VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**BUI HAI DUONG - 521H0220**

**TRAN QUANG LUAN - 52100254**

**BUILDING A CHATBOT TO ADVISE ON ROAD TRAFFIC LAWS USING RAG TECHNIQUES**

**INFORMATION TECHNOLOGY PROJECT**

**COMPUTER SCIENCE**

**HO CHI MINH, 2025**

VIETNAM GENERAL CONFEDERATION OF LABOR

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**BUI HAI DUONG – 521H0220**

**TRAN QUANG LUAN - 52100254**

**BUILDING A CHATBOT TO ADVISE ON ROAD TRAFFIC LAWS USING RAG TECHNIQUES**

**INFORMATION TECHNOLOGY PROJECT**

**COMPUTER SCIENCE**

Instructor

**Assoc. Prof. PhD. Le Anh Cuong**

**HO CHI MINH, 2025**

ACKNOWLEDGEMENT

To complete this essay, besides our own efforts, we have received a lot of help in terms of knowledge, experience, and skills from the school and teachers. First and foremost, we would like to express my special gratitude to Assoc. Prof. PhD. Le Anh Cuong - the lecturer of Information Technology Project course who has taught us valuable knowledge of the subject. That knowledge is the foundation for us to continue learning and effectively apply it to this essay. Additionally, We would like to thank the teacher for allowing us to complete this essay, which has helped us to further develop my understanding of the subject. Thank you for your guidance and support in helping us to complete this essay to the best of my ability. Moreover, we would also like to express our gratitude to the school and the teachers who have compiled the Information Technology Project materials, providing me with useful resources for research and essay writing. Thank you sincerely!

*Ho Chi Minh City, January 10, 2025*

*Author*

*(signature and full name)*

**COMPLETION OF THESIS**

**AT TON DUC THANG UNIVERSITY**

We here by certify that this thesis is my/our own work and was conducted under the guidance of PhD. Pham Van Huy. The research and results presented in this thesis are truthful and have not been published previously in any form. The data presented in tables and figures used for analysis, comments, and evaluations were collected by the author from various sources and are clearly cited in the reference section.

Moreover, this thesis includes some comments, evaluations, and data from other authors and organizations, which are properly cited and referenced.

**If any misconduct is detected, I fully take responsibility for the content of my thesis**. Ton Duc Thang University is not liable for any copyright infringement that may occur during the thesis completion process.

*Ho Chi Minh City, January 10, 2025*

*Author*

*(signature and full name)*

**BUILDING A CHATBOT TO ADVISE ON ROAD TRAFFIC LAWS USING RAG TECHNIQUES**

**ABSTRACT**

This report introduces a Retrieval-Augmented Generation (RAG) system designed for traffic law consultation, integrating advanced techniques to enhance retrieval and reasoning capabilities. The system employs Parent Document Retrieval to maintain contextual integrity by retrieving relevant legal texts while preserving their structural coherence. To improve response accuracy, we leverage Few-shot Chain-of-Thought (CoT) prompting, enabling the model to perform multi-step reasoning over legal queries. Additionally, we implement Query Transformation to rephrase user inputs for better retrieval coverage and Query Routing to dynamically select the most suitable retrieval strategy. Our evaluation demonstrates that these enhancements significantly improve faithfulness, relevance, and reasoning depth, making the system a reliable AI assistant for traffic law interpretation and consultation.

**TABLE OF CONTENTS**

[TABLE OF FIGURES vi](#_Toc190268476)

[LIST OF TABLES vii](#_Toc190268477)

[LIST OF ABBREVIATION viii](#_Toc190268478)

[CHAPTER 1. INTRODUCTION AND TOPIC OVERVIEW 1](#_Toc190268479)

[1.1 Reason for the topic 1](#_Toc190268480)

[1.2 Objectives of implementing this topic 1](#_Toc190268481)

[CHAPTER 2. THEORETICAL BASIS TO RAG 2](#_Toc190268482)

[2.1 What is RAG 2](#_Toc190268483)

[2.1.1 Introduction to RAG 2](#_Toc190268484)

[2.1.2 Why is RAG 2](#_Toc190268485)

[2.2 Components in RAG 3](#_Toc190268486)

[2.2.1 Retrieval Module 3](#_Toc190268487)

[2.2.2 Generative Module 7](#_Toc190268488)

[2.3 Advanced techniques 8](#_Toc190268489)

[2.3.1 Query transformation 8](#_Toc190268490)

[2.3.2 Query routing 8](#_Toc190268491)

[2.3.3 Advanced retrieval strategies 9](#_Toc190268492)

[2.3.4 Reranking 12](#_Toc190268493)

[2.3.5 Prompting 13](#_Toc190268494)

[CHAPTER 3. PROPOSED METHOD 15](#_Toc190268495)

[3.1 Approaching way 15](#_Toc190268496)

[3.1.1 What is Advanced RAG 15](#_Toc190268497)

[3.1.2 Why using Advanced RAG 15](#_Toc190268498)

[3.2 Generative and embedding model 15](#_Toc190268499)

[3.2.1 Embedding model 15](#_Toc190268500)

[3.2.2 Generative model 18](#_Toc190268501)

[3.3 Components of RAG system 18](#_Toc190268502)

[3.3.1 Advanced techniques used 19](#_Toc190268503)

[3.3.2 Retrieval module 19](#_Toc190268504)

[3.3.3 Generative module 20](#_Toc190268505)

[CHAPTER 4. EXPERIMENTAL SETUP 22](#_Toc190268506)

[4.1 Data usage 22](#_Toc190268507)

[4.2 Vectorstore and retrieval system 22](#_Toc190268508)

[4.3 Large language model 23](#_Toc190268509)

[4.3.1 Query transform 23](#_Toc190268510)

[4.3.2 Query routing 23](#_Toc190268511)

[4.3.3 Reponse 23](#_Toc190268512)

[CHAPTER 5. EXPERIMENTAL RESULTS 25](#_Toc190268513)

[5.1 Test dataset 25](#_Toc190268514)

[5.2 Evaluate with RAGAS 25](#_Toc190268515)

[5.2.1 What is RAGAS? 25](#_Toc190268516)

[5.2.2 Metrics for evaluating 26](#_Toc190268517)

[5.2.3 Evaluating result with RAGAS 26](#_Toc190268518)

[REFERENCES 28](#_Toc190268519)

# TABLE OF FIGURES

[Figure 2.1. Workfow of RAG 2](#_Toc190268463)

[Figure 2.2. Workflow of Storing content 3](#_Toc190268464)

[Figure 2.3. Workflow of retrieval 4](#_Toc190268465)

[Figure 2.4. Workflow of Parent Document Retrieval 9](#_Toc190268466)

[Figure 2.5. Workflow of Hybrid retrieval 11](#_Toc190268467)

[Figure 2.6. Workflow of Multi-query retrieval 12](#_Toc190268468)

# LIST OF TABLES

[Table 3.1. Embedding model evaluation based on Accuracy 16](#_Toc190268452)

[Table 3.2. Embedding model evaluation based on Precision 17](#_Toc190268453)

[Table 3.3. Embedding model evaluation based on Recall 17](#_Toc190268454)

[Table 5.1. RAGAS evaluation 26](#_Toc190268455)

# LIST OF ABBREVIATION

|  |  |
| --- | --- |
| RAG | Retrieval Augmented Generation |
| LLM | Grammatical Error Correction |
| FAISS | Facebook AI Similarity Search |
| HNSW | Hierarchical Navigable Small World |
| CoT | Chain of Thought |

# INTRODUCTION AND TOPIC OVERVIEW

## Reason for the topic

In 2025, traffic laws have undergone significant changes, greatly impacting people's daily lives, especially with increased penalties as stipulated in Decree 168, which took effect on January 1, 2025. Although road traffic is an essential part of everyday life, it is challenging for citizens to fully grasp all traffic regulations due to the large number of laws, circulars, and decrees issued. Not everyone has the time and effort to study and absorb all these legal documents. Therefore, developing and implementing a chatbot for road traffic laws is an effective solution, making legal information more accessible while saving time and effort. This helps improve people's knowledge and awareness when participating in traffic.

## Objectives of implementing this topic

The objective of this study is to research, design, and develop an intelligent chatbot capable of automatically answering questions related to traffic laws while also assisting users in resolving specific situations they may encounter while participating in traffic. This chatbot not only helps improve awareness of traffic regulations but also provides quick, accurate, and easy-to-understand information, supporting users in complying with the rules and minimizing traffic violations.

# THEORETICAL BASIS TO RAG

## What is RAG

### Introduction to RAG

Retrieval Augmented Generation (RAG) is a method that combines data retrieval from an external database with large language models. This approach enables injecting relevant data into the language model for question-answering tasks without requiring training or fine-tuning.

The operation of RAG differs from fine-tuning a model. In terms of functionality, RAG feeds a Large Language Model (LLM) with both the user's query and relevant retrieved documents, enabling the LLM to generate a response based on the provided context. This approach enhances the accuracy and relevance of responses without requiring extensive retraining of the model.

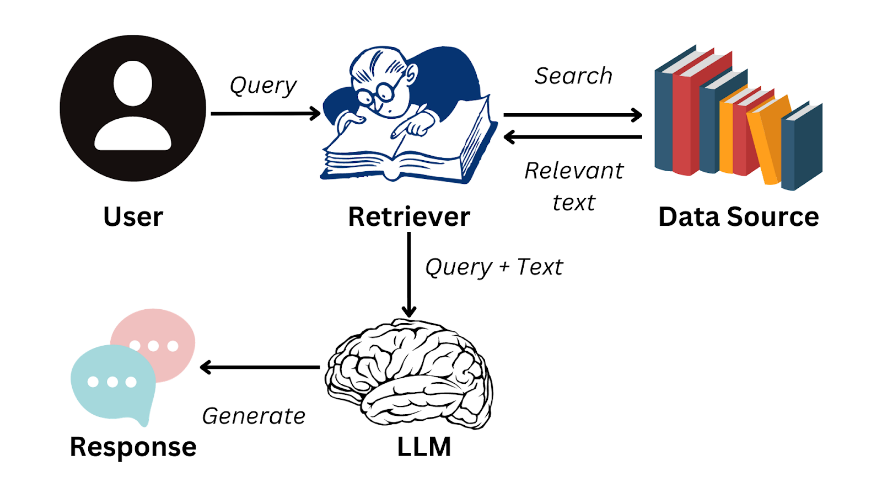


Figure 2.1. Workfow of RAG

The image above illustrates the workflow of RAG. First, our data is stored in one or more databases. When a user submits a query, the system searches for relevant documents within the stored database. The retrieved documents, along with the user's query, are then fed directly into the model, enabling it to generate a response based on the retrieved information. This approach ensures that the model provides accurate and up-to-date answers without requiring extensive fine-tuning.

### Why is RAG

RAG is an efficient alternative to fine-tuning, as it significantly reduces the time and cost associated with training or fine-tuning a model. Moreover, updating new data in an external database is far more convenient than repeatedly fine-tuning the model. This makes RAG a powerful solution designed to overcome the limitations of fine-tuning while ensuring models stay up-to-date with the latest information. This way of approaching also covers the hallucination of LLM in question answering task.

## Components in RAG

Typically, in a RAG system, there are two main components: the Retrieval Module and the Generation Module. The Retrieval Module is responsible for retrieving documents that are most relevant to the user’s query, while the Generation Module uses the retrieved documents to generate a response based on the user’s question and the previously retrieved texts. This RAG system is enhanced with advanced techniques to optimize the results provided to the user.

### Retrieval Module

The Retrieval module plays a crucial role in identifying and extracting relevant information based on the user's query. The retrieved data is then seamlessly integrated into the Generative module, enhancing its ability to generate more accurate and contextually relevant responses.

The Retrieval Module operates through two main workflows:

* The first workflow involves processing and embedding data before storing it in the vector database, ensuring efficient retrieval.

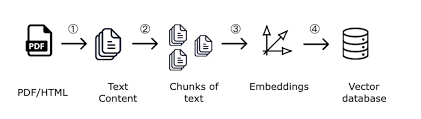


Figure 2.2. Workflow of Storing content

* The second workflow takes the user's query, converts it into an embedding, and searches the vector database to retrieve the most relevant chunks.

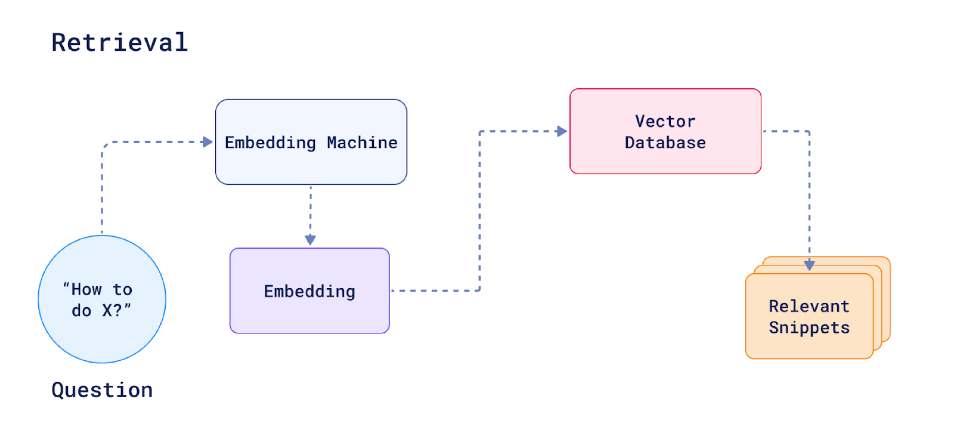


Figure 2.3. Workflow of retrieval

#### Chuking

The input of the chunking process consists of documents collected through searches on websites, books, and reliable information sources. Data stored in file formats like PDF, DOCX, and Markdown will be chunked for easier storage and retrieval. The output of this process is the chunks created from the original texts.

There are various chunking methods depending on the problem and the type of text being processed. Many chunking methods are used in text for different purposes. Common chunking methods that are widely used include:

* Fixed-size chunking: Divides the text into segments of fixed size. Example:
  + A long article is divided into chunks of exactly 100 words each.
  + Text: "The cat sat on the mat. It was a sunny day outside. The cat liked to play in the garden."
  + Chunk 1: "The cat sat on the mat. It was a sunny day outside. The cat liked to play"
  + Chunk 2: "in the garden."
* Recursive chunking: Splits the text according to segmenting markers (paragraphs, headings, etc.) recursively until the desired size is achieved.

Example:

* + A document is split based on its structure, like paragraphs or headings, and further divided recursively if the segment size exceeds the desired limit.
  + Text:

Heading 1

"Introduction to Science"

"Science is the study of the natural world..."

* + Chunk 1: "Introduction to Science"
  + Chunk 2: "Science is the study of the natural world..."
  + If any chunk exceeds the size, it is further divided by sections or sentences.
* Document-based chunking: Used for documents with special structures (Markdown, source code, tables, etc.).

Example:

* + A Markdown document with various elements (headings, tables, code) is chunked based on its structure.
  + Text:

Heading: Usage Instructions

"print('Hello World')"

"This function prints a greeting message."

* + Chunk 1: Heading: Usage Instructions
  + Chunk 2: "print('Hello World')"
  + Chunk 3: "This function prints a greeting message."
* Semantic chunking: Groups sentences based on semantic similarity to cluster related sentences.
* Example:
  + Sentences are grouped based on their meaning, so related ideas are clustered together.
  + Text:

"The cat sat on the mat. The dog ran outside. The cat loves to nap on the mat."

"The dog enjoys running around in the yard."

* + Chunk 1: "The cat sat on the mat. The cat loves to nap on the mat."
  + Chunk 2: "The dog ran outside. The dog enjoys running around in the yard."
* Agentic chunking: Uses a LLM to decide how to group text segments based on context.

Example:

* + LLM determines how to group sentences based on the context and content.
  + Text:

"The weather is nice today. I think I'll go for a walk. The cat is sleeping."

"I wonder if I should bring an umbrella."

* + Chunk 1: "The weather is nice today. I think I'll go for a walk."
  + Chunk 2: "The cat is sleeping. I wonder if I should bring an umbrella."
  + (The LLM might group sentences based on the theme or context, e.g., outdoor activities vs. the cat’s actions.)

#### Storing chunks

Storing is the process of saving the chunks that have been segmented during the chunking process. Depending on the problem, we will divide the data into different storage spaces. To speed up the retrieval process, the chunks are stored in vector spaces (Vectorstore). This process includes the following parts:

* Embedding: Using an embedding model to convert the chunks of text into vectors for storage in the Vectorstore.
* Vectorstore: There are many Vectorstores today that support storing vectors in the cloud, such as FAISS, ChromaDB, Pinecone, etc. Additionally, local Vectorstores can also be used to store vectors, such as PostgresSQL, etc.

#### Retrieval

The retrieval process is a critical component in augmenting the generative model with external knowledge, ensuring responses remain both accurate and contextually relevant. This process operates by leveraging an embedding model to transform the user's query into a high-dimensional vector representation. These vectors capture semantic meaning, enabling efficient information retrieval beyond simple keyword matching.

Once the query vector is generated, it is compared against a pre-indexed vector database containing segmented knowledge chunks. The comparison is typically performed using similarity metrics such as cosine similarity, Euclidean distance, or dot product similarity, depending on the characteristics of the embedding space. Advanced retrieval methods may also incorporate approximate nearest neighbor (ANN) search algorithms, such as FAISS (Facebook AI Similarity Search) or HNSW (Hierarchical Navigable Small World), to optimize speed and scalability when handling large datasets.

### Generative Module

The Generative module is responsible for processing the user's query along with the retrieved results from the Retrieval module. It then generates a coherent and contextually appropriate response based on the provided information.

After retrieving the relevant chunks, the user’s question and the related chunks are fed into the LLM to generate an answer and respond to the user. The main steps in the generation module are:

* Creating the input prompt: Combine the user’s question with the retrieved chunks.
* Using the LLM to answer the question: Use models such as T5, ViT5, GPT, Gemini, etc.

## Advanced techniques

### Query transformation

This is a technique used to transform user queries into shorter, more concise questions, helping to optimize the process of retrieving chunks more effectively.

This technique is applied by using large language models to perform prompting techniques that convert user queries into more suitable questions. Some techniques for query transformation include:

* Step-back prompting: A method that transforms specific questions into broader, more general questions, improving the ability to query documents.
* Query rewriting: A method that makes initial queries more specific and detailed. This approach helps increase accuracy in the document retrieval process.
* Sub-query decomposition: A method that breaks down complex queries into simpler, component queries. This process simplifies and enhances the retrieval process.

### Query routing

Query routing is a technique used to direct queries to the appropriate databases or LLMs with specific functionalities based on the query's intent. By intelligently routing queries to the most relevant data sources or specialized models, this approach optimizes the output of LLMs, enhancing the accuracy and relevance of responses.

Additionally, query routing helps improve system efficiency by reducing unnecessary computations and ensuring that each query is processed by the most suitable resource. This is particularly useful in multi-agent AI systems or hybrid architectures, where different models or databases specialize in distinct tasks, such as retrieving factual knowledge, generating code, or performing analytical reasoning. By leveraging query routing, organizations can build more scalable, precise, and context-aware AI solutions.

### Advanced retrieval strategies

Typically, there are various types of documents, so the choice of retrieval techniques will depend on the nature of the documents.

#### Parent document retrieval

This involves breaking down original documents into smaller chunks. Each original document has its own ID, and the chunks will point to the original document when retrieved through the user's query. This technique is optimized for retrieving the entire context of documents, such as legal texts, medical documents, etc.

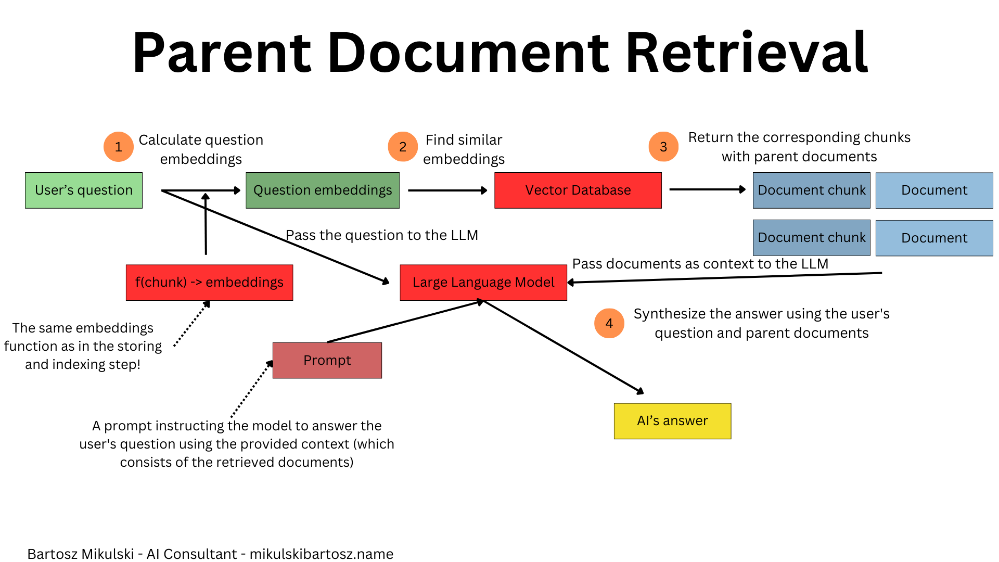


Figure 2.4. Workflow of Parent Document Retrieval

#### Hybrid retrieval

Hybrid retrieval is an advanced information retrieval technique that integrates keyword-based retrieval (sparse retrieval) and semantic-based retrieval (dense retrieval) to enhance the accuracy and effectiveness of retrieving relevant information. By leveraging both approaches, this method ensures that the retrieval system captures both explicit keyword matches and semantic similarities, leading to a more comprehensive and precise retrieval process.

Sparse Retrieval (Keyword-Based Retrieval): Sparse retrieval is a traditional approach that relies on lexical matching between the user’s query and indexed documents. It uses techniques such as BM25 and TF-IDF, which rank documents based on term frequency and inverse document frequency. This method is effective for retrieving exact keyword matches but often struggles with queries that require semantic understanding or involve synonyms and paraphrased expressions.

TF-IDF (Term Frequency - Inverse Document Frequency) is a statistical measure used to evaluate the importance of a term within a document relative to a collection of documents (corpus). It consists of two components: Term Frequency (TF) and Inverse Document Frequency (IDF). The general formula for TF-IDF is:

TF measures how often a term appears in a document . The most common formula is:

Where:

* is the number of times term appears in document (term frequency).
* is the total number of terms in document .

IDF measures how important or unique a term is across the entire corpus. It reduces the weight of frequently occurring words that are less informative (e.g., "the", "is") and increases the weight of rare terms. The standard formula is:

Where:

* is the total number of documents in the corpus.
* is the number of documents containing term .

Dense Retrieval (Semantic-Based Retrieval): Dense retrieval, in contrast, employs embedding models (e.g., BERT, Sentence-BERT, or ColBERT) to transform queries and documents into high-dimensional vector representations. These embeddings capture semantic relationships, allowing the retrieval system to identify relevant information even when there is no direct keyword overlap. The similarity between vectors is measured using cosine similarity, Euclidean distance, or dot product similarity to retrieve the most semantically relevant documents.

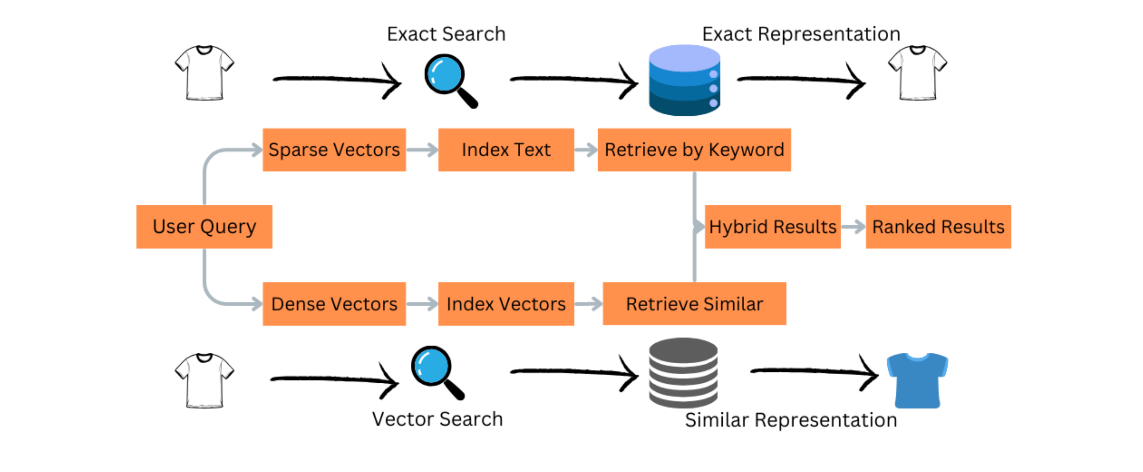


Figure 2.5. Workflow of Hybrid retrieval

#### Multi-query retrieval:

Multi-Query Retrieval (MQR) enhances retrieval by transforming a single query into multiple semantically similar variations, improving recall and robustness. It generates diverse query versions through synonyms, paraphrasing, and sub-query decomposition, then retrieves and merges results using sparse (BM25) and dense (embedding-based) retrieval. A reranking step ensures relevance and eliminates redundancy. This approach improves search accuracy, enhances RAG models, and is widely used in search engines, chatbots, academic research, and e-commerce to handle diverse query formulations effectively.

A diagram of a diagram

Description automatically generated

Figure 2.6. Workflow of Multi-query retrieval

### Reranking

Reranking is a method that utilizes a trained model to assess the similarity between text chunks and a query. It then ranks these chunks based on their relevance, arranging them in descending order of similarity. This approach is crucial because, in many cases, initial ranking or retrieval methods may not fully capture the context or relevance of the chunks. Reranking helps prioritize the most contextually relevant chunks, improving the accuracy and quality of the final result by ensuring that the most relevant information is highlighted.

For example, consider a search engine that retrieves several documents in response to a user query. Initially, the engine might rank the documents based on keywords or basic relevance. However, this initial ranking might not always highlight the most contextually relevant content.

Let's say the query is: "What are the benefits of using solar energy?" and the search engine retrieves the following chunks of text:

"Solar energy can reduce electricity bills."

"It is important to stay hydrated throughout the day."

"Solar panels are a renewable energy source that helps reduce carbon emissions."

"Drinking water is essential for maintaining good health."

In this case, chunk 2 and chunk 4 are irrelevant to the user's query. The reranking model would evaluate the similarity between the query and each chunk, recognizing that chunk 1 and chunk 3 are more relevant. It would then reorder the chunks, ranking chunk 3 first due to its relevance to the environmental benefits of solar energy, followed by chunk 1. This ensures that the most relevant information is presented to the user.

By reranking, the system improves the overall accuracy and relevance of the results, ensuring the user receives the most useful and contextually appropriate information.

### Prompting

#### Zero-shot prompting

Zero-shot prompting is a technique in which a LLM generates responses without receiving explicit instructions or examples. Instead, the model relies entirely on its pre-trained knowledge and reasoning capabilities to interpret and answer the query. This approach allows the LLM to generalize across various tasks but may lead to variability in response accuracy depending on the model's prior training data and contextual understanding.

For example:

* Prompt: "What is the capital of France?"
* LLM Response: "The capital of France is Paris."

#### Few-shot prompting

Few-shot prompting is a prompting technique where a LLM is provided with a few specific examples along with the query. These examples serve as a reference, guiding the model to generate responses that follow the demonstrated patterns. By leveraging these examples, the LLM can better understand the expected output format and improve response accuracy, especially in tasks requiring structured or domain-specific knowledge.

For example:

* Prompt: "Translate the following sentences from English to French:

'Hello, how are you?' → 'Bonjour, comment ça va ?'

'I love learning new languages.' → 'J'aime apprendre de nouvelles langues.'

'What is your name?' → ???"\*

* LLM Response: "Quel est ton nom ?"

#### Chain of thought

Chain of Thought (CoT) prompting is a technique that guides a LLM through a step-by-step reasoning process to solve a given task. By breaking down the problem into smaller, logical steps, this method helps the model follow a structured approach, improving its ability to generate accurate and well-reasoned responses. CoT prompting is particularly effective for complex tasks requiring multi-step reasoning, such as mathematical problem-solving, logical inference, and legal analysis.

For example:

* Prompt: "If a store sells apples at $2 each and oranges at $3 each, and a customer buys 3 apples and 2 oranges, how much does the customer spend? Let's think step by step."
* LLM Response:

The customer buys 3 apples, and each apple costs $2.

→ 3 × $2 = $6

The customer buys 2 oranges, and each orange costs $3.

→ 2 × $3 = $6

Total cost: $6 + $6 = $12

# PROPOSED METHOD

## Approaching way

### What is Advanced RAG

Advanced RAG (Retrieval-Augmented Generation) is a powerful approach that combines retrieval-based methods with generative models to enhance the quality of text generation tasks. Unlike traditional models that rely solely on pre-trained knowledge, Advanced RAG dynamically retrieves relevant information from external sources, such as databases or documents, in real time. This allows the model to generate more accurate, contextually rich, and up-to-date responses by incorporating external knowledge.

### Why using Advanced RAG

One of the key benefits of Advanced RAG is its ability to improve the quality of generated text by incorporating external context, helping the model to produce more informative and coherent answers. Unlike traditional models that rely solely on pre-trained knowledge, RAG systems can dynamically retrieve relevant information from external sources, ensuring that the generated responses are both relevant and factually accurate.

Furthermore, Advanced RAG systems can significantly reduce the dependency on extensive fine-tuning, making them more efficient and easier to deploy. They also help mitigate issues related to hallucinations (i.e., the generation of incorrect or fabricated information), as the retrieval step ensures that the generated content is grounded in actual data.

## Generative and embedding model

### Embedding model

The main function of the embedding model is to convert tokens such as images, videos, characters, or sounds into vector representations, which helps make data processing in machine learning easier.

In RAG, this embedding process occurs during the information retrieval step. Text chunks are extracted from the main documents, then embedded into vectors and stored in a database. When user queries are input, they are converted into vectors and compared with the vectors stored in the database to retrieve relevant data.

Below is a summary of the results of embedding models based on 20% of the Zalo law training set, which has not been trained on the models listed below. The results show evaluations of Accuracy, Precision, and Recall:

Table 3.1. Embedding model evaluation based on Accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy@1 | Accuracy@3 | Accuracy@5 | Accuracy@10 |
| vietnamese-bi-encoder | 0.8169 | 0.9108 | 0.9437 | 0.9640 |
| sup-SimCSE-VietNamese-phobert-base | 0.5540 | 0.7308 | 0.7981 | 0.8748 |
| halong\_embedding  (768) | 0.8294 | 0.9233 | 0.9437 | 0.9687 |
| halong\_embedding (512) | 0.8138 | 0.9233 | 0.9390 | 0.9703 |
| halong\_embedding (256) | 0.7934 | 0.8967 | 0.9280 | 0.9593 |
| halong\_embedding (128) | 0.7840 | 0.8951 | 0.9264 | 0.9515 |
| halong\_embedding (64) | 0.6980 | 0.8435 | 0.8920 | 0.9358 |

Table 3.2. Embedding model evaluation based on Precision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Precision@1 | Precision@3 | Precision@5 | Precision@10 |
| vietnamese-bi-encoder | 0.8169 | 0.3099 | 0.1931 | 0.0987 |
| sup-SimCSE-VietNamese-phobert-base | 0.5540 | 0.2473 | 0.1621 | 0.0892 |
| halong\_embedding  (768) | 0.8294 | 0.3146 | 0.1931 | 0.0991 |
| halong\_embedding (512) | 0.8138 | 0.3146 | 0.1922 | 0.0992 |
| halong\_embedding (256) | 0.7934 | 0.3062 | 0.1900 | 0.0981 |
| halong\_embedding (128) | 0.7840 | 0.3046 | 0.1894 | 0.0975 |
| halong\_embedding (64) | 0.6980 | 0.2864 | 0.1815 | 0.0954 |

Table 3.3. Embedding model evaluation based on Recall

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Recall@1 | Recall@3 | Recall@5 | Recall@10 |
| vietnamese-bi-encoder | 0.8020 | 0.9045 | 0.9390 | 0.9601 |
| sup-SimCSE-VietNamese-phobert-base | 0.5446 | 0.7246 | 0.7903 | 0.8693 |
| halong\_embedding  (768) | 0.8146 | 0.9178 | 0.9390 | 0.9640 |
| halong\_embedding (512) | 0.7989 | 0.9178 | 0.9343 | 0.9656 |
| halong\_embedding (256) | 0.7786 | 0.8920 | 0.9233 | 0.9546 |
| halong\_embedding (128) | 0.7707 | 0.8889 | 0.9210 | 0.9476 |
| halong\_embedding (64) | 0.6854 | 0.8365 | 0.8842 | 0.9311 |

Based on the statistics above, we can see that the embedding model yielding the highest result is halong\_embedding (768). Therefore, halong\_embedding (768) will be the chosen model for use as the embedding model.

### Generative model

Generative models are large language models (LLMs) capable of creating new content based on the data they have been trained on and user input. These models are trained on vast datasets that cover a wide range of topics to understand and generate natural language text.

In the context of the RAG system, popular generative models come from organizations like OpenAI, Google, Meta, as well as open-source models on GitHub or HuggingFace.

In particular, Google's Gemini model offers versions like Gemini 1.5 Flash and Gemini 1.5 Pro, which can handle long contexts, supporting context windows of up to 1 million and 2 million tokens, respectively. This allows for the creation of detailed and complex prompts.

To use these models, users need an API key provided through a Google account. The choice of model will depend on the specific requirements of the project, including task complexity, context limits, and budget.

## Components of RAG system

### Advanced techniques used

#### Query transformation

To optimize query inputs for better retrieval results, this approach transforms queries using various transformation techniques.

* If a query is too complex, step-back prompting is applied to break it down into simpler, more structured subqueries, making it easier to retrieve relevant information.
* If a query is too vague or lacks detail, query rewriting is used to enhance its clarity and specificity, ensuring that it provides enough context for accurate retrieval.

By dynamically adapting query transformations based on their characteristics, this method improves retrieval efficiency and enhances the overall quality of responses generated by the system.

#### Query routing

Query routing enables efficient navigation across multiple databases, ensuring that queries are directed to the most relevant sources. This is especially important in the legal domain, where different databases contain distinct legal documents, regulations, or case laws. By intelligently routing queries to the appropriate legal database, this method optimizes retrieval accuracy and enhances the relevance of retrieved information.

Additionally, if a query consists of general or non-legal questions, it is directly forwarded to the language model for processing instead of querying legal databases. This selective routing mechanism ensures both efficiency and precision, making it an essential component of this system.

### Retrieval module

#### Chunking method

Recursive Chunking is a technique that breaks down large documents into smaller, structured chunks while preserving essential context. This method optimizes information retrieval, making it particularly useful for RAG systems.

#### Retrieval method

The retrieval method best suited for legal documents is **Parent Document Retrieval**. This approach enables efficient retrieval of legal texts by searching within smaller chunks while still returning the full document. By doing so, it ensures both fast and precise access to relevant legal information while maintaining the integrity of the original content. This is particularly crucial in legal applications, where complete context is essential for accurate interpretation and decision-making.

#### Vectorstore and docstore

When using the Parent Document Retrieval method, two types of databases are required:

* Vector Database – Stores chunks in vector form, allowing efficient retrieval based on semantic similarity.
* Document Database – Stores full documents, ensuring that once relevant chunks are retrieved, they can be linked back to their original source.

During retrieval, the system first identifies relevant chunks from the vector database. These chunks then point to their corresponding full documents in the document database, providing complete context for the Generative Module. This ensures that responses generated by the LLM are both accurate and contextually comprehensive.

### Generative module

#### Generative model

After retrieving the relevant chunks, the user’s query along with these chunks is processed by an LLM to generate a response. The key steps in the generation module are as follows:

* Constructing the input prompt – The retrieved chunks are combined with the user’s query to form a structured prompt that provides the necessary context for the model.
* Generating the response using an LLM – A large language model LLM such as Gemini is used to generate an answer based on the enriched prompt.

Techniques used to optimize the output:

* Few-shot Chain of Thought: Guides the model through step-by-step reasoning to provide an answer, offering additional specific examples for the model to follow.
* Memory-Augmented RAG: Stores the conversation history to improve interactivity and accuracy in responses.

# EXPERIMENTAL SETUP

## Data usage

The data used in this study were collected from the Legal Library, consisting of 115 DOCX formatted documents. Among these, 61 documents are laws enacted before 2025, and 48 are new laws that came into effect on January 1, 2025. These documents include Laws, Circulars, and Decrees, providing a rich and diverse source of information for the research.

During the preprocessing phase, the entire text is converted to lowercase and all unnecessary whitespace is removed.

## Vectorstore and retrieval system

Traffic law documents are typically structured with sections such as Title, Summary, Articles, Article Titles, and Article Contents. To optimize the information retrieval process, we apply the Parent Document Retrieval method. Specifically, the original law documents are divided into sections based on individual articles. Each article serves as a Parent Document and contains related text segments chunks.

The chunking process is carried out using the Recursive Chunking method, which breaks down the main text into smaller sections based on the hierarchical structure of the document. This helps create appropriately sized chunks for effective storage and retrieval.

By using the Langchain library to assist in building the Parent Document Retrieval system and data storage, Langchain provides tools to implement this technique efficiently.

To store both the embedded vectors and the original text, we use PostgreSQL combined with the pgvector extension. This combination allows for efficient vector storage and retrieval while leveraging PostgreSQL's powerful features, such as ACID compliance and time-based recovery capabilities.

Using Langchain along with PostgreSQL and pgvector makes the system flexible, efficient, and easily scalable during implementation and operation.

## Large language model

The LLM will be used in three main components: Query transformation, Routing, and the Generation Module. A temperature setting of 0 is used in these models to return the most accurate results.

The prompting method used in all three components is Few-shot CoT, which guides the model step by step while providing specific examples for the model to follow. This method improves the model's output, making the results more accurate.

### Query transform

Query transformation will modify the user’s questions as follows:

* Remove terms such as "new law" or "old law" to improve retrieval efficiency.
* For law-related questions, if they are too detailed, Step-back prompting will be used to broaden the question, increasing retrieval capability, and providing additional relevant information.
* For questions not related to the law, the original query will be kept unchanged.

### Query routing

Routing will lead to three different directions. The Routing directions include:

* A database containing old traffic law documents prior to 2025, if the user’s question is about current laws or mentions new laws.
* A database containing traffic law documents that are effective from January 1, 2025, if the user's question refers to old laws.
* The LLM model will directly answer if the user's question is related to normal chatting.

### Reponse

In the Generation Module, the output will be determined by the following prompt:

* The prompt will include the user's query and the Parent documents linked to chunks relevant to the user’s question. These documents will be attached with metadata, and the metadata will be crucial in formatting the output response for the user.
* The prompt will include the chat history to interact with the user and answer the question more accurately. The information in the chat history can also be used for comparisons.

The main format for the output will be:

* Source document: Document name
* Document title: Title of the document
* Article: Title of the article
* Content:
  + Content of the article

Along with the main format, additional criteria will enhance the aesthetics of the user response:

* Capitalizing the first letter of each line
* Breaking lines between sub-sections: a), b), c), ...

# EXPERIMENTAL RESULTS

## Test dataset

The test dataset consists of questions and answers collected from websites related to road traffic laws, comprising 98 questions along with their ground truth answers. These question-answer pairs cover various topics, including:

* Concepts and Traffic Rules: Definitions of key terms such as "lane," "median strip," and "highway," as well as regulations governing road use.
* Traffic Signs Regulations: Meanings and compliance requirements for different types of traffic signs, including prohibitory signs, warning signs, mandatory signs, and directional signs.
* Highway Traffic Rules: Regulations on lane merging, lane exiting, speed limits (both minimum and maximum), and prohibited behaviors on highways.
* Alcohol Limit Regulations for Vehicle Operators: Maximum permissible blood alcohol concentration and breath alcohol levels for drivers.
* Technical Safety and Environmental Protection Regulations for Vehicles: Technical standards, safety inspections, and environmental protection requirements for vehicles in traffic.
* Traffic Violation Penalties: Fines and enforcement measures for various violations of road traffic laws.

## Evaluate with RAGAS

### What is RAGAS?

The Retrieval-Augmented Generation Assessment System (RAGAS) is an advanced framework designed for the automated, reference-free evaluation of RAG pipelines. RAG systems combine retrieval mechanisms with LLMs to provide accurate and contextually relevant responses by accessing information from reference textual databases. Evaluating these systems is complex, as it involves assessing various dimensions, including the retrieval component's ability to identify pertinent context passages and the LLM's capacity to generate faithful and relevant content. RAGAS addresses this challenge by offering a suite of metrics that evaluate these aspects without relying on human-annotated data, thereby facilitating faster and more efficient evaluation cycles.

### Metrics for evaluating

The RAGAS offers a comprehensive suite of metrics to evaluate the performance of RAG systems. These metrics are designed to assess the effectiveness of each component within the RAG pipeline, including:

* Faithfulness: Evaluates the factual accuracy of the generated content by comparing it to the information in the retrieved documents.
* Answer Relevancy: Measures how well the generated answer aligns with the original question.
* Context Recall: Assesses the system's ability to retrieve relevant text segments from the database.
* Context Precision: Evaluates the accuracy of the retrieved text segments in meeting the information needs of the question.

### Evaluating result with RAGAS

Represented scores by RAGAS:

Table 5.1. RAGAS evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Faithfulness | Answer relevancy | Context recall | Context precision |
| Gemini-1.5-pro | 0.73 | 0.42 | 0.38 | 0.23 |
| Gemini-1.5-flash | 0.72 | 0.42 | 0.36 | 0.25 |

# REFERENCES

Vietnamese

…

English

Shanmukha Ranganath (2024). *RAG 101: Chunking Strategies*. https://towardsdatascience.com/rag-101-chunking-strategies-fdc6f6c2aaec/

Rajan Sethi (2024). *Retrieval Strategies in Retrieval-Augmented Generation.* https://medium.com/@rajan.sethi36/retrieval-strategies-that-can-be-helpful-in-retrieval-augmented-generation-72e5856112ee

Samar Singh (2024). *Mastering RAG: Advanced Methods to Enhance Retrieval-Augmented Generation.* https://medium.com/@samarrana407/mastering-rag-advanced-methods-to-enhance-retrieval-augmented-generation-4b611f6ca99a

Guy Korland (2024). Advanced RAG Techniques: *What They Are & How to Use Them.* https://www.falkordb.com/blog/advanced-rag/

Leonie Monigatti (2023). *Evaluating RAG Applications with RAGAs.* https://medium.com/towards-data-science/evaluating-rag-applications-with-ragas-81d67b0ee31a