4 Web Searching Retrieval Block

The web searching retrieval block in the system is designed to retrieve from external sources when a query is not related to university admission problem.

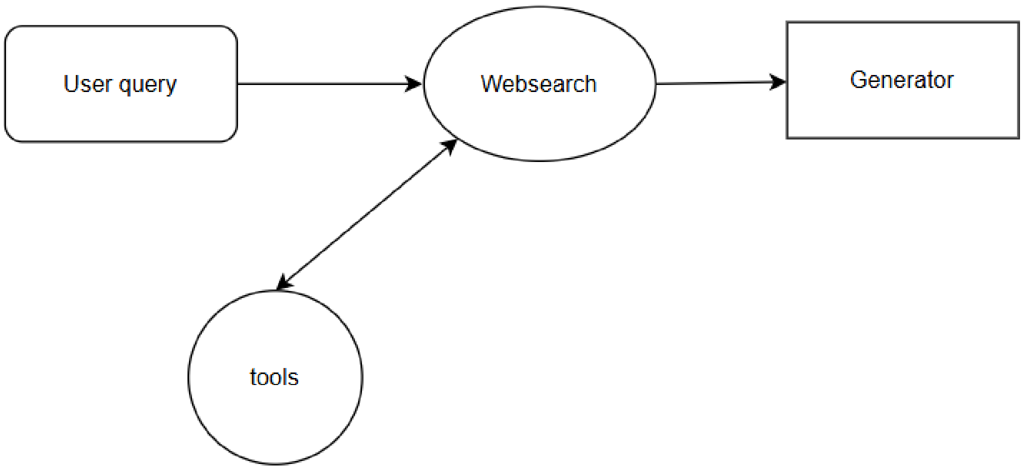


Figure 15. Architecture of web searching block

Web searching retrieval components:

- Web searching retriever: this module sends query to external web search engines. It retrieves relevant documents.
- Generating node: after retrieving documents, it passes query and fetches all retrieved document to the gemma-2-9b-it model to generate response.

4.5 Optimizing the system

To be able enhance the accuracy and relevance of retrieved documents, the system incorporates multiple optimization techniques. These methods ensure that user queries are effectively processed. The key optimization strategies include: query enhancement, chunking improvement and search optimization.

4.5.1 Optimizing RAG retrieval

Query enhancement improves the effectiveness of document retrieval by refining user queries before executing search. This involves two major components:

- Query router: the system first identifies which university the user refers to, it then routes the query to the appropriate document collection that contains relevant data about that university. This prevents unnecessary retrieval from unrelated collections where the data of universities stored are, making the retrieval process faster and more accurate.
- Query transformation: if the retrieved documents are not relevant or incomplete, the system analyzes the query complexity and determines the best transformation approach. It can either: decompose complex queries (break a long or multi-part query into simpler sub-queries that can be easier to process and retrieve information) or enhance the query (reformulate the query using synonyms, contextual expansion, or additional filtering criteria to improve retrieval performance

4.5.2 Optimizing Chunking

Chunking is the process of dividing large documents into smaller, manageable text segments for efficient retrieval, indexing, and searching in RAG and vector databases. Proper chunking ensures that information is retrieved accurately and preserves meaningful context, which is essential for semantic search and LLMs

Semantic chunking is an optimized form of chunking that preserves the meaning and context of documents by splitting it based on logical and topic-based structures, rather than arbitrary fixed lengths. This approach improves search accuracy and retrieval performance in vector database

4.5.3 Optimizing Searching

Vector databases are designed to **store, index, and retrieve high-dimensional vector embeddings** efficiently. Unlike traditional databases that use exact matches for querying, vector databases perform **similarity searches**, helping find the most relevant data points based on their semantic meaning.

Hybrid search: the system utilizes a hybrid search approach, combining lexical search (keyword-based retrieval) and semantic search (vector-based retrieval). Lexical search matches keywords in the query to documents using traditional search techniques: BM25. Semantic search uses embeddings to find documents that are conceptually related to the query, even if exact keywords do not match

Re-ranking: Once documents are retrieved, the system employs a re-ranking mechanism to prioritize the most relevance ones. This involves using cross-encoder to score the retrieved documents, pushing the most useful documents to the top.

4.6 System architecture

This report presents a comparison of two versions of my agent system, outlining the differences in architecture, tools, and functionality, as well as the improvements made based on testing and experimentation.

Version 1 utilizes an embedding technique through recursive splitting and vector search for efficient querying. The system interacts with an agent that is equipped with several tools, including:

- Time: A tool that simply provides the current time in Vietnamese.
- getRetrieval: Inspired by self-RAG, this tool fetches retrieval-based information.
- Websearch: A tool to perform web searches for additional information.
- getScore: A tool that converts queries into SQL and returns the relevant results based on scoring.

After thorough testing and experimentation, Version 2 emerged as the final version. It incorporates significant improvements, including:

- Semantic Chunking: A more refined method of breaking down information to enhance the understanding and relevance of data.
- Hybrid Search: An advanced search technique combining multiple search strategies for better results.

The tools in Version 2 have been streamlined and modified as follows:

- Time: Retained from Version 1 for providing the current time in Vietnamese.
- Corrective_RAG: A new tool designed to guide queries through three channels:
- Retrieval: For fetching relevant information.
- SQL: For structured data querying.
- Websearch: For gathering additional information from the web.
- The Corrective_RAG tool utilizes a router to direct the query appropriately to the relevant channel, ensuring more efficient and accurate results.

Inclusion, Version 2 after rigorous testing, proves to be a more advanced and efficient system, with improvements in semantic chunking, search accuracy, and streamlined tool usage. These enhancements allow for a more responsive and capable agent.

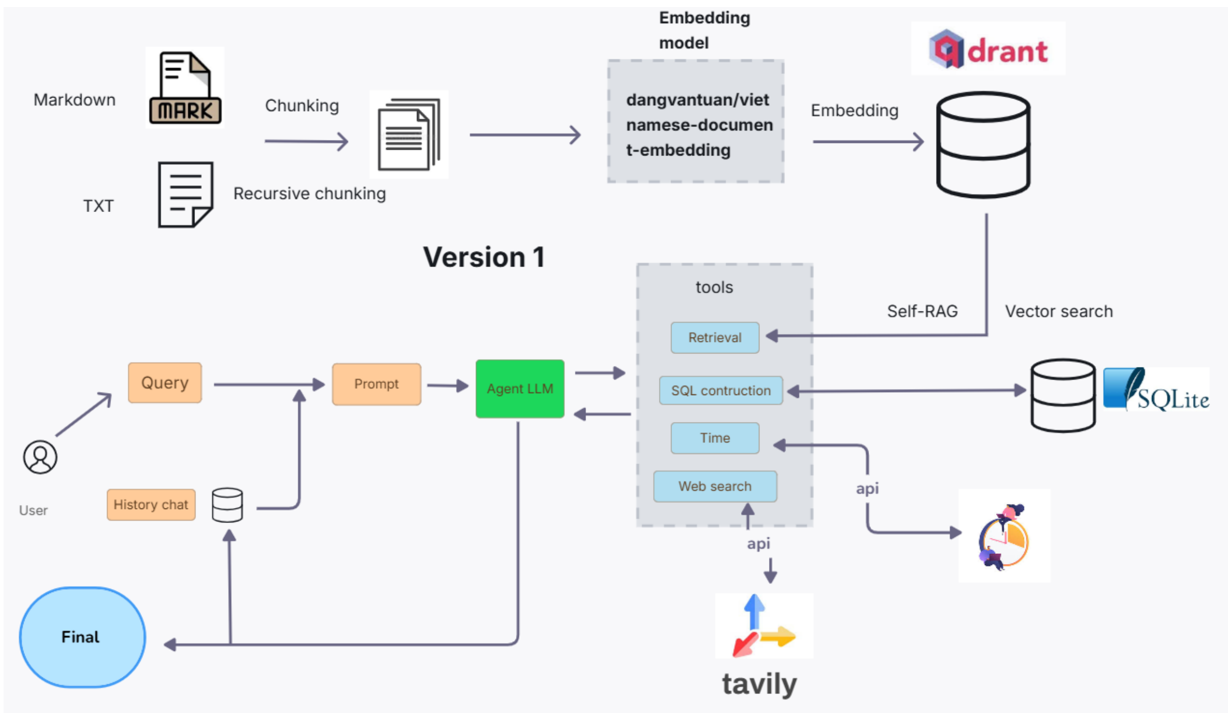


Figure 16 Version 1

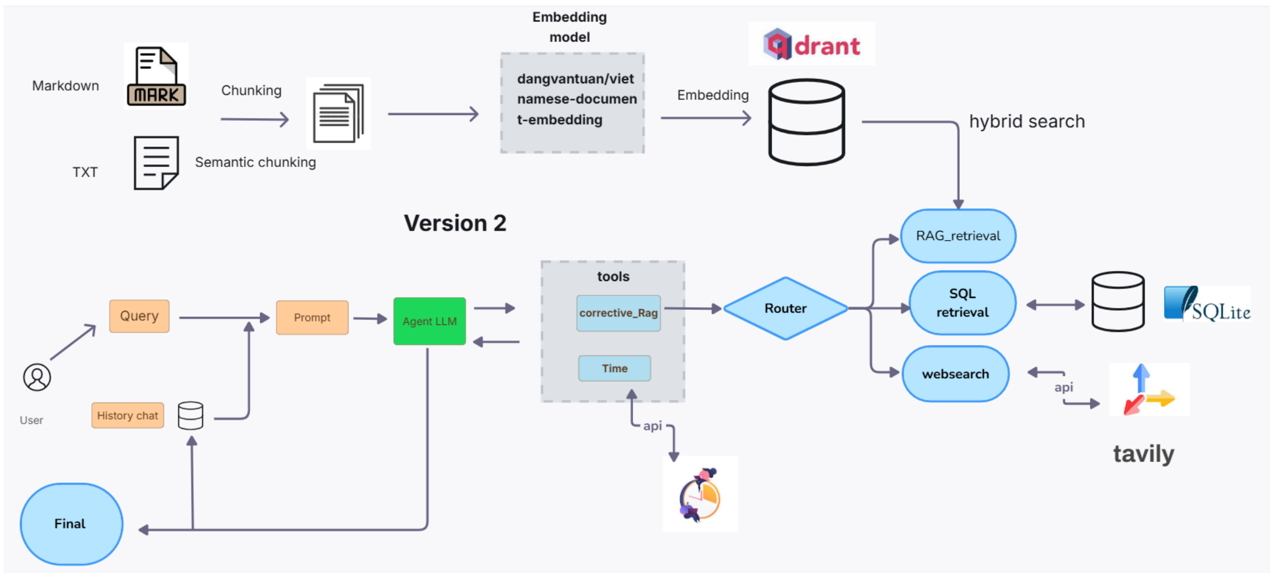


Figure 17 Version 2

4.7 Deploy

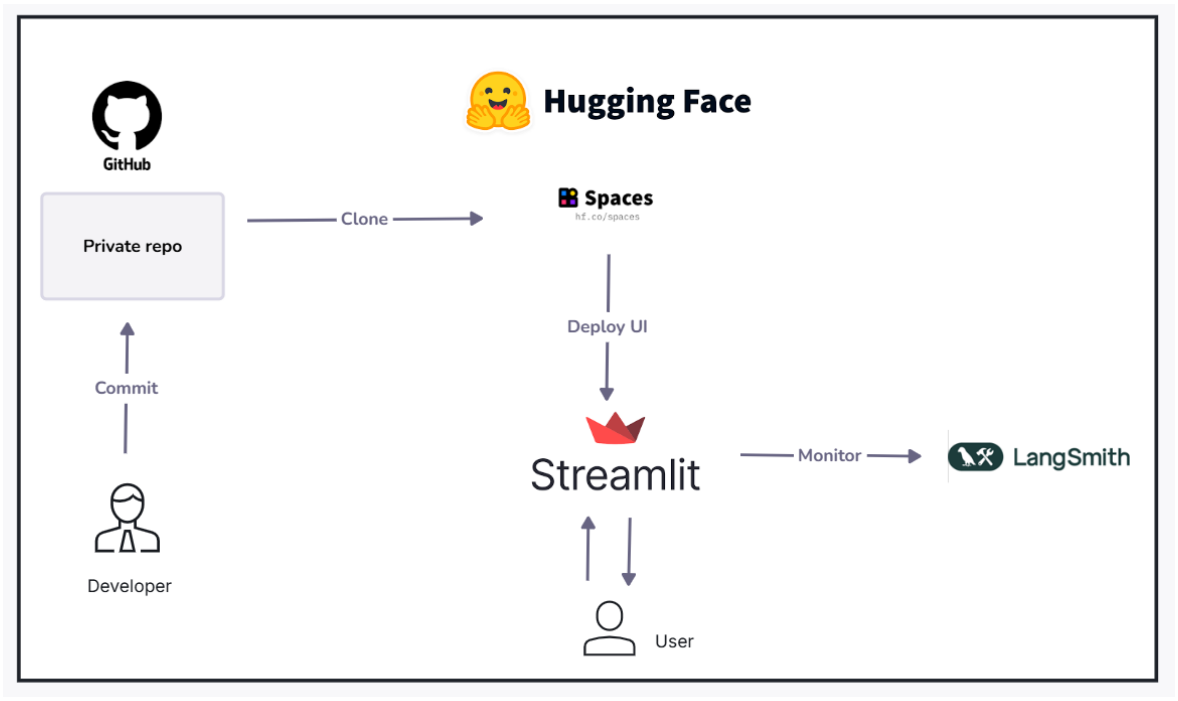


Figure 18 Architecture Deploy

The project was implemented using a combination of powerful tools and platforms, including GitHub, Hugging Face Spaces, Streamlit, and LangSmith. This process was designed to ensure the security, efficiency, and visibility of the application.

Key implementation steps:

- Source code development and hosting: The project's source code was developed and managed on GitHub, a popular source code hosting and management platform. The use of a private repository ensures that only authorized people can access and modify the source code.
- Application deployment: The "Admissions" application was deployed on Hugging Face Spaces, a specialized platform for deploying Machine Learning applications. Streamlit was used to build an interactive user interface for the application, making it easy for users to interact and use the project's functions.

- Monitoring and Evaluation: LangSmith is integrated into the process to monitor and evaluate the performance of the application. This allows the development team to track key metrics, identify potential issues, and improve the quality of the application over time.

Process Advantages:

- Security: Using a private repository on GitHub ensures the security of the source code.
- Efficiency: Hugging Face Spaces and Streamlit simplify the process of deploying and building user interfaces, increasing development speed.
- Monitoring Capabilities: LangSmith enables continuous monitoring of application performance, helping to detect and fix problems promptly.

CHAPTER 5. EXPERIMENT

5.1 Data

In this experiment, we used two different data chunking approaches to compare their usefulness in information retrieval: recursive split and semantic chunking. The goal was to assess the efficiency and accuracy of different approaches in dealing with huge datasets, notably their capacity to retain the semantic integrity of the data while optimizing search and retrieval tasks.

5.1.1 Data Qdrant

The data for vector search includes information from 15 universities in Ho Chi Minh City:

- Ho Chi Minh City University of Technology
- Ho Chi Minh City University of Technology and Education
- FPT University
- Foreign Trade University - Campus 2
- University of Science, Ho Chi Minh City
- Ton Duc Thang University
- University of Economics Ho Chi Minh City
- University of Finance and Marketing
- University of Information Technology, Ho Chi Minh City
- Van Lang University
- University of Medicine and Pharmacy, Ho Chi Minh City
- Pham Ngoc Thach University of Medicine
- Ho Chi Minh City Open University
- Nguyen Tat Thanh University
- Ho Chi Minh City University of Education

5.1.2 Data SQL

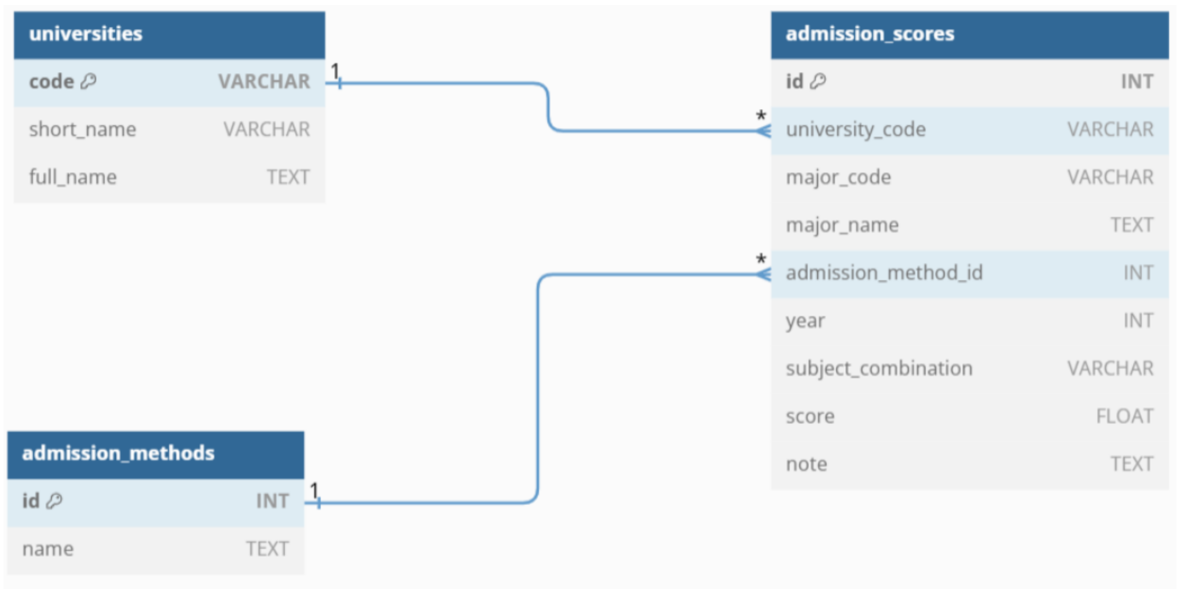


Figure 19 SQL relational diagram

The SQL data consists of three main tables, each serving a distinct purpose:

Universities Table:

- This table contains a list of 15 universities in Ho Chi Minh City. Each entry in this table represents one university. The universities include names like "Ho Chi Minh City University of Technology" and "FPT University."

Admission Methods Table:

- This table outlines the various admission methods used by the universities. The methods include:
    - National High School Graduation Exam
    - High School Transcript - First Semester Results
    - High School Transcript - Second Semester Results
    - General Knowledge Test
    - University Entrance Test - First Round
    - University Entrance Test - Second Round
    - Separate Examination

- Combined Admission Score
- International Certification
- VSAT Entrance Evaluation

Admission Scores Table:

- This table stores the required scores for each admission method at each university. It defines the minimum scores needed to qualify for each method, helping students understand the criteria for entry.

5.1.3 Data Ground Truth

The **ground truth data** is used for testing retrieval accuracy.

When assessing an information retrieval system's performance and accuracy, ground truth data is essential. This experiment uses two primary kinds of ground truth data, each with a unique structure and use case:

Standard Ground Truth Data

The main purpose of this kind of ground truth data is to evaluate the system's retrieval accuracy. It is made up of the following parts:

- id_question: A unique identifier assigned to each question to differentiate it from other queries in the dataset.
- question: The actual query or question being tested. This represents the information retrieval request that the system needs to address.
- id_corpus: A unique identifier that links the question to the relevant document or corpus in the database, ensuring that the question is mapped to the correct data source for retrieval.

LLM and RAGAS-Based Ground Truth Data

The second type of ground truth data is generated through Large Language Models (LLM) and RAGAS (retrieval-based system). This dataset has the following structure:

- question: The query that is being tested, similar to the first type of ground truth data.
- contexts: The relevant information extracted from the corpus that supports answering the question. These are the key pieces of information used to generate accurate responses.
- answer: The answer generated based on the context provided. This is the system’s response to the question.
- ground_truth: The correct or expected answer, used as the benchmark for evaluating the accuracy and quality of the answer generated by the system.

This type of ground truth data is generated using LLMs and RAGAS, and it is particularly useful for evaluating the system's performance in generating relevant and accurate answers.

5.2 Result

In this experiment, we compare the performance of two information retrieval methods, Navie RAG and Hybrid Search, when using the chunking methodology with recursive splitting. The two models are evaluated on prominent assessment measures such as Precision@k (P@k), Recall@k, Mean Reciprocal Rank (MRR@k), Discounted Cumulative Gain (DCG@k), and Normalized DCG (NDCG@k), with k = 3 and k = 5, respectively.

The experiment's purpose is to establish which strategy is more effective at finding and sorting relevant results. The comparison results will assist in determining the benefits and drawbacks of each strategy, as well as suggesting areas for development to optimize the search model.

5.2.1 Retrieval Testing with Recursive Chunking

This experiment tests retrieval performance on a database with recursive chunking. Two retrieval models are tested: Vector Search and Hybrid Search, with performance measures calculated at k = 3 and k = 5. Precision@k, Recall@k, MRR@k, DCG@k, and NDCG@k are the primary metrics evaluated.

Table 1 Evaluate retrieval by Recursive chunking with k =3

Model	Precision@3	Recall@3	MRR@3	DCG@3	NDCG@3
Vector search	0.106	0.317	0.229	0.251	0.251
Hybrid search	0.108	0.325	0.242	0.263	0.263

Table 2 Evaluate retrieval by Recursive chunking with k =5

Model	Precision@5	Recall@5	MRR@5	DCG@5	NDCG@5
Vector search	0.085	0.423	0.254	0.296	0.296
Hybrid search	0.081	0.405	0.262	0.298	0.298

Observations Overall, Hybrid Search demonstrates a slight advantage:

- At k = 3, Hybrid Search exhibits a minor enhancement across all metrics when compared to Vector Search, achieving higher Recall (0.325 vs. 0.317), MRR (0.242 vs. 0.229), and NDCG (0.263 vs. 0.251). At k = 5, Hybrid Search continues to lead with a better MRR (0.262 vs. 0.254) and NDCG (0.298 vs. 0.296), although its Precision (0.081 vs. 0.085) is slightly lower than that of Vector Search.
- Vector Search slightly outperforms in Precision@5: The Precision@5 for Vector Search (0.085) is marginally superior to that of Hybrid Search (0.081), suggesting that vector-based retrieval can occasionally yield more consistently relevant results among the top-ranked outcomes.

- Recall is improved at k = 5 for both approaches: As anticipated, Recall@5 is considerably greater than Recall@3 for both methods, confirming that retrieving a larger number of results enhances the likelihood of identifying relevant documents.

Conclusion Hybrid Search offers a more well-rounded performance, excelling in recall and ranking quality (MRR, DCG, NDCG), along with overall relevance. However, Vector Search retains a slight edge in precision at k = 5, which may be advantageous in situations where obtaining fewer but highly pertinent results is prioritized. Additional optimizations, such as fine-tuning hybrid weighting, modifying chunking strategies, or employing reranking models, could boost retrieval effectiveness.

5.2.2 Retrieval Testing with Semantic Chunking

This experiment assesses the retrieval performance of Semantic Chunking for database retrieval. Two retrieval algorithms are evaluated: vector search and hybrid search, with key performance metrics measured at k = 3 and k = 5. The purpose is to see if semantic chunking enhances retrieval quality over other chunking algorithms.

Table 3 Evaluate retrieval by Semantic chunking k = 3

Model	Precision@3	Recall@3	MRR@3	DCG@3	NDCG@3
Vector search	0.152	0.457	0.337	0.368	0.368
Hybrid search	0.157	0.471	0.325	0.362	0.362

Table 4 Evaluate retrieval by Semantic chunking k = 5

Model	Precision@5	Recall@5	MRR@5	DCG@5	NDCG@5
Vector search	0.114	0.571	0.363	0.415	0.415
Hybrid search	0.117	0.587	0.351	0.41	0.41

Observations There is a noticeable improvement when compared to recursive chunking:

- Both Vector Search and Hybrid Search demonstrate enhanced precision, recall, and ranking metrics in contrast to the results from Recursive Chunking (5.2.1). Precision@3 rose from 0.106 (recursive) to 0.152 (semantic) for Vector Search, while it increased from 0.108 to 0.157 for Hybrid Search. Recall@3 experienced a substantial rise, moving from 0.317 to 0.457 (Vector) and from 0.325 to 0.471 (Hybrid).
- Hybrid Search marginally surpasses Vector Search in recall but falls short in ranking quality: Hybrid Search exhibits superior Recall@3 (0.471 vs. 0.457) and Recall@5 (0.587 vs. 0.571), indicating it retrieves a greater number of relevant documents. Nevertheless, Vector Search achieves a higher Mean Reciprocal Rank (MRR) and DCG for both k = 3 and k = 5, which signifies it ranks the relevant results higher on average.
- The performance at k = 5 mirrors this trend: Recall@5 improves for both models, reinforcing the idea that retrieving more documents enhances the chance of obtaining relevant information. Precision@5 is slightly greater for Hybrid Search (0.117 vs. 0.114), but the contrast is minimal.

Semantic Chunking demonstrates significant enhancements in retrieval quality when compared to Recursive Chunking, exhibiting improved recall, precision, and ranking effectiveness. Hybrid Search continues to excel in recall, positioning it as an optimal option for maximizing the retrieval of relevant documents. Meanwhile, Vector Search outperforms in terms of ranking quality (MRR, DCG, NDCG), making it more adept at retrieving the most pertinent top results.

The findings indicate that Semantic Chunking is a more effective approach to retrieval and should be favored over Recursive Chunking whenever feasible. Future investigations could delve into the adjustments of hybrid weighting or reranking strategies to further improve retrieval efficacy.

5.2.3 RAGAS

To test RAGAS, we ran trials with two different versions: self-RAG and corrective-RAG. In addition, I used LangChain Smith to test the workflow and determine processing times.

About corrective-RAG:

The recalculated average values for the updated RAGAS metrics are as follows:

- Faithfulness: 0.834
- Answer Relevancy: 0.666
- Context Recall: 0.504
- Context Precision: 0.571
- Semantic Similarity: 0.886
- Answer Correctness: 0.565

The data above depicts the runtimes gathered by LangChain Smith over several measurements. Processing durations vary significantly, ranging from 7.66 to 55.31seconds.

Some runs are brief (about 10 seconds), while others last longer than 20 seconds, even up to 55 seconds. This could be influenced by the size of the data, the intricacy of the query, or the system's performance at the time of measurement. If optimization is required, try caching, query optimization, or enhancing the performance of the processing pipeline to reduce delays.

About self-RAG:

- Faithfulness: 0.7241
- Answer Relevancy: 0.7161
- Context Recall: 0.3113
- Context Precision: 0.4167
- Semantic Similarity: 0.8835
- Answer Correctness: 0.4014

The data shows the execution time of the Self-RAG model over numerous runs. We can see the following:

- Execution times vary significantly: most runs last between 5.64 and 11.84 seconds, but some extreme examples take 88.55 and 88.64 seconds.
- High inconsistency: Some searches run rapidly (<7 seconds), while others take nearly 90 seconds, showing considerable swings in processing time.

One major disadvantage of this strategy is that certain sophisticated queries fail to perform efficiently, resulting in high execution times. This indicates that Self-RAG may struggle with more complex queries, necessitating additional optimization to improve its stability and speed.

CHAPTER 6. INCLUSION

Version 2, Solves a hard problem which at least has productive steps towards obtaining admissions data for Vietnamese student correctly and in time. The state of university admissions in Vietnam at present can be described using 2 words: messy trend, messy that probably students cannot find and pick correct information amidst scattered, inconsistent with usually updated information. Version 2, on top of Retrieval-Augmented Generation (RAG) architecture, has blown your mind with better results for retrieving factual, context-aware and recent admissions information.

Analysis results of rigorous testing reveal version 2 to be significantly outperformed with the previous version, specifically in semantic chunking, search accuracy and tool usage optimization. The implementation of corrective RAG has fixed most of the RAG issues by making a feedback system responsive and efficient over time. Quantitatively, the results are as follows:

Faithfulness (0.834): The informative ability to faithfully report truth to source. Answer Relevance: 0.666 — Chatbot's answer is related to the user question. Context Recall: 0.504 — Demonstrates the inference results from reference contexts. Context Recall: 0.571 —How good is in recovery the information in context Semantic Similarity: 0.886 — Indicates how similar in meaning the question is to the answer Answer Ambiguity: 0.565 — Rates the integrity of the data offered.

Results are promising, although there are still weaknesses, particularly recall and contextual precision — but overall provide evidence of great potential for chatbots supporting students easy and responsive accuracy to admissions information. It is not just an information search tool, but also a useful support tool that will help students reduce their anxiety and uncertainty when making crucial decisions about future education.

6.1 Development orientation

Limitation

Although version 2 has achieved positive results, the project still faces several limitations that need to be addressed in the future:

- Limited Data: Currently, the chatbot is trained with data from only 15 universities in Ho Chi Minh City. This restricts its ability to provide comprehensive and accurate information for universities across the entire country.
- Processing Time: The chatbot's response time remains relatively long, especially when handling multiple simultaneous requests. This can negatively impact the user experience.
- Scalability: As it relies on free APIs, the chatbot's capacity to process requests is limited. With an increasing number of users, it may struggle to respond promptly.
- Testing Data: The current testing data is insufficient and does not cover all potential scenarios. This lack of comprehensive testing may result in the chatbot not being fully validated or robust.
- Accuracy: While the testing results are promising, there are still instances where the chatbot provides inaccurate or irrelevant information.

To address the restrictions listed above, the project team offers the following upgrade directions:

- Expand the data collection by gathering information from other universities, including those in other provinces and cities, so that the chatbot can provide comprehensive and reliable information for all schools.
- Optimize the system. Use more sophisticated APIs, optimize algorithms and databases, and increase the chatbot's processing speed and responsiveness.
- Improve scalability: Switch to commercial APIs or more powerful solutions to boost the chatbot's request processing capacity and ensure quick responses to a big number of users.

- Enhance test data: Collect more test data, including special cases, to thoroughly evaluate the chatbot and ensure optimal performance in all scenarios.

In the future, the project team will continue to research and develop the chatbot, focusing on improving accuracy and contextual processing capabilities, as well as expanding the scope of support, integrating additional features such as career counseling, online application support, etc. We believe that the university admissions consulting chatbot will be increasingly improved and contribute positively to the development of Vietnam's education industry. We sincerely thank all project members for their contributions and the interest and support of the community.

REFERENCES

[1] Wang, H., Zhang, R., Tao, M., & Liu, Y. (2023). *Retriever-Augmented Generation for Knowledge-Intensive NLP Tasks: A Survey*. arXiv. https://arxiv.org/pdf/2307.06435

[2] Ram, O., Shreter, U., Shoham, N., & Levy, O. (2023). *Corrective RAG: Intervention-Based Retrieval for Mitigating Hallucination in LLMs*. arXiv. https://arxiv.org/pdf/2303.18223

[3] Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). *Corrective Retrieval Augmented Generation*. *arXiv*. https://arxiv.org/abs/2401.15884

[4] Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., & Cui, B. (2024). *Retrieval-Augmented Generation for AI-Generated Content: A Survey*. *arXiv*. https://arxiv.org/abs/2402.19473

[5] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. *arXiv*. https://arxiv.org/abs/2312.10997

[6] Ye, Q., Axmed, M., Pryzant, R., & Khani, F. (2023). *Prompt Engineering a Prompt Engineer*. *arXiv*. https://arxiv.org/abs/2311.05661

[7] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). RAGAS: Automated Evaluation of Retrieval Augmented Generation. arXiv. https://arxiv.org/abs/2309.15217

[8] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2023). *SELF-RAG: Learning to retrieve, generate, and critique through self-reflection* [Preprint]. arXiv. https://arxiv.org/abs/2310.11511