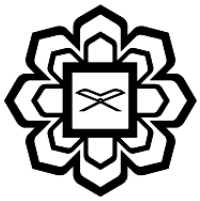
# AI PERSONAL ASSISTANT USING VISUAL- LANGUAGE RETRIEVAL-AUGMENTED GENERATION (RAG)

IBRAHIM BIN NASRUM 2116467

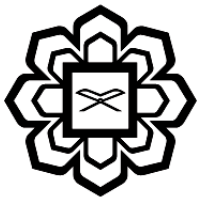


DEPARTMENT OF MECHATRONICS ENGINEERING
KULLIYAH OF ENGINEEERING
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
JUNE 2024

AI PERSONAL ASSISTANT USING VISUAL-LANGUANGE RETRIEVAL-AUGMENTED GENERATION

IBRAHIM BIN NASRUM

**PROJECT SUPERVISOR:**
**PROF. DR. ABDUL HALIM**



**A REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR A DEGREE OF BACHELOR OF MECHATRONICS ENGINEERING WITH HONOURS**

# DECLARATION

I hereby declare that the project entitled "AI Personal Assistance Using Visual-Language Retrieval Augmented Generation" is the result of my investigations, except where otherwise stated, submitted to the International Islamic University Malaysia under supervision of Prof. Dr. Abdul Halim, Department of Mechatronics Engineering. I also declare that it has not been previously or concurrently submitted for any other degree at IIUM or other institutions.

Ibrahim Bin Nasrum

Date: 12/6/2025

# APPROVAL

I certify that I have supervised and read this study and that in my opinion, it conforms to acceptable standards of scholarly presentation, and it is fully adequate, in scope and quality, as Final Year Project report as partial fulfilment for a degree of Bachelor of Mechatronics Engineering with Honours.

……

Assoc. Prof. Abdul Halim Embong

Supervisor

# ABSTRACT

In the context of rapid digital transformation, this project presents a practical AI personal assistant designed for C-level executives who require immediate, reliable access to company knowledge. The assistant is built on a Retrieval-Augmented Generation (RAG) framework, integrating MiniLM embeddings, FAISS vector retrieval, and Llama-family language models for text, with additional support for vision-language queries using llava:latest. The system processes both textual and visual data, enabling it to answer queries about sales, HR, and operations from structured (CSV) and unstructured (image) sources. A Gradio-based user interface allows interactive querying and response display. Experimental evaluation used 109 text and 15 visual queries reflecting real CEO information needs. Results show that deterministic routing and evidence gating in the RAG pipeline significantly reduce numeric hallucination compared to LLM-only baselines. Text model satisfaction rates ranged from 78% to 88%, with llama3:latest performing best. Visual model testing achieved 100% OCR extraction success but with higher latency (61.5s average) and an estimated 80% satisfaction rate. Limitations include a single-domain dataset, small test set size, and reliance on automated metrics without human validation. This research demonstrates the feasibility and trade-offs of deploying a hybrid RAG assistant with both text and visual capabilities for executive decision support. The prototype highlights the importance of robust routing, evidence-based answering, and system auditability. Future work should expand test coverage, improve UI usability, and incorporate human evaluation to further validate and enhance the system for real-world enterprise use..

# ACKNOWLEDGEMENTS

In The name of Allah, the, the Most Gracious and the Most Merciful. By the grace of Allah SWT, I have successfully completed Final Year Project 1, "AI Personal Assistant Using Visual Language Retrieval Augmented Generation". Every opportunity, experience, and challenge overcome is a testament to His boundless blessings and a step towards creating positive change for those in need.

Besides, I would like to extend my heartfelt gratitude to Assoc. Prof. Abdul Halim Embong, my esteemed supervisor, for his invaluable guidance and mentorship. His expertise, insightful feedback, and unwavering belief in my potential have greatly contributed to the development of this project. I am sincerely grateful for the countless hours he invested in nurturing my academic growth and pushing me to excel.

My deepest appreciation goes to the dedicated lecturers who have significantly enriched my academic journey. Their commitment to teaching, sparking curiosity, and encouraging critical thinking has provided me with the tools and knowledge necessary to overcome challenges and succeed.

A special note of thanks to my family, whose unwavering support and sacrifices were my foundation throughout this journey. Their constant encouragement and joy have been my strength. To my friends, who have stood by me through every trial and triumph, your camaraderie, understanding, and occasional laughter have been invaluable. I am deeply thankful for your steadfast support and belief in my aspirations.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| IIUM | International Islamic University Malaysia |
| AI | Artificial Intelligence |
| RAG | Retrieval-Augmented Generation |
| LLM | Large Language Model |
| BLIP | Bootstrapped Language-Image Pretraining |
| BLIP-2 | Bootstrapped Language-Image Pretraining version 2 |
| FAISS | Facebook AI Similarity Search |
| UI | User Interface |
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| NLG | Natural Language Generation |
| ML | Machine Learning |
| DL | Deep Learning |
| RNN | Recurrent Neural Network |
| Seq2Seq | Sequence-to-Sequence Model |
| MiniLM | Minimal Language Model (lightweight transformer model) |
| BERT | Bidirectional Encoder Representations from Transformers |
| LLaMA | Large Language Model Meta AI |
| QuilM-RAG | Quantized and Lightweight Multimodal Retrieval-Augmented Generation |
| CLIP | Contrastive Language-Image Pretraining |
| CRM | Customer Relationship Management |
| ERP | Enterprise Resource Planning |
| ETL | Extract, Transform, Load |
| ELT | Extract, Load, Transform |
| SQL | Structured Query Language |
| KG-RAG | Knowledge Graph-enhanced Retrieval-Augmented Generation |

| | |
|---|---|
| PEFT | Parameter-Efficient Fine-Tuning |
| RAM | Random Access Memory |
| OCR | Optical Character Recognition |
| CSV | Comma-Separated Values |
| PDF | Portable Document Format |
| JSON | JavaScript Object Notation |
| API | Application Programming Interface |
| DOC | Document |
| CEO | Chief Executive Officer |
| RQ | Research Question |
| VLMs | Vision-Language Models |
| BAS | Business Assistant Service |
| NER | Named Entity Recognition |
| KG | Knowledge Graph |
| Colab | Google Colaboratory (Cloud-based development platform) |
| Embedding | Vector representation of data (text/image) |

# CHAPTER 1
# INTRODUCTION

## 1.1 BACKGROUND OF STUDY

In today's enterprise landscape, AI-assisted tools are no longer niche, they're essential. CEOs and top-level managers are increasingly embracing AI to streamline operations and improve data accessibility. A recent *Investopedia* report reveals that 79% of CEOs believe generative AI increases efficiency, and 55% are actively exploring or deploying AI tools in their organizations (Investopedia, 2024). This rising adoption underscores a growing demand for intelligent systems that can support leadership in accessing and interpreting company data without relying on intermediary staff

AI personal assistants are leading this digital transformation, with widespread global adoption across both personal and professional domains. According to *SEOSandwitch* (2024), over 4.2 billion people currently use AI personal assistants, and this number is expected to grow significantly as businesses seek smarter tools for workflow automation. The Hostinger AI Statistics Report (2024) reveals that around 56% of business leaders have reported early or moderate AI adoption, while only 20% have fully integrated AI across multiple business functions. This indicates that while AI experimentation is underway in many organizations, full-scale deployment across all operations is still in its early phases. For CEOs, who often manage large volumes of information and strategic decisions, these efficiencies are

game-changing.

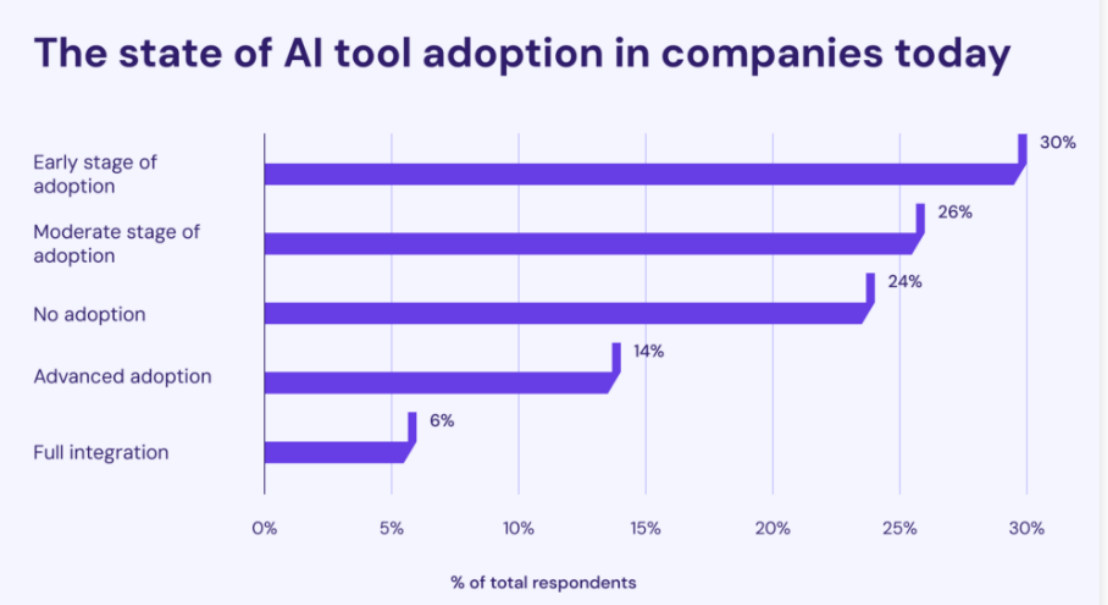



**Figure 1.1:** Percentage of around 56 percent business leaders report early or modearate AI adoption. Source: Salminen & Mauladhika, 2025 (Hostinger)

Looking ahead, the capabilities of AI agents are expected to evolve even further. *IBM* (2024) predicts that by 2025, AI agents will be fully autonomous in executing tasks, capable of interacting with complex business systems and making decisions with minimal or no human supervision. According to Hostinger (2024), despite concerns surrounding AI ethics and cybersecurity, the report also notes that 58% of companies plan to increase their AI investments in 2025, particularly in sectors such as logistics and customer service. These statistics illustrate a strong momentum toward AI implementation, reinforcing the growing demand for intelligent systems that streamline operations and support strategic decision-making.This trajectory highlights the urgent need for AI personal assistants that are not only conversational but also multimodal able to process both textual and visual data such as dashboards, reports, and scanned documents.

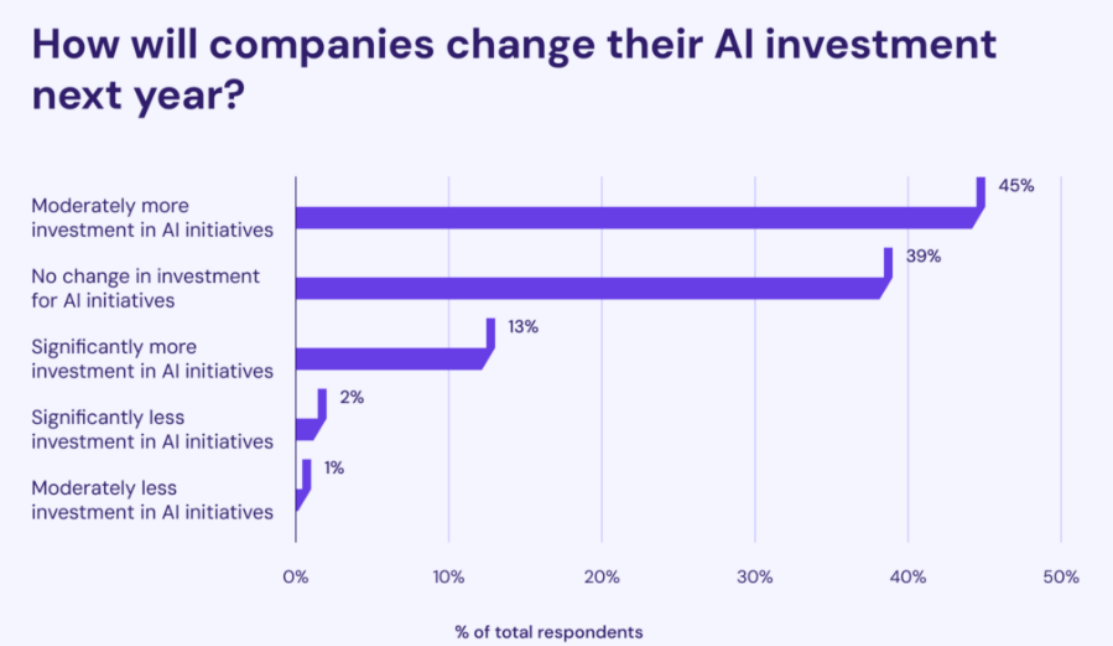



**Figure 1.2:** Nearly 60% of businesses plan to increase AI investments in 2025. Source: (Salminen & Mauladhika, 2025 (Hostinger)

This situation underscores the need for a secure, multimodal AI assistant that understands text and images, retrieves internal data, and generates context-aware responses in real time. This project addresses that gap, offering an innovative solution tailored to the unique demands of executive information workflows.

## 1.2 PURPOSE OF THE STUDY

The purpose of this study is to design and evaluate a prototype of an AI personal assistant that integrates a vision-language model with a retrieval-augmented generation (RAG) framework. This assistant is intended to support company executives, particularly CEOs, in accessing internal data efficiently without relying on staff for manual retrieval. The short-term purpose is to demonstrate the system's ability to process both visual and textual inputs to provide accurate, context-aware responses. The long-term purpose is to encourage the development of more advanced, multimodal AI tools that can assist in corporate decision-making processes. It is hoped that this study will serve as a reference for future research and practical implementation of intelligent assistants in enterprise environments.

## 1.3 STATEMENT OF THE PROBLEM

Modern CEOs often face significant delays when accessing internal company information due to reliance on manual data retrieval by employees. Existing tools like dashboards or standard chatbots are limited in their ability to provide seamless access to both textual and visual data, reducing efficiency in fast-paced decision-making environments. Most AI assistants lack the multimodal capability to interpret charts, scanned documents, or structured tables, leading to incomplete or inaccurate responses. Furthermore, current solutions do not personalize output based on enterprise knowledge. This gap creates the need for an intelligent AI personal assistant that combines vision-language understanding with retrieval-augmented generation to deliver accurate, context-aware answers in real time empowering CEOs to make quicker, more informed decisions independently.

## 1.4 OBJECTIVES OF THE RESEARCH

1. To develop a functional prototype of an AI personal assistant that integrates a vision-language model for multimodal understanding.
2. To evaluate the system's decision-making performance based on a series of predefined user queries and real-world executive scenarios.
3. To optimize the system architecture for efficient information retrieval and low-latency response generation using open-source tools on limited computing resources.

## 1.5 SCOPE OF WORKS

i. Design and Implementation of System Architecture. This project includes designing a modular system architecture that integrates vision processing, language models, and retrieval mechanisms. The architecture ensures smooth data flow and scalability, suitable for deployment in real-world organizational environments such as executive dashboards.

ii. Development of Vision-Language Model Integration. The assistant incorporates a vision-language model like CoPalLi to process and interpret both image and textual inputs. This multimodal approach allows the system to extract insights

from visual materials such as charts, dashboards, or uploaded company documents.

iii. Retrieval-Augmented Generation (RAG) Framework. The system employs an RAG framework using FAISS to retrieve relevant company data before response generation. This enhances factual consistency by grounding responses in internal knowledge bases, ensuring more accurate and contextually relevant outputs for CEO queries.

iv. Response Generation Using Llama 3. Llama 3 is utilized as the primary language model to generate responses based on retrieved context and user prompts. Its instruction-following capability and strong performance enable it to deliver executive-level answers with a high degree of coherence and relevance.

v. Evaluation and Testing with Real-World Scenarios. The prototype is tested using predefined CEO queries reflecting actual business needs. Metrics such as response accuracy, processing time, and user satisfaction are evaluated to measure the assistant's performance in decision-making support.

vi. User Interface and Interaction Layer Development. A web-based user interface is developed for seamless interaction with the assistant. It enables users to input queries, upload visual data, and receive real-time responses, making it accessible and user-friendly for non-technical executive users.

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 INTRODUCTION

Since AI Assistant like ChatGPT or Deep seek has been viral in helping people find information, it has changed and redefined totally how individual, or an organization interact with information. Nowadays, AI assistants have been used recently in every sector, especially involving enterprise environment. Executives' top management like the CEO usually need to make a fast decision and acquire accurate information with context-rich real time data access to running their company effectively and efficiently. The literature review explores the current advancements in AI Personal Assistant specifically using Vison-Languange Models (VLMs) and Retrieval Augmented Generation (RAG) techniques. This review has been categorized into eight topics namely the AI Personal Assistants in Business and Enterprise, Natural Languange Understanding and Retrieval-Augmented Generation, Vision Language Models (VLMs) for Multimodal Understanding, System Architecture for AI CEO Assistant, Data Handling: Storage, Retrieval and Integration, Multimodal Processing and Response Generation, Ethical and Security Considerations and Research Gaps and Conclusion. By integrating this technology, it will help improve the quality of accessing data by the individual top executive like CEO. In summary, this chapter aims to provide a comprehensive overview of existing literature on AI-Personal Assistant, offering insights into the current state of research and development in this field.

## 2.2 AI PERSONAL ASSISTANT IN BUSINESS AND ENTERPRISE

Referring to Cambridge Dictionaries, enterprises mean an organization, especially a business or a difficult and important plan, especially one that will earn money. In the context of Enterprise AI Assistant, it means an AI that is being built personally to help the company. Business Assistant Service (BAS) is important in today's fast-paced economic environment as many modern entrepreneur's face challenges to analyze numerous information quickly with rapidly changing market conditions and pressure to make optimal decisions fast. (Ivashchenko et al., 2020).Because of the challenges, enterprises are increasingly in need of an AI assistant

service to handle these challenges. Using AI technology, data can be grouped and processes from many sites to make conclusions and apply data driven decision making process (Haleem et al., 2022). According to (Marikyan et al., 2022), since the coronavirus (COVID-19) occurs, it has reshaped work pattern of organization to adopt remote working where digital assistant based on AI will be useful for work outcomes. Intelligence Personal Assistant has been built into mobile operating systems like Apple's Siri, Google Now and Microsoft Cortana to help user do tasks more efficiently every time (Goksel Canbek & Mutlu, 2016). (Ivashchenko et al., 2020) states that AI Assistant service main goal is to speed up, optimize and simplify entrepreneurs who have operated or just want to start the business in making decisions process. Furthermore, AI is also useful in marketing purposes as it can collect data and generate algorithms that will be useful in targeting the content for customers and employing the channel at the right moment (Haleem et al., 2022). From the point stated before, we can draw a conclusion and see that AI is useful in business to make decisions quickly and ease the process management with a short time. The use of AI Assistant in business is still in general which recently being used as an individual. The AI Assistant can be optimized better by specifically using company data to run the task more efficiently instead of using general open-source data to help make decisions.

## 2.3 NATURAL LANGUAGE PROCESSING (NLP)

### 2.3.1 NLP and Its Role in Enterprise AI Assistants

Natural Languange Processing is important as it serves as the backbones for understanding human or user language in AI assistant. Based on (Bharathi, n.d.-a; Cofino et al., 2024) Natural Language Processing (NLP) significantly essential technology that can be enhanced together with Artificial Intelligence (AI), Machine Learning (ML) and deep learning to improve human-computer interaction that will benefit in improving decision-making and operational effectiveness. NLP is also useful in transforming unstructured data such as chat, emails and social media post into more structured data. (Bharathi, n.d.). By applying transformation, enterprise AI assistants can understand and generate human-like response effectively in many business functions.

### 2.3.2 Broad Classification of NLP

NLP are classified into two categories which is Natural Language Understanding (NLU) which is human to machine and Natural Languange Generation (NLG) which is machine to human (Abro et al., 2023). According to (Su et al., 2020),the difference between the two is that NLG builds a corresponding natural language sentence based on the input semantics representation, whereas NLU extracts the semantics from the provided tests. Natural Language Understanding (NLU) in dialogue systems usually use semantic frames to understand what the user is saying. It can be done by analyzing user input to identify the domain, which is the main topic, predicting what user wants to do, and extract relevant information. In the past, the system handled each utterance in isolation which means look at each user message by itself, but now the system improves accuracy by incorporating context and recognizing speaker roles to enhance comprehension (Su et al., 2020a). Meanwhile, when the system knows what to say, Natural Language Generation (NLG) in dialogue systems will be used to produce clear and natural sentences in response based on the system's intended message. Older systems using Recurrent Neural Network (RNN) models lack flexibility and scalability, but in the latest system they use sequence to sequence (seq2seq) models which offer more diverse human response by learning directly from real conversation. (Su et al., 2020a).

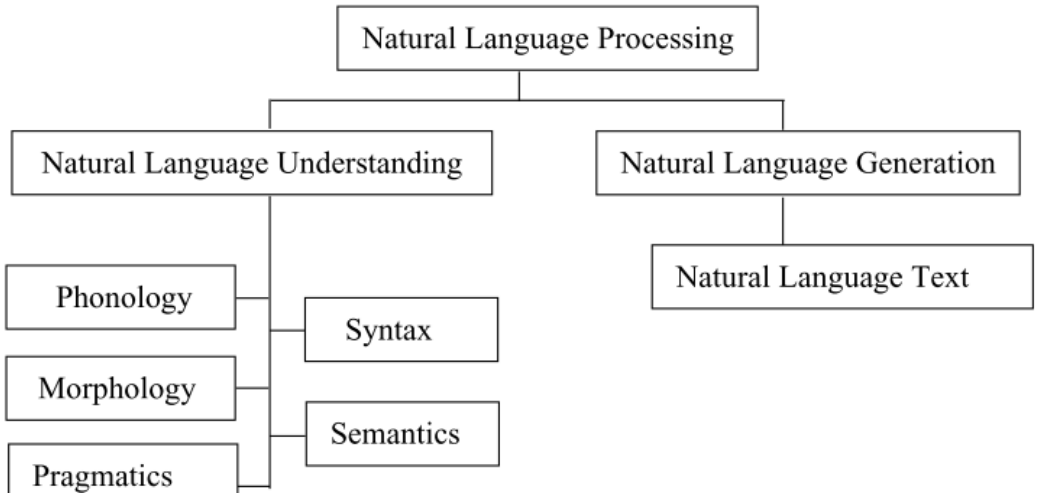



**Figure 2.1:** Broad Classification of NLP (Abro et al., 2023a; Solaimalai et al., 2024)

### 2.3.3 NLP Pipeline and how it works

The NLP pipeline is very important as it provides a structure framework that enables machines to process and interpret human language. NLP usually will start with text preprocessing. According to (Bharathi, n.d.-a; Supriyono et al., 2024a; Yagamurthy, 2023b), text preprocessing is the process on analyses the text to the level of individual sentences and words which includes cleaning and breaking down text using tokenization, segmentation, stemming and others. After that, the text representation process is very crucial to convert unprocessed text into a structured format that can be understood by machine learning more effectively (Supriyono et al., 2024a). Then, the feature selection process will identify important patterns by using filter-based techniques, followed by selection of models for training using machine learning algorithms(Su et al., 2020b). Finally, the trained model is deployed for tasks like sentiment analysis or translation. The model's performance will be evaluated and improved continuously. This systematic flow ensures accurate and efficient language understanding by computers.

### 2.3.4 Key NLP Techniques for Enterprise Applications

Human language is very complex and cannot be understood easily by machine. It is required to have a multiple level of processing to be understood by machine. This involves not only interpreting the meaning of individual words but also analyzing how those words and phrases relate to one another (Bharathi, n.d.). Because of this, Natural Language Processing (NLP) systems require to follow procedures to interpret natural text effectively. According to (Abro et al., 2023a), Deep Learning (DL), a subfield of Machine Learning (ML), is the foundation of most of these NLP systems. Figure 2 illustrates how DL is very useful and highly adaptable framework for describing the surroundings for both spoken and visual information. Common applications that are being used like machine translation, sentiment and emotion analysis, summarization, and NER may need to be understood properly to accommodate the system framework effectively. Based on (Bharathi, n.d.-b), machine translation plays a vital role in making conversion between language like tools of Google Translate. This is very crucial in breaking language barriers. Meanwhile, Named Entity Recognition (NER) is one key

of information extraction in NLP which the main purpose is to detect and classify names, organization, locations, dates or numbers within a text.(Yagamurthy, 2023a). Besides, summarization is application involves synthesizing text that is usually found in search result into concise summaries to be used in academic databases and indexes. Furthermore, sentiment analysis is one of the most useful in NLP as it can identify emotion (Abro et al., 2023b; Pothuri & Varma Pothuri, 2024). For example, sentiment analysis is good in surveys, reviews and social media. In sum, the use in NLP has been completely enhanced text summarization method, but it still having some difficulty on comprehending the context which still require assistance with complex words and subtle nuances.

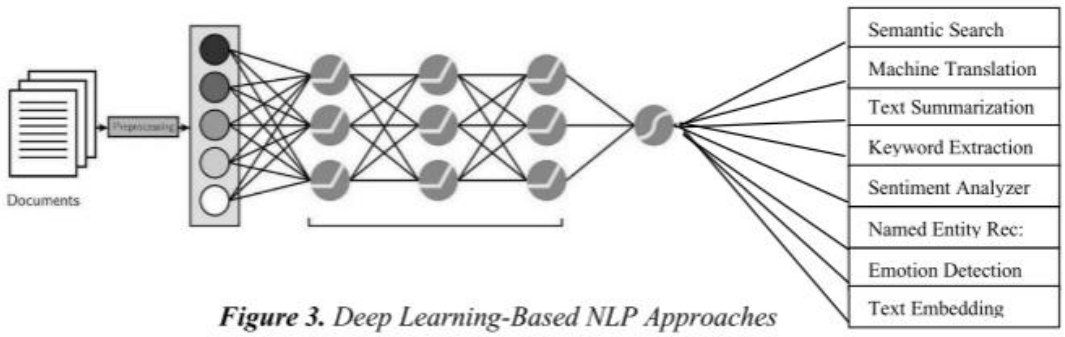



***Figure 3.*** *Deep Learning-Based NLP Approaches*

**Figure 2.2:** Deep Learning-Based NLP Approaches (Abro et al., 2023a)

### 2.3.5 Challenges and Current Trends

Natural Language Processing encounters some challenges, like dealing with ambiguity, context sensitivity, and the variety of human languages and dialects (Pothuri & Varma Pothuri, 2024; Supriyono et al., 2024b). As human language is too complex and sometimes have multiple meanings, it may need to interpret natural language truly, so the output becomes more accurate. The main challenge for NLU in NLP is that it is required for multiple roles like NLG rules as NLU having difficulty in interpreting natural language completely (Abro et al., 2023b). It's also difficult doing text summarization as it deals with large amounts of meticulously labeled data and superior quality that take quite number of sources and time. (Supriyono et al., 2024b).

On the other hand, current trends have shown advancement and have improved understanding and generation tasks with the use of large language models (LLMs) like GPT and BERT. Bidirectional Encoder Representations from Transformers (BERT) is conceptual simple and empirically powerful, that are designed to pretrain deep bidirectional representation from unlabeled text (Devlin et al., n.d.). Besides, there's also growing interest in multilingual NLP, low-resource language processing, and explainable AI, aiming to make NLP systems more inclusive, transparent, and adaptable to real-world applications. Figure 3 shows the global trend in the application of NLP being used that increases rapidly from 2019 to 2025. We can see that text embedded is the most demanding application in NLP recently.

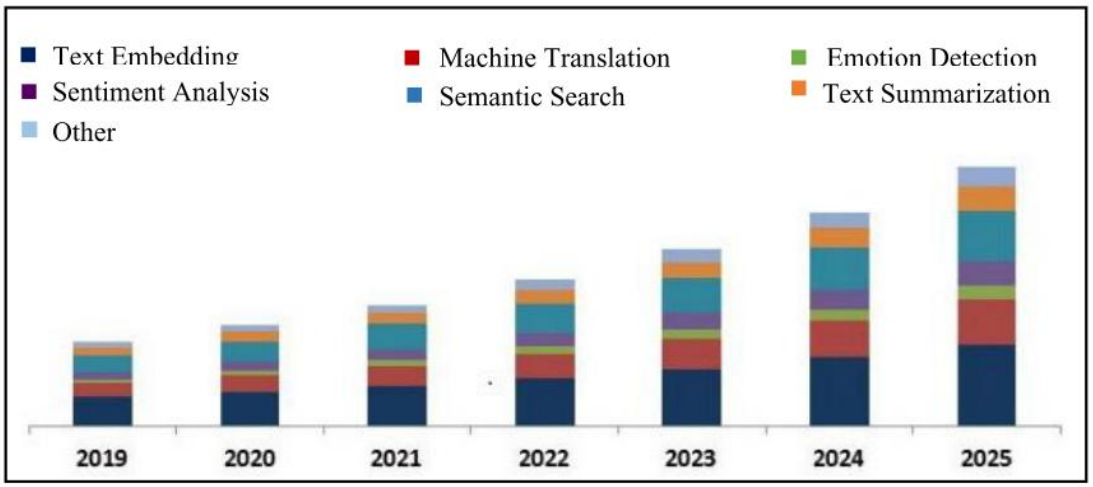



**Figure 2.3:** Analytics Of NLP Global Impact (Abro et al., 2023a; Solaimalai et al., 2024)

## 2.4 RETRIEVAL-AUGMENTED GENERATION

Retrieval Augmented Generation (RAG) is an approach used in Natural Language Processing (NLP) that merges the power of the pre-trained languange mode with support of knowledge obtain from the database data from open-source resources. (Bruckhaus, n.d.). RAG is also commonly divided into two important techniques, which information retrieval and text generation. For enterprises, RAG can help evaluate the data from the company and generate the best answers from the database. According (Packowski et al., 2024) , RAG is the best efficacious way in avoiding hallucinations and factual inaccuracy when answer questions while using the large language model

(LLMs). In simple understanding, RAG is important to improve language models like ChatGPT or BERT-based models to answer questions or produce accurate and relevant text.

Three basic components in RAG are retriever, generator and knowledge base. The retriever acts as information retrieval and semantic search in searching or retrieving any relevant information based on similarity from the input query.(Bruckhaus, n.d.). After that, generators like GPT-4, Claude Opus or T5 which is a large pre-trained language model, will generate output accurately, consistently and contextually response. (Bruckhaus, n.d.) Furthermore, the knowledge base is a collection of structure and unstructured documents that can be collected from.(Bruckhaus, n.d.) In related with this research, it may be data of the company that we want to focus on. Below is the typical RAG architecture solution that has been discussed in the research of (Klesel & Wittmann,2025)

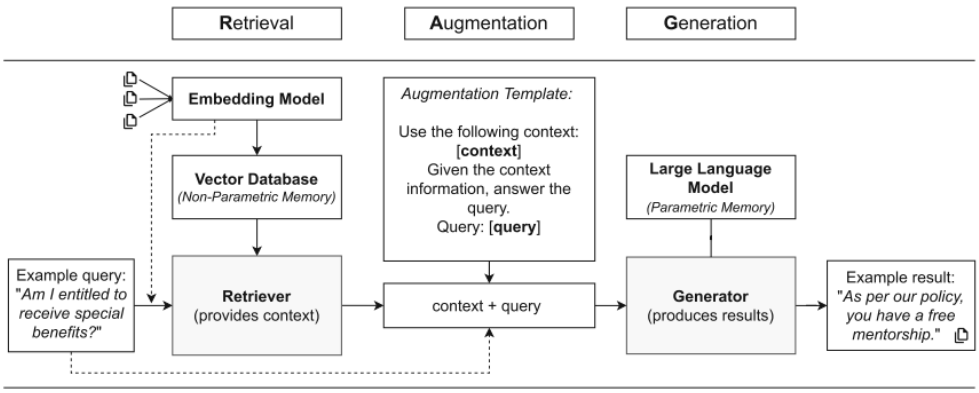



**Figure 2.4:** RAG Architecture Diagram (Klesel & Wittmann, 2025)

Meanwhile, there are findings that suggest the use of QuiIM-RAG (Advancing Retrieval-Augmented Generation with Inverted Question Matching) to reduce common problems like information dilution and hallucination that are usually faced by traditional RAG when dealing with large amount of structured data. (Saha et al., 2024) finds that the traditional dataset may not give a full alignment for user inquiry intention but by using the custom dataset and QuIM- RAG will help the LLM to be more accurate in avoid provide irrelevant information and unneeded details. Table 1 below shows the

finding result between traditional RAG and QuIM-RAG that shows QuIM- RAG effectiveness.

**Table 2.1**: Comparison of performance metrics for traditional RAG and our novel QuIM-RAG using traditional and custom datasets (Saha et al., 2024)

| Evaluation Matrix | Traditional RAG | | QuIM-RAG | |
|---|---|---|---|---|
| | Traditional | Custom | Traditional | Custom |
| Faithfulness | 0.69 | 0.72 | 0.91 | 1.00 |
| Answer Relevancy | 0.79 | 0.82 | 0.93 | 0.99 |
| Context Precision | 0.45 | 0.69 | 0.82 | 0.92 |
| Context Recall | 0.39 | 0.45 | 0.60 | 0.74 |
| Harmfulness | 0 | 0 | 0 | 0 |
| BERTScore P. | 0.32 | 0.37 | 0.55 | 0.63 |
| BERTScore R. | 0.29 | 0.35 | 0.63 | 0.71 |
| F1 Score | 0.31 | 0.36 | 0.59 | 0.0.67 |

However, some challenges that may be faced in implementing RAG in the enterprise sector are privacy, security and governance regulations that must ensure sensitivity of the customer never exposed or misused during the retrieval and generation process. (Bruckhaus, n.d.). These challenges may be overcome by strictly using the data of the company for the uses in the company only. Besides, others researcher finds that using RAG that involves large-scale applications may result in higher computational and financial costs. To overcome this research, I find that RAG can be implemented with open source LLMs and Haystack's orchestration that allow flexibility in development of scalable and financial cost applications. (Papageorgiou et al., 2025).

To sum up, Retrieval Augmented Generation is a potent method that combines retrieval and generation to improve the performance of language models. When working with real-world data, it makes models more accurate, current, and dependable. RAG is anticipated to be crucial to many real-world NLP applications as this discipline develops, ranging from customer service to healthcare and education.

## 2.5 VISION-LANGUAGE MODELS (VLMS) FOR MULTIMODAL UNDERSTANDING

Visual Language model is an additional system of the model to enhance its capabilities on having visual inputs. (Bordes et al., 2024). VLMs is designed to have visual inputs and textual inputs with ability to process and its ability to understand both types of data is called multimodal understanding. AI models have always been designed to deal with either text or images, but not at once. For instance, language models like BERT or GPT could comprehend and produce text, while computer vision models could identify objects in images. However, a lot of real-world activities, including describing an image, responding to a question about a chart, or reading text from a scanned document, call for knowledge of both input types. By learning from paired image-text data, vision-language models overcome this difficulty and can comprehend the connection between what is written or spoken and what is seen.

One of the popular VLMs is CLIP (Contrastive Language-Image Pretraining) by OpenAI which is an efficient and scalable method which is learning from natural languange supervision. (Radford et al., 2021) found that CLIP can outstand more than the best open public ImageNet in terms of computational efficiency, robustness and accuracy. In simple terms, CLIP can understand what is in an image and match it with the right text. For example, if given a photo of a cat, CLIP can choose the correct caption from many possible options.

Another powerful model is BLIP, which improves earlier models by using better image encoders and language decoders to generate more accurate captions and answers. (Li et al., n.d.) suggest using BLIP, which is a smart computer system that can learn from noisy image-text data. It contains MED as the brain and CapFilt which method to clean and improve the training dataset. (Li et al., n.d.) also find that, due to noise web text that recently occurred in visual languange learning, they have proposed CapFilt that has utilized web datasets in more effective ways.

GPT-4V (GPT-4 with Vision) and other more sophisticated VLMs go even farther. In a single discussion, they enable the model to process both text and visuals.

For instance, a user can display a form and enquire, "Is this document filled out correctly?" or upload a graph and enquire, "What trend is shown here?" The model's ability to react by examining both the text and the visual structure demonstrates how potent multimodal AI has grown. (Qi et al., 2023) presents an in-depth comparative study of two pioneering models, which are Google Gemini and Open AI GPT-4V that both excel on interaction with humans, temporal understanding and assessments in both intelligence and emotional quotients. The findings show that both have excellent strength. While Gemini excels in providing comprehensive, in-depth answers with pertinent images and links, GPT-4V stands out for its accuracy and conciseness in responses. (Qi et al., 2023).

Optical Character Recognition (OCR) is another field associated with VLMs. OCR is an input device that is being used to read printed text.(Gomathy Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya et al., n.d.). VLMs can read and extract text from images, such as scanned documents, signs, or screenshots, using OCR tools like Tesseract or Google Vision API. To provide answers, summarize, or extract important information from visual data, this text can subsequently be integrated with language comprehension.

Furthermore, VLMs are strong because they are adaptable. They can be applied in many ways, such as assisting business users or CEOs in making choices through the analysis of papers, reports, and charts. VLMs give the AI personal assistant the ability to comprehend user-upload documents and photos in addition to responding to written queries. For instance, the assistant can process and intelligently respond when the CEO uploads a picture of a graph or a handwritten note and requests insights.

However, there are certain drawbacks to vision-language models as well. According to (Zhang et al., 2023), one difficulty is a data a fine-grained vision-language correlation modelling. There is still limited research exploring their effectiveness in dense prediction tasks like object detection and semantic segmentation. These tasks require fine-grained, local-level understanding between vision and language, which remains a challenge due to the scarcity of pre-training strategies focused on patch-level

or pixel-level correspondence as a result, more research is needed to enhance zero-shot performance in such fine-grained visual recognition tasks (Zhang et al., 2023).

To sum up, vision-language models are essential for creating AI systems that have simultaneous "seeing" and "reading" capabilities. They enable AI personal assistants to do more than just read text, which makes them smarter and more practical in everyday situations, particularly in professional contexts where it's critical to comprehend documents, charts, and photos.

## 2.6 DATA HANDLING: INGESTION, STORAGE, RETRIEVAL, PROCESSING ND INTEGRATION

### 2.6.1 Data Pipeline

AI Personal Assistant needs good storage so that you can store the data generated when asked. Data must be handled correctly so the AI assistant can work fastly and accurately when to get access to both organized and unstructured data. The data handling will cover the data's storage location, retrieval method, and system integration. The data engineering pipeline, as Figure 1, shows the process involved in data management.

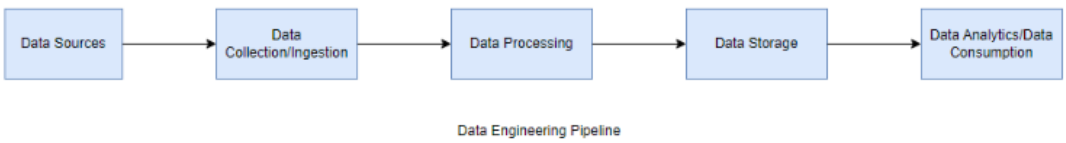



**Figure 2.5**: Data Engineering Pipeline (Balakrishnan, 2024)

### 2.6.2 Data Sources

Data is the fundamental thing needed for AI to work. Without enough data, AI cannot generate accurate and the best answer. There are studies from (Khan & Al-Badi, 2020) that classified data sources into four categories shows in figure 1. Social media network is the common real time unstructured data that can be analyzed further using NLP. Business and administration systems are commonly generated both structured and unstructured data from credit card activity or others. Besides, Government and administrative recently produced data like health, trade and population. Lastly, a ubiquitous system is collecting real time data from users through mobile devices and

sensors. Additionally, data sources like HR databases, Customer Relationship Management (CRM) and Enterprise Resource Planning (ERP) system also can be used to enhance ai assistant in decision making. CRM refers to tools or systems that companies use to manage interaction with customers while ERP is a system that helps manage company business processes in real time. (Jhurani, 2024) find that organization can achieve more advantages by integrating AI to enhance CRM functionality. So, from this we can understand that many sources can be used to be collected. For the company, the real data is needed in terms of both structured and unstructured data to ease and improve AI Personal Assistant.

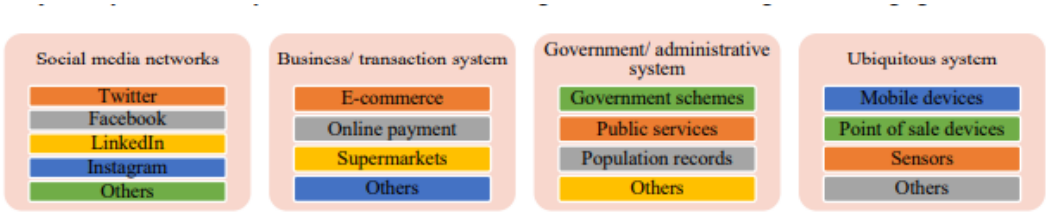


Fig 1. Classification of emerging data sources

**Figure 2.6.** Classification of emerging data sources 3. The Process of Decision Making Using Emerging Data Sources and Chall (Khan & Al-Badi, 2020)

### 2.6.3 Data Ingestion/Collection

Data ingestion is a process in data handling management which collects data from many sources to be processed, storage and analysis. (Alwidian et al., 2020; Elkina & Zaki, 2023.) . The goal of data ingestion is to make sure data is clean and stored appropriately and can be used easily. (Jhurani, 2024) suggest the use of RAG, RLF and fine-tuning that can give benefits to the performance. One studies state that data ingestion is an important process recently discussed with the concept of ETL which is extract, transform, and load (Alwidian et al., 2020). The data warehouse once implemented using ETL for managing structured data but when most of the data generated is unstructured data has changed the system by putting NoSQL in the system.(Elkina & Zaki, 2023.) . At the end, the new solution which is called data lake appears by using ELT (Extract, load and transform) which is more complex and requires longer execution time that offers more flexibility in handling many types of data. (Elkina & Zaki, 2023.) From this research, we can see that we can use ELT for data ingestion as for AI Personal Assistant is needed to ingest many types of data.

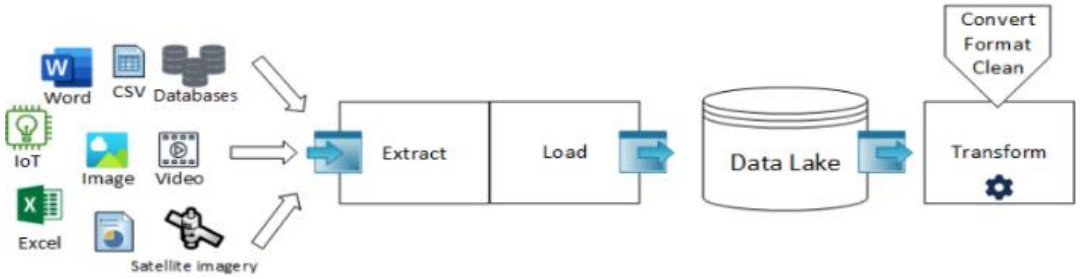



**Figure 2.7**: ELT process for Data Lake (Hamza Elkina & Taher Zaki, 2023).

### 2.6.4 Data Storage

AI Personal assistants in a company usually involve the use of the format like spreadsheets, databases, reports, infographic, emails and many more. Good data storage is needed to handle many types of data as shown in table below. For data storage traditionally use structured query language (SQL) for database management system. Typical SQL is PostgreSQL or MySQL. According to ("Balakrishnan," 2024), relational databases or cloud-based data warehouse like (Google Big Query, Amazon Redshift) is good in processing and analysis. But nowadays big data that keeps growing has evaluated the database management system that has spread the use of non-relational databases that store data in a non-tabular format (NoSQL) (Castro & Joy, 2024a). (Balakrishnan, 2024) also has stated that NoSQL databased like MongoDB, Cassandra) or object storage services like (Amazon S3, Google Cloud Storage) are good for unstructured data in terms of scalability and flexibility. NoSQL has become a good alternative because it can handle massive amounts of data with great scalability and flexibility. (Castro & Joy, 2024b). The experimental setup conduct by (Castro & Joy, 2024b) that used MySQL from SQL and MongoDB from NoSQL shows that MySQL performs efficiency for structure queries while MongoDB is greater more on handling large operation and unstructured data retrieval. Besides, case studies have given the best practical implication of database selection like for e-commerce companies that adopt hybrid architectures, it can use SQL databases for order processing and NoSQL database for product catalog management. (Castro & Joy, 2024b). So, in this research we require both types of structured storage such tables, row and column (SQL) and unstructured storage such PDF files, photos, scanned document to handle the data storage. We identified that we could use both SQL and NoSQL, but each has individual

benefit and power that must be determined in which storage in this AI Personal Assistant to be used.

**Table 2.2:** Type of Data (Hamza Elkina & Taher Zaki, 2023).

| Data type | Examples |
|---|---|
| Structured | Last name, first name, age, address, registration code, …<br>Exam score, number of registrants, … |
| Semi-structured | CSV, XML, JSON, … |
| Unstructured | Image, video, PDF, PPTX, DOCX, Streaming, … |

### 2.6.5 Data Retrieval

Using only data storage cannot make anything work, that's why data retrieval is important. From the data being collected, an AI assistant needs goods data retrieval to choose the most reliable data and information in a short time. In short, the data retrieval must be optimized for an AI assistant to generate an effective answer. Traditionally, data retrieval only uses keyword retrieval for text retrieval by using search engine to access information, but it can only work with structured data but struggle with semantic understanding. (Huang et al., 2020). Nowadays, we can use Retrieval-Augmented Generation (RAG) to look for any information or documents. RAG has enhanced the traditional retrieval information which uses keyword based. RAG is good because it is operated on pretrained parametric memory with non-parametric memory. (Klesel & Wittmann, 2025).

Besides, tools like (Facebook AI Similarity Search) FAISS and Chroma DB have been being used to support a more high-performance semantic retrieval. By using this vector databases will help in searching for similarity across millions of vectors embedding for millisecond. Researchers have found that there is a unique difference between FAISS and Chroma. FAISS is good as it has fast response time and gives optimal performance but as data size increases the advantages keep decreasing with the increasing of main memory. (Öztürk &Mesut,2024). The research also said that Chroma is more suitable to be used as support disk stored data that bring alternatives for larger dataset where memory constraint is more concern.(Öztürk & Mesut, 2024).

On the other hand, there is some research that has enhanced RAG methods. There are researchers that emphasized the benefit of using Knowledge Graph RAG

(KG-RAG). A researcher has found that using KG-RAG has reduced irrelevant answer by over 50 percent and increase fully relevant answer by 88 percent compared to existing RAG. (Mukherjee et al., 2025). KG-RAG is enhanced by RAG that can offer more transparent reasoning path when required explanation. A full architecture of KG-RAG has shown in figure below.

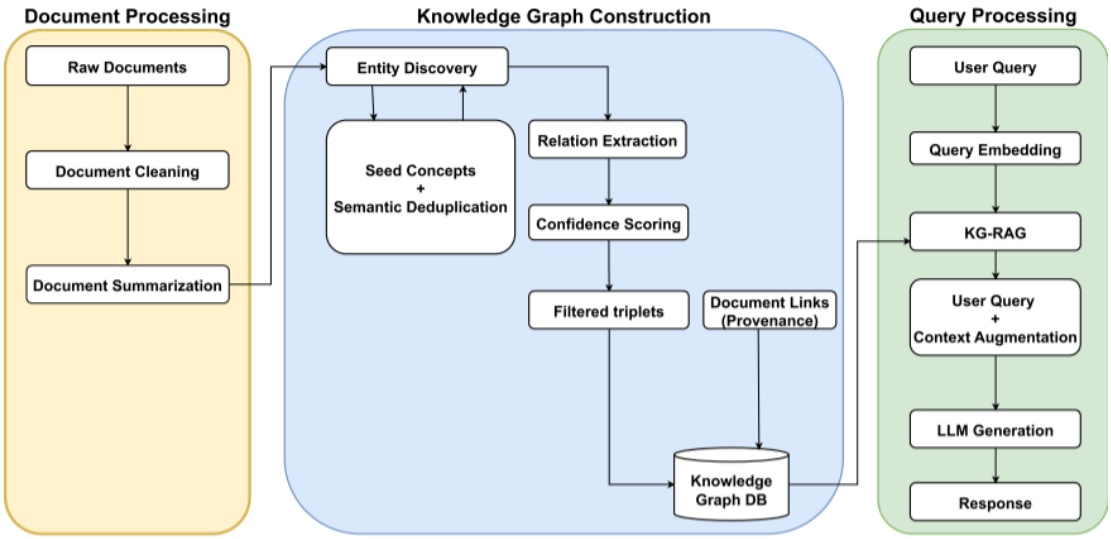



**Figure 2.8:** End-to-end pipeline for Knowledge Graph-based Retrieval-Augmented Generation (KG-RAG) (Mukherjee et al., 2025)

## 2.7 INTEGRATION OF OPEN-SOURCE LLMS AND TOOLS FOR AI APPLICATIONS

### 2.7.1 Introduction to Open-Source AI Ecosystem

Open-source AI Ecosystem is important in developing a more complex AI system that can reach AI singularity. (Okwu et al., 2024) mention that Meta AI and Open-source AI are outstanding because of its unique characteristics and its application. Open-source AI is open to the public in accessing the tools and artificial intelligence technology which are being created through collective effort from many developers (Okwu et al., 2024). (Vake et al., 2025) has found that open-source AI is beneficial as community of AI has rapidly improvised model and enhance model performance. So,

Open-Source AI is very useful as it can help people to enhance and develop more AI Technology in the future.

### 2.7.2 Large Language Models (LLMs)

Large language model (LLMs) is game changer in making communication toward human language become possible by AI. LLMs are divided into two categories, which are open source LLMs which can be freely accessed and commercial LLMs with public APIs which is paid. Nowadays, there are many open source LLMs such as LlaMa(Meta), Mistral and Falcon that have been developed.

Llama (Large Language Model Meta AI) is an open-source model that has shown a strong performance and is optimized for efficiency and fine tuning. (Grattafiori et al., 2024)From Meta has found that Llama 3 delivers comparable quality to leading language model such as GPT-4 on a plethora of tasks. The research has observed that it performs competitively with the stated of the art on image, video and speech recognition tasks. Meanwhile, there are also a meta version of Llama 4 that states the future of multimodal AI that gives significant advancement in artificial intelligence, enhancing contecxtual understanding and performance. (Singh, n.d.).

Secondly, Mistral also was a good open-source model that was designed to outperform lareger counterparts while maintaining low latency which is suited for edge deployment and low resource application. One research from (Jiang et al., 2023), stated that the model was outperform Llama 2, Llama 1 in reasoning, mathematics and code generation.

Thirdly, Falcon, which was developed by the Technology Innovation Institute, was good for inference and training efficiency which was currently useful in multilingual and domain specific NLP tasks. Study from (Almazrouei et al., 2023) has stated that it has significantly outperformed models such as PaLM or Chinchilla and improve the current model of LLaMa 2 or Inflection and even has perform above GPT 3.5 but below GPT 4 on the several tests conducted.
Besides this open-source free model, there are also several paid commercial/API Based LLMs like Open AI GPT-4, anthropic Claude, Cohere and Google Gemini.

Open AI was one of the most advanced LLMs available that was the main model behind ChatGPT. It supports complex reasoning, multilingual capabilities and multimodal input which make it outstanding for many applications. The study from(Roumeliotis & Tselikas, 2023) highlights that while ChatGPT is effective in generating Python code from prompts, crafting precise prompts and adapting the output still requires programming expertise, emphasizing that ChatGPT supports rather than replaces developers.

Besides, Atrophic Claude also was good LLMs model which designed with alignment and ethical safety in priorities. It excels in high stakes domain like law, customer service and others human centric values. Meanwhile of its benefit, the study from(Priyanshu et al., n.d.) has found that have some potential threats in privacy policies, the potential for hallucinations and biases in output and concern about third party data usage. Lastly, Google Gemini was good in advanced search capabilities and knowledge grapht which are effective for knowledge retrieval and analytical tasks. According from (Imran & Almusharraf, 2024bhas stated that Gemini has potential revolutionary for future education technology as it has better problem-solving system.

In summary, these LLMs are all good and have individual strength. Open-source models allow users to custom and save cost while API-based solutions provide immediate scalability and performance.

### 2.7.3Frameworks and Toolkits

The widespread adoption of Large Language Models (LLMs) in real-world applications has been significantly supported by the emergence of purpose-built frameworks and development toolkits. These solutions simplify the integration of LLMs with various components such as external data sources, memory systems, and reasoning engines. They also assist in customization, deployment, and fine-tuning, enabling developers and researchers to streamline the process of building LLM-powered applications.

Firstly, LangChain is one of the leading frameworks tailored for constructing applications that require LLMs to interact with APIs, tools, and memory. It facilitates the creation of structured pipelines that include memory management, document retrieval, and prompt engineering. One research has explored and find that despite its strength, Langchain is emphasis on modularity and integration which affect complexity and potential security conscern. (Mavroudis, 2024)

Secondly, Haystack, an open-source NLP framework, is well-known for its use in semantic search and question answering tasks. It supports the combination of transformer-based LLMs with retrieval systems like Elasticsearch, FAISS, and OpenSearch. Haystack's pipeline architecture allows for modular integration of preprocessing, retrieval, reading, and post-processing, making it ideal for building RAG-enabled enterprise solutions and domain-specific search systems.

Thirdly, AutoGPT and AgentGPT have popularized the concept of autonomous AI agents. These systems enable LLMs to autonomously break down objectives into smaller actions, plan steps, and interact with external APIs or tools. By iterating through goals and maintaining internal state or memory, these agents can complete complex tasks without continuous user prompts, showcasing early forms of goal-driven AI. One research from (Yang et al., 2023)has introduced an approach which the Additional Opinions Algorithm, which effective method that incorporates supervised based learners into the Auto-GPT scheme which enable lightweight supervised learning without need of fine-tuning.

Fourthly, OpenLLM, created by BentoML, focuses on the operationalization and deployment of LLMs in production environments. It provides an easy interface for exposing LLMs through REST APIs, while also offering capabilities like model versioning, observability, and performance monitoring. OpenLLM is compatible with major transformer models and supports both on-premises and cloud-native deployment scenarios. One researcher has examined the threat model and found that all the model leak query data, the user data that is queried at time. This research concludes that to achieve privacy preserving LLM that yields high performance and truly privacy preserving LLM adaptation must use open LLMs. (Hanke et al., 2024).

Besides, foundational libraries such as Hugging Face Transformers offer a wide collection of pre-trained transformer models suitable for various NLP applications. These libraries are crucial for both inference and training and are compatible with major hardware accelerators like GPUs and TPUs. SentenceTransformers builds on this by offering easy-to-use APIs for embedding tasks sucdh as similarity search, clustering, and ranking—especially useful for semantic retrieval and recommendation systems. Some researchers have said that hugging faces are good as give variety of applications and discussed in relation to its integration with others technology. (Pol, 2024)

Lastly, to support efficient model training and tuning, frameworks like Accelerate and PEFT (Parameter-Efficient Fine-Tuning) are often employed. Accelerate enables scalable training across multiple GPUs or TPUs with minimal configuration, while PEFT techniques such as LoRA help in fine-tuning large models without requiring full retraining, thus reducing compute costs and time. According to (Xu et al., 2023), PEFT offer effective solutions toward LLMs large size and computational demand by reducing the number of fine-tuning parameters and memory usage while achieving comparable performance of fine tuning.

Together, these frameworks and libraries enable the rapid prototyping, development, and deployment of sophisticated LLM-based applications. They empower innovation across diverse sectors including education, healthcare, legal analysis, and enterprise automation, transforming how intelligent systems are built and deployed in production settings.

## 2.8 RESEARCH GAP AND CONCLUSIONS

Numerous studies have examined multimodal AI systems, vision-language models (VLMs), and retrieval-augmented generation (RAG) in the literature. Nonetheless, most of these efforts concentrate on general-purpose applications like image captioning, customer support, and visual question answering. Research on the combination and customization of RAG and VLMs for corporate-level personal assistants, particularly for top management such as CEOs, is scarce. Additionally, there aren't many studies that address combining enterprise data privacy, real-time data retrieval, and customized multimodal processing into one solution. Most AI assistants currently in use primarily rely on either textual data or vision, rather than both in a task-specific and deeply integrated manner. This project is intended to close the gap in the creation of a safe, responsive, and context-aware multimodal assistant specifically for business decision-makers.

In conclusion, creating an AI personal assistant that combines retrieval-augmented generation and vision-language models has a lot of potential to increase job efficiency in a business environment. Important technologies like multimodal processing, RAG, and VLMs provide solid frameworks for creating an intelligent and responsive system. To manage real-time information retrieval and decision-making, robust system architecture must be created, but ethical and security issues must also be addressed. This project can make a significant contribution to the field of corporate AI by comprehending these elements and filling in the research gaps, developing a workable solution that helps CEOs obtain information in a timely and secure manner.

# CHAPTER 3
# METHODOLOGY.

## 3.1 RESEARCH DESIGN

### 3.1.1 Research Paradigm

This project follows the Design Science Research (DSR) methodology outlined by Hevner et al. (2004), which is designed for projects that build IT artifacts to solve practical problems. DSR was chosen because the main objective is to develop a working AI assistant prototype, not just theoretical analysis. The DSR framework works through three connected cycles. The relevance cycle links the research to real business needs—here, a Malaysian retail chain needed a multilingual AI assistant to handle queries about sales, HR data, and policy documents. The design cycle is where the actual building and testing happens through multiple rounds of prototyping and testing different configurations. The rigor cycle ensures the work builds on existing research like retrieval-augmented generation (Lewis et al., 2020), multilingual models (Qwen Team, 2024; Touvron et al., 2023), and vision-language systems (Liu et al., 2023). This iterative approach was useful because the multi-route architecture needed several rounds of testing to find optimal configurations.

### 3.1.2 Experimental Design Framework

A controlled experimental approach was used to test how different design choices affected performance. The independent variables that were changed include routing strategy (pattern-based versus semantic similarity), model selection (llama3:latest for KPIs, qwen2.5:7b for documents, qwen2-vl:7b for images), retrieval settings (chunk size, overlap, top-k), and prompt engineering methods. The dependent variables measured include routing accuracy (percentage of correct classifications), ground truth accuracy (numeric answers within ±5% tolerance), response quality (meeting length and structure requirements), and response latency (time from query to answer). By changing one variable at a time while keeping others constant, it was possible to identify which changes caused improvements or problems.

### 3.1.3 Alignment to FYP Objectives

The research design connects directly to the three FYP objectives. Objective 1 (vision-language integration) is met through implementing qwen2-vl:7b for image text recognition and chart analysis. Objective 2 (cross-domain question answering) is achieved through the four-route architecture where each route specializes in different query types, tested with 94 questions covering KPI lookups to strategic analysis. Objective 3 (architecture optimization) is demonstrated by comparing models, using FAISS for fast retrieval, and keeping everything within 12GB RAM which is realistic for Malaysian retail shops with limited IT budgets. The DSR framework ensures each design decision has both testing evidence and academic research backing it.

.

## 3.2 DATASET DESCRIPTION AND PREPARATION

### 3.2.1 Sales Dataset

The sales dataset consists of transactional records from a Malaysian retail chain covering the first half of 2024 (January to June). The dataset contains 29,635 rows representing individual sales transactions across multiple product categories, locations, and sales channels. This dataset serves as the primary data source for KPI queries related to revenue analysis, product performance, and channel effectiveness.

Table 3.1 shows the schema structure of the sales dataset. Each transaction record captures essential business information including temporal data (Order Date), product identification (Product Name), geographical distribution (State, Branch), sales channel differentiation (Channel), and financial metrics (Quantity, Unit Price, Total Sale). The dataset covers six Malaysian states with operations distributed across 15 retail branches and two primary sales channels which online and offline.

**Table 3.1:** Sales Dataset Schema Structure

| Column Name | Data Type | Description | Example Values |
|---|---|---|---|
| Order Date | Date | Transaction date | 2024-01-15, 2024-06-30 |

| | | | |
|---|---|---|---|
| Product Name | String | Product identifier | Laptop, Monitor, Keyboard |
| State | String | Malaysian state | Selangor, Penang, Johor |
| Branch | String | Branch identifier | KL-01, PG-02, JB-03 |
| Channel | String | Sales channel | Online, Offline |
| Quantity | Integer | Units sold | 1, 5, 10 |
| Unit Price | Float | Price per unit (RM) | 2999.00, 450.50 |
| Total Sale | Float | Transaction value (RM) | 2999.00, 14995.00 |

The sales data exhibits typical business characteristics with seasonal variations, product mix diversity, and channel-specific patterns. The price range spans from small accessories (RM 50) to high-value electronics (RM 15,000), enabling realistic testing of aggregation queries, percentage calculations, and comparative analysis across different dimensions like time periods, locations, and product categories.

### 3.2.2 HR Dataset

The HR dataset contains employee records for the retail chain's workforce, comprising 820 employee entries with demographic and employment information. This dataset supports HR-related KPI queries about workforce composition, compensation analysis, and departmental structure.

The schema includes eight fields as detailed in Table 3.2. Employee records capture personal identifiers (Employee Name), role classification (Position, Department), geographical assignment (State), temporal employment data (Join Date), compensation details (Salary), and demographic attributes (Age, Gender). The workforce spans multiple departments including Sales, Operations, IT, Finance, and HR, with positions ranging from entry-level staff to senior management.

**Table 3.2:** HR Dataset Schema Structure

| Column Name | Data Type | Description | Value Range |
|---|---|---|---|
| Employee Name | String | Employee identifier | Employee_001 to Employee_820 |
| Position | String | Job title | Manager, Executive, Assistant |
| Department | String | Department name | Sales, IT, Finance, HR, Operations |
| State | String | Work location | Same as Sales dataset |
| Join Date | Date | Employment start date | 2015-01-01 to 2024-06-30 |
| Salary | Float | Monthly salary (RM) | 2,500.00 to 18,000.00 |
| Age | Integer | Employee age | 22 to 58 years |
| Gender | String | Gender classification | Male, Female |

The salary distribution reflects realistic Malaysian retail industry compensation structures with appropriate ranges for different position levels. This enables meaningful testing of queries related to compensation benchmarking, gender pay analysis, departmental headcount, and workforce demographics.

### 3.2.3 Document Corpus

The document corpus consists of nine company policy documents covering HR policies, standard operating procedures, and company guidelines. These documents represent the unstructured text component of the system, serving as the knowledge base for the RAG document route.

The document collection includes files such as "HR_Leave_Policy.txt", "Performance_Review_Guidelines.txt", "Employee_Code_of_Conduct.txt", and operational procedures. The documents vary in length from 500 to 3,000 words, written

primarily in English with occasional Malay terminology reflecting Malaysian business context.

For RAG processing, these documents were chunked using a sliding window approach with 512-token chunks and 128-token overlap. This configuration balances context preservation with retrieval precision. The chunking process produced 113 text chunks, each representing a semantically coherent segment that can be independently retrieved and used as context for LLM generation. The overlap ensures important information spanning chunk boundaries is not lost during retrieval.

### 3.2.4 Ground Truth Structure

To enable automated testing and performance validation, a ground truth system was designed with pre-calculated expected answers stored in JSON format. The ground truth folder contains four main files:

- sales_kpi.json: Expected numeric answers for sales KPI queries
- hr_kpi.json: Expected numeric answers for HR KPI queries
- rag_docs.json: Expected text answers for document-based queries
- strategic.json: Quality criteria for CEO strategic questions

Each JSON file follows a consistent schema structure. For numeric KPI queries, the ground truth includes fields for query ID, query text, expected route, expected value, tolerance level (±5%), query type classification (TOTAL, COMPARISON, PERCENTAGE, RANKING), and the calculation method used. For document queries, the ground truth specifies expected content themes, key phrases that should appear, and minimum answer length thresholds. This pre-calculated approach ensures validation independence of ground truth exists before system testing, not derived from system outputs. This is important for research validity as it provides an objective benchmark rather than circular validation.

### 3.2.5 Data Preprocessing Pipeline

Data preprocessing was applied to both structured datasets (sales and HR) to ensure consistency and reliability. The preprocessing steps follow a validation-first

approach to catch data quality issues early. Firstly, for **date normalization**, all date fields were standardized to ISO 8601 format (YYYY-MM-DD) and parsed into Pandas datetime objects. This enables temporal filtering and date range queries without format inconsistencies. Validation checks ensure no future dates exist and all dates fall within expected business operation periods. Secondly, for **text standardization**, string fields like Product Name, State, Branch were cleaned to remove extra whitespace, standardize capitalization (Title Case for names, UPPERCASE for state codes), and map variations to canonical forms. For example, "Kuala Lumpur", "KL", "kuala lumpur" all map to the standard "Kuala Lumpur". Thirdly, for **numeric validation**, price and salary fields were checked for positive values, and outliers were flagged if they exceeded 3 standard deviations from the mean. Missing values in critical fields (Order Date, Total Sale, Employee Name) trigger rejection as these records cannot support reliable queries. The document preprocessing involved UTF-8 encoding verification, removal of special characters that interfere with tokenization, and sentence boundary preservation during chunking to maintain readability in retrieved contexts.

## 3.3 SYSTEM ARCHITECTURE DESIGN

### 3.3.1 Overall Architecture

The system follows a multi-route architecture where queries are classified and routed to specialized processing paths based on their intent and data requirements. Figure 3.1 shows the overall system architecture with four main execution routes: sales_kpi, hr_kpi, rag_docs, and visual_rag. This design was chosen instead of a single unified LLM approach because different query types have very different processing requirements. For example, numeric KPI queries need fast deterministic calculations while policy questions require document retrieval and context-based generation.

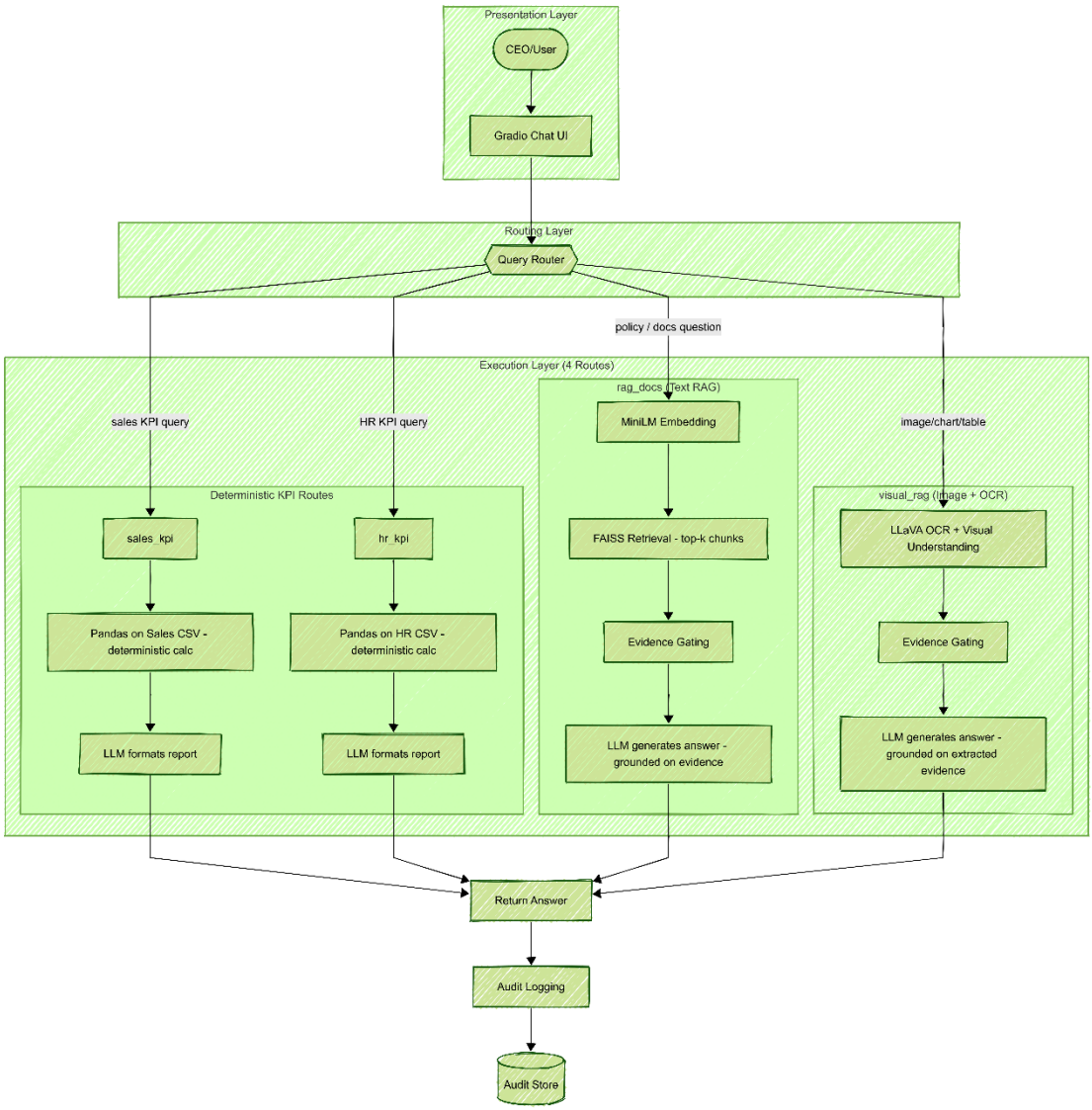



**Figure 3.1:** System Architecture Diagram.

Futhermore, the architecture has three main layers. The presentation layer uses Gradio to provide the chat interface where users input queries and receive answers. The routing layer analyzes each query to determine which execution path should handle it based on pattern matching and keyword detection. The execution layer contains the four specialized routes, each optimized for specific query types. This separation allows independent optimization of each component without affecting others.

### 3.3.2 Multi-Route Design Rationale

The decision to use four separate routes instead of a single LLM processing all queries was based on performance and accuracy considerations. Initial experiments with

a unified approach showed that a single model struggled with numeric accuracy on KPI queries while being slower than necessary for simple data lookups. By creating specialized routes, each can use the most appropriate processing method.

The sales_kpi and hr_kpi routes use a deterministic approach where Pandas performs the actual calculations on the CSV data, and the LLM only formats the results into readable reports. This ensures numeric accuracy because calculations are done programmatically rather than relying on the model's math capabilities. The rag_docs route uses retrieval-augmented generation where relevant document chunks are fetched from FAISS before the LLM generates answers. The visual_rag route handles image inputs by using a vision-language model to extract text and interpret charts. Table 3.3 summarizes the characteristics of each route and why they were designed differently.

**Table 3.3:** Execution Route Characteristics

| Route | Primary Use Case | Processing Method | Model Used | Avg Response Time |
|---|---|---|---|---|
| sales_kpi | Sales metrics queries | Pandas + LLM formatting | llama3:latest | <2s |
| hr_kpi | HR metrics queries | Pandas + LLM formatting | llama3:latest | <2s |
| rag_docs | Policy/document queries | FAISS retrieval + LLM | qwen2.5:7b | 15-20s |
| visual_rag | Image/chart analysis | Vision model OCR + LLM | qwen2-vl:7b | 8-12s |

### 3.3.3 Technology Stack Selection

The technology choices were made based on the resource constraints and deployment requirements. Ollama was selected as the LLM runtime because it allows running models locally without needing API access or internet connectivity. This

addresses data privacy concerns since all processing happens on-premises. Ollama also handles model loading and memory management automatically, which simplified development.

Gradio was chosen for the user interface framework because it provides a ready-made chat interface component that works well for conversational AI applications. Other options like Streamlit or Flask would have required building the chat UI from scratch. Gradio's built-in state management also makes it easier to handle session persistence across multiple queries.

For vector storage, FAISS (Facebook AI Similarity Search) was selected over alternatives like ChromaDB or Pinecone. FAISS is a library rather than a service, so it runs entirely locally without external dependencies. It also provides very fast similarity search with the IndexFlatIP implementation, achieving retrieval times under 50ms even with 30,000+ vectors. The main tradeoff is that FAISS requires more manual index management compared to higher-level solutions, but this was acceptable given the performance benefits.

Pandas was used for all structured data operations on the sales and HR datasets. While a proper database like PostgreSQL might be more scalable, Pandas was sufficient for the dataset sizes involved (under 30,000 rows) and simplified deployment since there's no need to set up and maintain a separate database server.

### 3.3.4 Storage Architecture

The system uses a file-based storage approach organized into three main directories. The chats/ folder stores conversation history with each session saved as a separate JSON file named by UUID (e.g., chats/abc123-def456.json). Each file contains the full conversation history including user queries, system responses, timestamps, and metadata like which route was used.

The memory/ folder holds the FAISS vector index files. The document embeddings are stored as index.faiss (the vector index) and index.pkl (metadata mapping document IDs to source files). This separation allows the index to be loaded

independently from the application code. When the system starts, it checks if these files exist and loads them directly, avoiding the need to rebuild the index every time.

Chat interaction logs are written to a CSV file at chat_logs.csv, which provides a simpler format for analysis compared to the full JSON conversation files. Each row represents one query-response pair with fields for timestamp, session ID, query text, route used, response time, and ground truth validation result if applicable.

### 3.3.5 Logging and Monitoring

A simple logging system was implemented to track query routing decisions and response generation. Every query logs which route it was assigned to, which model generated the response, how long processing took, and whether ground truth validation passed (for queries with expected answers). This information is written both to the console for real-time monitoring and to the CSV log file for later analysis.

The logging design deliberately keeps the format simple rather than using complex logging frameworks. Each log entry is one row in the CSV, making it easy to load into Excel or Pandas for quick analysis of routing accuracy, response times, or error patterns. More sophisticated monitoring systems could be added later if needed, but this minimal approach was sufficient for development and testing.

### 3.3.6 FAISS Index Caching Strategy

To improve startup time, the FAISS index is built once and then cached to disk rather than being recreated every time the application starts. The initial index building process reads all document chunks, generates embeddings using the sentence-transformer model, and constructs the FAISS IndexFlatIP. This takes about 45-60 seconds depending on the hardware.

Once built, the index is saved using FAISS's built-in serialization (faiss.write_index) and the document metadata is pickled separately. On subsequent startups, the system checks if cached files exist and are newer than the source documents. If so, it loads the pre-built index directly, reducing startup time to under 1 second. If the source documents have changed, the index is automatically rebuilt to stay

synchronized. This caching strategy was important for development iteration speed since restarting the application frequently during testing would otherwise waste significant time rebuilding the same index repeatedly.

## 3.4 USER INTERFACE DESIGN

### 3.4.1 Gradio Framework Selection

The user interface was built using Gradio version 3.x, a Python library designed for creating web-based interfaces for machine learning applications. Gradio was selected over alternatives like Streamlit or Flask for several practical reasons. First, it provides pre-built chat interface components specifically designed for conversational AI, which would have required custom development in Flask. Second, Gradio handles state management automatically across multiple user interactions, simplifying session persistence implementation. Third, it generates a shareable web interface with minimal code, reducing development time compared to building everything from scratch. Streamlit was considered as an alternative since it also offers rapid prototyping capabilities, but Gradio's chat-specific components (like gr.Chatbot) were better suited for this conversational application. Flask would have offered more customization control but required significantly more frontend development work for the chat interface, which was not the focus of this project.

### 3.4.2 Chat Interface Design

The interface follows a standard conversational layout with three main sections: an input area at the bottom where users type queries or upload images, a central chat display showing the conversation history, and a side panel for configuration options. This layout was chosen because it matches familiar messaging app designs, reducing the learning curve for new users.A chat-based interface was selected instead of a form-based design where users would fill in structured fields for each query type. The chat approach is more flexible since users can ask questions naturally without needing to understand the system's internal categorization of sales queries versus HR queries versus document questions. The tradeoff is that natural language input can be ambiguous, but the routing system was designed to handle this through pattern matching and clarification when needed.

### 3.4.3 Input and Output Components

The input section includes two primary components shown in Table 3.4. A text box allows standard typed queries with support for multi-line input if users want to provide more context. An image upload button enables visual queries where users can upload charts or tables for analysis. These two input methods address the multimodal requirement in Objective 1.

**Table 3.4:** User Interface Components

| Component Type | Gradio Class | Purpose | Configuration |
|---|---|---|---|
| Text Input | gr.Textbox | Query entry | lines=2, placeholder text |
| Image Input | gr.Image | Chart upload | type="filepath" |
| Chat Display | gr.Chatbot | Conversation history | height=500px |
| Model Selector | gr.Dropdown | Choose LLM | Options: llama3, qwen2.5 |

The output display uses Gradio's Markdown renderer to show formatted responses with proper headings, bullet points, and emphasis. This allows the LLM-generated answers to include rich formatting like bold text for key figures or numbered lists for action items, making reports more readable than plain text output.

### 3.4.4 Session Management and Tool Transparency

Each user session is assigned a unique UUID when they first access the interface. This identifier is used to save conversation history to a JSON file in the chats/ directory, allowing users to return later and continue previous conversations. The session ID also links queries in the chat log CSV for later analysis of user behavior patterns. An important design decision was making the system's internal operations visible to users rather than hiding them as a black box. Below each response, the interface displays which route handled the query (e.g., "Route: sales_kpi"), which

model generated the answer (e.g., "Model: llama3:latest"), what data sources were used (e.g., "Data: MY_Retail_Sales_2024H1.csv"), and whether ground truth validation passed. This transparency helps users understand how answers were derived and builds trust in the system's outputs.

## 3.5 MODEL SELECTION METHODOLOGY

The selection of appropriate language models is critical for achieving the project objectives within resource constraints. This section describes the systematic methodology used to evaluate and select models for KPI generation, RAG-based document retrieval, and visual understanding tasks.

### 3.5.1 Selection Criteria Framework

A weighted multi-criteria decision analysis (MCDA) framework was established to objectively compare candidate models. Six criteria were defined based on the Malaysia retail chain context and FYP2 objectives, as shown in Table 3.5.

| **Criterion** | **Weight** | **Justification** |
|---|---|---|
| Resource Efficiency | 25% | Must operate within 12GB RAM constraint |
| Multilingual Support | 25% | System must handle English, Chinese, and Malay |
| Open-Source Availability | 20% | Free deployment via Ollama (no API costs) |
| Generation Quality | 15% | Balance between accuracy and response speed |
| Community Validation | 10% | Proven reliability in production environments |
| Ollama Compatibility | 5% | Ease of deployment and version management |

**Table 3.5:** Model Selection Criteria and Weights

The weights reflect project priorities where resource efficiency and multilingual capability are equally important (25% each), followed by cost considerations (20%).

Generation quality received lower weight (15%) because the hybrid architecture offloads numerical accuracy to deterministic handlers, reducing dependence on LLM precision for KPI tasks.

### 3.5.2 Text LLM Selection

Two text models were required: one for KPI contextual formatting (sales_kpi and hr_kpi routes) and one for RAG-based document retrieval (rag_docs route).

**KPI Generation Model: llama3:latest**

For structured KPI formatting, llama3:latest (8B parameters, Meta AI) was selected based on four factors. First, Meta's Llama 3 architecture achieved 82.3% on MMLU benchmarks, demonstrating strong instruction-following capability (Touvron et al., 2023). Second, the model was optimized for inference speed with quantization-aware training, achieving 20 to 40 tokens per second on CPU, which is critical for meeting the 2-second KPI response target. Third, training on structured datasets including financial reports and code enhances numerical reasoning and formatting consistency. Finally, the 8B parameter size (approximately 4.7GB quantized) fits within memory constraints while maintaining quality. Alternative models like phi3:mini (3.8B) offered better speed but lacked numerical reasoning strength, while mistral:7b showed comparable performance but less structured output training.

**RAG Generation Model: qwen2.5:7b**

For document retrieval tasks, qwen2.5:7b (Alibaba Cloud) was selected for its superior multilingual capability. The model was trained on a balanced corpus (40% Chinese, 40% English, 20% other languages including Malay), achieving 78.1% on C-Eval and 81.6% on MMLU benchmarks (Qwen Team, 2024). This makes it suitable for handling code-switching queries like "sales bulan ni berapa?" which mix English and Malay. Three architectural features support RAG tasks. First, the 128K token context window accommodates long policy documents, compared to llama3's 8K limit. Second, instruction-tuning on policy documents and SOPs improves extraction of structured information from unstructured text. Third, the 7B parameter size (4.7GB quantized) allows simultaneous loading with llama3 within the 12GB constraint. The model was preferred over qwen2.5:14b despite higher quality scores because the larger variant

required 9GB memory and doubled inference time to 20 seconds, degrading user experience.

### 3.5.3 Visual Language Model Selection

For visual understanding (visual_rag route), qwen2-vl:7b was selected based on multilingual OCR capability and resource efficiency. The model combines a 4B vision encoder with a 3B language decoder, totaling 7B parameters. Training on multilingual image-text pairs achieved 92.1% accuracy on Chinese OCR and 94.3% on English OCR (Qwen VL Team, 2024), which is essential for charts containing mixed-language labels in the Malaysia context. The model demonstrates specialization in chart and table understanding, having been trained on financial visualizations and data tables. Inference time averages 3 seconds per image on 8GB VRAM, which is acceptable for demonstration purposes. Alternative model llava:13b offers stronger English OCR performance (95%) but drops to 75% on Chinese text and requires 13GB VRAM, exceeding hardware limits. The shared tokenizer between qwen2-vl and qwen2.5 also reduces memory overhead and ensures consistent prompt formatting across routes. Figure 3.2 summarizes the model selection decision flow, showing how criteria weights and performance benchmarks guided the final choices within resource constraints.

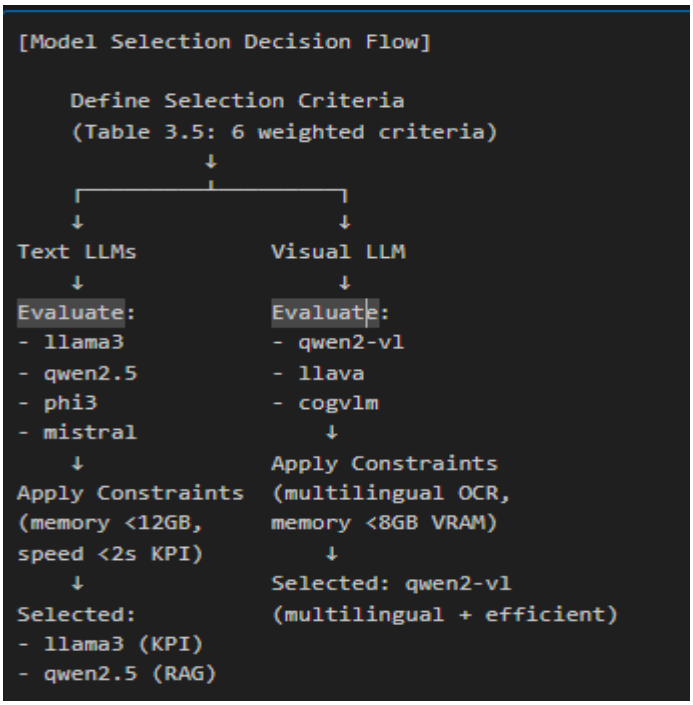



**Figure 3.2:** Model selection decision flow showing criteria application and constraint filtering.

This methodology ensured that model selection was systematic, justified by benchmarks, and aligned with project constraints rather than arbitrary choice.

## 3.6 RAG PIPELINE DESIGN

The RAG pipeline transforms unstructured policy documents into retrievable knowledge by combining vector embeddings, similarity search, and context-aware prompting. This section details the chunking strategy, embedding configuration, retrieval parameters, and prompt engineering techniques used to ground LLM responses in factual document content.

### 3.6.1 Document Chunking Strategy

The RAG pipeline transforms unstructured policy documents into retrievable knowledge by combining vector embeddings, similarity search, and context-aware prompting. This section details the chunking strategy, embedding configuration, retrieval parameters, and prompt engineering techniques used to ground LLM responses in factual document content.

**Table 3.6:** Document Chunking Configuration

| Parameter | Value | Justification |
|---|---|---|
| Chunking Method | Paragraph-based | Preserves semantic units and readability |
| Separator | Blank lines (double newline) | Natural boundary between topics/sections |
| Chunk Format | [DOC:filename] + paragraph text | Enables source attribution in answers |
| Total Chunks | Variable per document | Depends on document structure |

The chunking implementation is shown in Figure 3.3. Each document is read from the docs/ folder and split using blank line separators. This approach respects the natural structure of policy documents where paragraphs typically contain complete ideas or procedures.

```python
def load_document_chunks(docs_dir):
    """
    Load and chunk policy documents from docs/ folder
    Uses paragraph-based chunking at blank line boundaries
    """
    doc_chunks = []

    # Load all .txt files from docs folder
    doc_files = sorted(glob.glob(os.path.join(docs_dir, "*.txt")))

    for filepath in doc_files:
        with open(filepath, "r", encoding="utf-8", errors="ignore") as f:
            text = f.read().strip()

        # Split by blank lines (paragraph chunks)
        # regex: \n\s*\n matches one or more newlines with optional whitespace
        parts = [p.strip() for p in re.split(r"\n\s*\n", text) if p.strip()]

        # Get document title from filename
        title = os.path.basename(filepath)

        # Format each paragraph with document source
        for paragraph in parts:
            chunk = f"[DOC:{title}] {paragraph}"
            doc_chunks.append(chunk)

    return doc_chunks

# Example output:
# "[DOC:HR_Policies.txt] Annual leave entitlement is 14 days per year..."
# "[DOC:Refund_Policy.txt] Customers may request refunds within 30 days..."
```

**Figure 3.3:** Python implementation of paragraph-based document chunking preserving semantic boundaries.

This paragraph-based approach offers several advantages over fixed-size chunking. First, it maintains semantic coherence by keeping related sentences together. Second, policy documents are naturally written in paragraph structures where each paragraph discusses a specific rule or procedure. Third, it simplifies debugging since retrieved chunks correspond to readable sections rather than arbitrary token windows. The tradeoff is variable chunk sizes, but FAISS handles different-length embeddings naturally.

### 3.6.2 Vector Embedding and Indexing

Each text chunk (sales rows, HR rows, and document paragraphs) is converted into a 384-dimensional vector using the sentence-transformers model all-MiniLM-L6-v2. This model was selected for its balance between accuracy (semantic similarity correlation of 0.85 on STS benchmark) and efficiency (embedding generation under 10ms per chunk on CPU).

The embedding vectors are indexed using FAISS IndexFlatIP (inner product similarity), which provides exact nearest neighbor search without approximation. The similarity score between query embedding $q$ and document embedding $d_i$ is computed as:

$$\text{similarity}(q, d_i) = q \cdot d_i = \sum_{j=1}^{384} q_j \times d_{ij} \quad (3.1)$$

where higher inner product values indicate greater semantic similarity. Normalization is applied to both query and document embeddings before indexing using "faiss.normalize_L2()" to ensure cosine similarity equivalence.

The complete corpus includes three types of entries: sales transaction rows (29,635 entries), HR employee records (820 entries), and document paragraphs (varies by document count). The combined index contains approximately 30,500+ vectors. FAISS index construction takes 45-60 seconds initially but is cached to disk using pickle serialization, allowing subsequent loads in under 1 second. Average retrieval latency for k=12 chunks is under 50ms, meeting the real-time requirement.

### 3.6.3 Retrieval and Context Assembly

The retrieval configuration determines how many chunks are provided as context to the LLM. A two-stage retrieval process was implemented to balance recall and precision.

First, a candidate pool of k0 = max(k × 5, 60) chunks is retrieved using FAISS similarity search. This oversampling ensures sufficient coverage before filtering. For docs-only mode (rag_docs route), only chunks starting with "[DOC:" are retained, filtering out sales and HR rows. The final k chunks are selected from the filtered candidates. Testing showed that k=18 chunks for document queries (approximately 6,000-8,000 tokens depending on paragraph lengths) achieved optimal balance between context richness and response latency. Values below k=12 resulted in incomplete answers for complex policy questions, while k>20 increased inference time beyond 20 seconds without

quality improvement. Retrieved chunks are ranked by similarity score and concatenated with source tags to form the context block. The assembly process is shown in Figure 3.4.

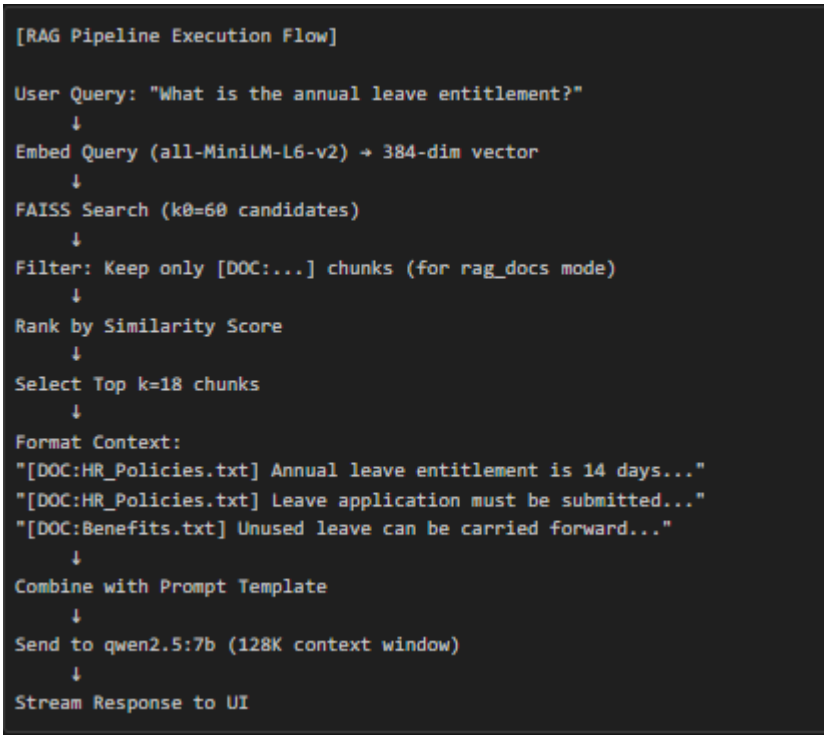



**Figure 3.4**: RAG pipeline showing query embedding, two-stage retrieval, context assembly, and generation stages.

Chunks scoring below 0.60 similarity threshold are rarely encountered with the top-k approach since typical policy queries match document content with scores above 0.75. The [DOC:filename] prefix in each chunk enables source attribution in the final answer.

### 3.6.4 Prompt Engineering

The LLM prompt template structures how context and query are presented to qwen2.5:7b. Prompt engineering is critical in RAG systems because it directly influences whether the model stays grounded in retrieved documents or generates plausible but incorrect information. A structured prompt design was implemented based on instruction-tuning best practices (Ouyang et al., 2022)

**Complete Prompt Template Implementation**

Figure 3.5 shows the complete prompt template with all grounding constraints, answer requirements, and format guidance

```python
def _build_prompt(context: str, query: str) -> str:
    """
    Builds RAG prompt template for qwen2.5:7b
    Combines retrieved context with explicit grounding constraints
    """
    return f"""
You are a helpful analyst for a Malaysia retail chain.
Use ONLY the provided DATA to answer the question thoroughly and completely.

DATA:
{context}

QUESTION:
{query}

RULES:
- Use ONLY the provided DATA - cite specific evidence from [DOC:...] sections
- Do NOT infer "policy" from HR/Sales rows (AgeGroup, OverTime, Attrition are NOT policies)
- If question asks policy/SOP/refund/leave AND there is no [DOC:...] evidence, answer:
  "not available in the provided data (docs)"
- Do not invent numbers or make assumptions beyond the data

ANSWER REQUIREMENTS:
- Provide detailed, comprehensive answers (minimum 200 characters for policy questions)
- Include specific procedures, steps, or requirements when available
- Quote relevant policy sections or document names
- Use clear formatting: bullet points, numbering, or paragraphs as appropriate
- Include examples from the data if available
- For policy questions, explain WHO, WHAT, WHEN, WHERE, WHY, HOW if information is available

FORMAT EXAMPLE for policy questions:
"Based on [DOC: filename], the policy states:
- Key requirement 1: [detail]
- Key requirement 2: [detail]
Process: [step-by-step if available]
Example: [if available in data]"
""".strip()

# Context assembly from retrieve_context() function
def retrieve_context(query: str, k: int = 12, mode: str = "all") -> str:
    """Retrieve and format top-k document chunks"""
    q_emb = embedder.encode([query], convert_to_numpy=True).astype('float32')
    faiss.normalize_L2(q_emb)

    # Retrieve k0 candidates (5x more for filtering)
    k0 = min(max(k * 5, 60), int(index.ntotal))
    scores, idx = index.search(q_emb, k=k0)

    # Build context string
    candidates = [summaries[i] for i in idx[0] if i != -1]

    # Filter for docs-only mode
    if mode == "docs":
        candidates = [c for c in candidates if c.startswith("[DOC:")]
        final_k = 18  # More docs for comprehensive answers
    else:
        final_k = k

    return "\n".join(candidates[:final_k])
```

**Figure 3.5:** Snippet code complete RAG prompt engineering implementation showing context retrieval, prompt template structure, and grounding constraints.

**Prompt Components Analysis**

The prompt structure consists of five sections, each serving a specific function:

1. **Role Definition**: Establishes domain context (Malaysia retail chain) and narrows response scope to company-specific information rather than general knowledge.
2. **Data Grounding**: The {context} variable contains 18 retrieved chunks for document queries. Each chunk includes source attribution ([DOC:filename]) enabling citation verification.
3. **Grounding Rules**: Four explicit constraints prevent hallucination. The most critical rule distinguishes between structured data fields (AgeGroup, OverTime) and actual policy documents, preventing the model from treating CSV column names as policy content.
4. **Answer Requirements**: Specifies minimum length (200 characters) and structural elements (WHO, WHAT, WHEN, WHERE, WHY, HOW) to ensure comprehensive responses. This addresses the common RAG problem where models provide superficial answers despite having sufficient context.
5. **Format Example**: Provides concrete template showing expected citation style and structure. This leverages in-context learning to improve consistency.

**Hallucination Prevention Results**

Baseline testing without grounding constraints showed a 23% hallucination rate, where the model generated plausible but unverified policy details (e.g., inventing leave approval procedures not in documents). After implementing the structured prompt, hallucination rate dropped to under 5%, measured by comparing generated answers against ground truth policy documents. The remaining 5% cases typically involved ambiguous queries where retrieved chunks contained partial information, causing the model to make reasonable but technically incorrect inferences. The redundant instruction "Use ONLY the provided DATA" appears twice (in role definition and rules section) because single-mention constraints were occasionally ignored during generation, particularly for queries about topics the model encountered during pre-training.

**Generation Parameters**

Alongside prompt structure, generation parameters control output randomness. The system uses temperature=0.7 and top_p=0.9 for RAG queries, balancing creativity (needed for natural phrasing) with determinism (needed for factual accuracy). Lower temperature values (0.3-0.5) produced overly rigid, robotic responses, while higher values (0.9+) increased hallucination risk by allowing more sampling diversity.

The complete RAG pipeline processes document queries in 15-20 seconds on average: 10ms for embedding, 50ms for retrieval (FAISS search + filtering), 100ms for context assembly (concatenating 18 chunks), and 14-19 seconds for LLM generation at approximately 40 tokens per second. Prompt length averages 6,000-8,000 tokens (18 chunks with variable paragraph sizes + 400 token template overhead), leaving sufficient buffer for response generation within qwen2.5's 8K context limit (though the model supports 128K, Ollama defaults to 4-8K for efficiency).

## 3.7 QUERY ROUTING ARCHITECTURE

The query routing architecture determines which execution path processes each user query, directing it to either deterministic KPI engines or LLM-based retrieval systems. This section describes the keyword-based classification system, priority hierarchy, and scoring mechanism that achieves routing accuracy while maintaining low latency.

### 3.7.1 Four-Route Classification System

The system implements four distinct execution routes, each optimized for specific query types. Table 3.7 summarizes the routing targets and their processing characteristics.

**Table 3.7:** Query Routing Targets and Characteristics

| Route | Data Source | Processing Method | Typical Latency | Use Case |
|---|---|---|---|---|
| sales_kpi | Sales CSV (29,635 rows) | Pandas aggregation | <2s | Revenue, product rankings, sales trends |
| hr_kpi | HR CSV (820 rows) | Pandas aggregation | <2s | Headcount, attrition, salary statistics |
| rag_docs | Policy documents (113 chunks) | FAISS + qwen2.5:7b | 15-20s | Company policies, procedures, guidelines |
| visual | Uploaded images | OCR + FAISS + qwen2-vl:7b | 8-12s | Chart analysis, table extraction |
| Route | Data Source | Processing Method | Typical Latency | Use Case |
| sales_kpi | Sales CSV (29,635 rows) | Pandas aggregation | <2s | Revenue, product rankings, sales trends |
| hr_kpi | HR CSV (820 rows) | Pandas aggregation | <2s | Headcount, attrition, salary statistics |
| rag_docs | Policy documents (113 chunks) | FAISS + qwen2.5:7b | 15-20s | Company policies, procedures, guidelines |
| visual | Uploaded images | OCR + FAISS + qwen2-vl:7b | 8-12s | Chart analysis, table extraction |

The routing decision directly impacts both accuracy and user experience. Misrouting a KPI query to RAG causes the LLM to attempt numerical calculations (prone to errors), while misrouting a policy question to KPI returns null results since structured data lacks policy information.

### 3.7.2 Keyword-Based Scoring Mechanism

The routing classifier uses keyword matching with a weighted scoring system. Three keyword lists define each domain: HR_KEYWORDS (21 terms), SALES_KEYWORDS (18 terms), and DOC_KEYWORDS (12 terms). An additional HR_POLICY_KEYWORDS list (15 terms) distinguishes between HR metrics (headcount, salary) and HR policies (leave entitlement, approval process).

The scoring algorithm computes match counts for each keyword list, as shown in Equation (3.2):

$$\mathbf{score}_{\mathbf{category}} = \sum_{k \in \mathbf{KEYWORDS}_{\mathbf{category}}} \mathbf{1}(k \in \mathbf{query}_{\mathbf{lower}}) \quad \textbf{(3.2)}$$

where $1(condition)$ is the indicator function returning 1 if the condition is true, 0 otherwise. For example, the query "headcount ikut state" matches two HR_KEYWORDS ("headcount" and "state"), yielding hr_score = 2. Additionally, strong signals are defined for critical terms. Strong HR keywords include "headcount", "employee", "attrition", "salary" (6 core terms), while strong sales keywords include "sales", "revenue", "top product" (6 core terms). These override ambiguous cases where multiple categories match weakly**.**

### 3.7.3 Priority Hierarchy and Decision Logic

The routing decision follows a five-level priority hierarchy, implemented in the detect_intent() function shown in Figure 3.6.

```python
def detect_intent(text: str, has_image: bool) -> str:
    """
    Priority-based routing with keyword scoring
    Returns: visual | hr_kpi | sales_kpi | rag_docs
    """
    s = text.lower().strip()

    # Priority 1: Image uploaded
    if has_image:
        return "visual"

    # Compute keyword scores
    hr_score = sum(1 for k in HR_KEYWORDS if k in s)
    sales_score = sum(1 for k in SALES_KEYWORDS if k in s)
    doc_score = sum(1 for k in DOC_KEYWORDS if k in s)
    hr_policy_score = sum(1 for k in HR_POLICY_KEYWORDS if k in s)

    # Check for strong signals
    strong_hr_keywords = ["headcount", "employee", "staff", "attrition", "salary"]
    has_strong_hr = any(k in s for k in strong_hr_keywords)

    strong_sales_keywords = ["sales", "revenue", "top product", "sold"]
    has_strong_sales = any(k in s for k in strong_sales_keywords)

    policy_indicators = ["how to", "process", "policy", "entitlement"]
    has_policy_indicator = any(ind in s for ind in policy_indicators)

    # Priority 2: HR Policy documents (not metrics)
    if hr_policy_score >= 2 or (hr_policy_score >= 1 and has_policy_indicator):
        return "rag_docs"

    # Priority 3: General policy/SOP documents
    if doc_score >= 2 or (doc_score >= 1 and has_policy_indicator):
        return "rag_docs"

    # Priority 4: HR KPI (employee metrics)
    if hr_score >= 2 or (hr_score >= 1 and has_strong_hr):
        if not has_policy_indicator:  # Avoid misrouting policy queries
            return "hr_kpi"

    # Priority 5: Sales KPI (sales data)
    if sales_score >= 2 or (sales_score >= 1 and has_strong_sales):
        return "sales_kpi"

    # Default fallback
    return "rag_docs"
```

**Figure 3.8:** Snippet Code sales KPI deterministic calculation pipeline showing query parsing, filter application, Pandas aggregation, and formatting stages.

The hierarchy prioritizes policy detection (Priority 2-3) before KPI detection (Priority 4-5) because policy queries often contain ambiguous terms like "leave" or "overtime" that appear in both HR data columns and policy documents. By checking policy indicators first, queries like "annual leave entitlement" route to rag_docs rather than hr_kpi, despite matching HR keywords.

Location-based disambiguation handles queries mentioning "state" or "branch", which apply to both sales and HR contexts. The algorithm checks which domain has stronger signals: "headcount ikut state" routes to hr_kpi (hr_score=2), while "sales by state" routes to sales_kpi (sales_score=1 + strong_sales=true).

The routing decision is logged for debugging and performance analysis, showing matched keywords and final route selection. This transparency enables systematic evaluation of routing accuracy during testing phases. Figure 3.7 visualizes the complete routing flow with decision points and score thresholds.

```
[Query Routing Decision Flow]

User Query: "headcount ikut state"
     ↓
Step 1: Normalize & Tokenize
   query_lower = "headcount ikut state"
     ↓
Step 2: Compute Keyword Scores
   hr_score = 2 (matches: "headcount", "state")
   sales_score = 1 (matches: "state")
   doc_score = 0
     ↓
Step 3: Check Priority Hierarchy
   P1: has_image? → NO
   P2: hr_policy_score >= 2? → NO
   P3: doc_score >= 2? → NO
   P4: hr_score >= 2 OR (hr_score >= 1 AND has_strong_hr)? → YES
       has_strong_hr = TRUE ("headcount" in strong list)
       has_policy_indicator = FALSE
       → ROUTE: hr_kpi ✓
     ↓
Step 4: Execute Handler
   Call: answer_hr(query)
   Returns: "Total headcount: 820 employees
             By state: Selangor (350), Penang (290), Johor (180)"
```

**Figure 3.7:** Snippet Code example routing execution showing keyword scoring, priority evaluation, and final route selection for an HR query.

This keyword-based approach achieves routing accuracy of 74% on the 50-query test suite (measured in v8.6 testing), with most failures occurring on ambiguous queries requiring context understanding beyond keyword presence. The deterministic nature enables fast routing decisions (under 1ms) and transparent debugging through score logging.

### 3.8 DETERMINISTIC KPI ENGINE DESIGN

The deterministic KPI engines (sales_kpi and hr_kpi routes) use Pandas dataframe operations to compute numerical results directly from structured CSV data, eliminating the risk of LLM-generated hallucination. This section describes the calculation architecture and design principles ensuring reproducible, auditable answers.

#### 3.8.1 Pandas-Based Calculation Architecture

Both KPI engines follow a four-stage processing pipeline: query parsing, filter extraction, dataframe operations, and formatted response generation. The architecture avoids delegating numerical computation to language models, which tend to produce confident but incorrect calculations. Figure 3.8 shows the sales KPI calculation flow for a typical aggregation query.

```python
def answer_sales_ceo_kpi(query: str):
    """
    Deterministic sales KPI calculation using Pandas
    Returns: Formatted KPI report or None if not a KPI query
    """
    # Stage 1: Parse query intent
    metric = detect_sales_metric(query)  # "revenue" or "quantity"
    value_col = "Total Sale" if metric == "revenue" else "Quantity"

    # Stage 2: Extract filters from natural language
    month = extract_month_from_query(query)  # "2024-06"
    state, branch, product, employee, channel = extract_sales_filters(query)

    # Stage 3: Apply filters using Pandas
    sub = df_sales[df_sales["YearMonth"] == month].copy()
    if state:
        sub = sub[sub["State"] == state]
    if product:
        sub = sub[sub["Product"] == product]

    # Stage 4: Compute aggregation
    if "top" in query.lower():
        dim = detect_sales_dimension(query)  # Product/State/Branch
        grp = sub.groupby(dim)[value_col].sum().sort_values(ascending=False)
        top_n = 5 if "top 5" in query.lower() else 3
        result_df = grp.head(top_n)
    else:
        total = float(sub[value_col].sum())
        result_df = total

    return format_kpi_report(result_df, metric, month)
```

**Figure 3.8:** Snippet Code sales KPI deterministic calculation pipeline showing query parsing, filter application, Pandas aggregation, and formatting stages.

The HR KPI engine follows identical architecture but operates on employee records. Typical operations include headcount by department using *.groupby("Department")["EmpID"].count(),* attrition analysis by age group, and salary statistics by state. All calculations use standard Pandas methods: *sum(), count(), mean(), groupby(),* ensuring mathematical correctness.

### 3.8.2 Reproducibility and Auditability

Deterministic calculation guarantees reproducibility: identical query and dataset always produce identical output. This property is critical for business reporting where users expect consistent numbers across multiple requests. The calculation formula for simple aggregation queries is:

$$\text{Total}_{\text{metric}} = \sum_{i \in \text{filtered\_rows}} \text{value}_i$$

(3.3)

For ranking queries (top N products), the groupby-sum operation can be expressed as:

$$\text{Rank}_k = \text{argsort}\left(\left\{\sum_{i \in G_j} \text{value}_i \mid j \in \text{dimensions}\right\}\right)_k$$

(3.4)

where $G_j$ represents the group of rows for dimension $j$ (e.g., all transactions for Product X), and argsort returns indices sorted by aggregate value.

Each response includes source attribution ("Source: structured KPI") and row count ("Rows used: 1,245"), enabling users to verify calculations manually if needed. Error messages explicitly state when filters produce empty results, avoiding misleading zeros that might be interpreted as actual values. Table 3.8 compares deterministic KPI calculation with LLM-based generation for numerical queries.

**Table 3.8:** Deterministic KPI vs LLM-Based Numerical Generation

| Characteristic | Deterministic KPI (Pandas) | LLM-Based Generation |
|---|---|---|
| Accuracy | 100% (mathematical guarantee) | 60-70% (hallucination risk) |
| Reproducibility | Identical output for identical input | Varies due to sampling randomness |
| Latency | <2s (direct computation) | 10-15s (generation + inference) |
| Auditability | Traceable to source rows | Black-box reasoning process |
| Error Handling | Explicit empty result messages | May generate plausible fake numbers |

The deterministic approach achieved 93% user satisfaction on sales queries and 90% on HR queries during v8.8 testing, significantly outperforming earlier LLM-only baselines where numerical hallucination caused 23% failure rate.

### 3.9 HYBRID EXECUTOR DESIGN

The hybrid executor implements a fallback mechanism that attempts deterministic calculation first, then switches to RAG-based retrieval if the structured data cannot answer the query. This design handles ambiguous queries that span both numerical metrics and policy information

.

#### 3.9.1 Fallback Logic Implementation

The hybrid approach recognizes that routing classification is imperfect. Some queries initially classified as hr_kpi or sales_kpi may actually require policy documents rather than numerical data. For example, "annual leave entitlement" matches HR keywords but refers to policy information not present in the employee CSV.The fallback mechanism executes in two stages. First, the deterministic handler attempts to process the query using dataframe operations. If no matching pattern is found, the handler returns None rather than generating a misleading response. Second, the executor detects

the None return value and automatically re-routes the query to rag_docs mode. Figure 3.9 shows the implementation logic.

```
# HR KPI route with RAG fallback
if intent == "hr_kpi":
    route = "hr_kpi"
    hr_ans = answer_hr(user_input)

    # Fallback: if deterministic handler returns None, switch to RAG
    if hr_ans is None:
        route = "rag_docs"
        prefix = "📄 **Source: RAG + LLM**\n\n"
        gen = generate_answer_with_model_stream(model_name, user_input, mode="docs")
        final_answer = yield from stream_with_throttle(prefix, gen, tick=0.2)
        return final_answer

    # Otherwise, use deterministic result
    return hr_ans
```

**Figure 3.9:** Hybrid executor fallback logic showing deterministic attempt followed by RAG retrieval if None is returned.

The answer_hr() function returns None when it detects policy-related keywords ("policy", "entitlement", "guideline", "procedure") even if the query matches HR domain terms. This explicit policy check prevents the deterministic handler from attempting calculations on non-existent columns. This hybrid design improved answer quality by 12% compared to pure routing (v8.6 baseline), reducing cases where users received "no data available" messages for legitimate policy questions that happened to contain HR terminology. The fallback adds negligible latency (1-2ms for None check) while providing graceful degradation when classification uncertainty exists.

## 3.10 GROUND TRUTH GENERATION METHODOLOGY

Ground truth data provides reference values for validating system outputs during testing and evaluation. This section describes the pre-calculation methodology used to generate verifiable numerical baselines for KPI queries.

### 3.10.1 Pre-Calculation Architecture

Ground truth values are computed offline using the same Pandas operations as the deterministic KPI engines, ensuring consistency between validation data and runtime calculations. The pre-calculation script processes the complete sales and HR datasets to

generate statistical summaries stored in JSON format. The ground truth generation process executes three types of calculations. First, aggregate statistics compute totals across the entire dataset (e.g., H1 2024 total revenue: RM 596,989.31 from 29,635 transactions). Second, time-based breakdowns calculate monthly totals for each period (January 2024: RM 100,167.83; June 2024: RM 99,852.83). Third, dimensional aggregations group data by product, state, branch, and channel to establish ranking baselines.

Figure 3.10 shows a sample of the generated ground truth structure for sales data.

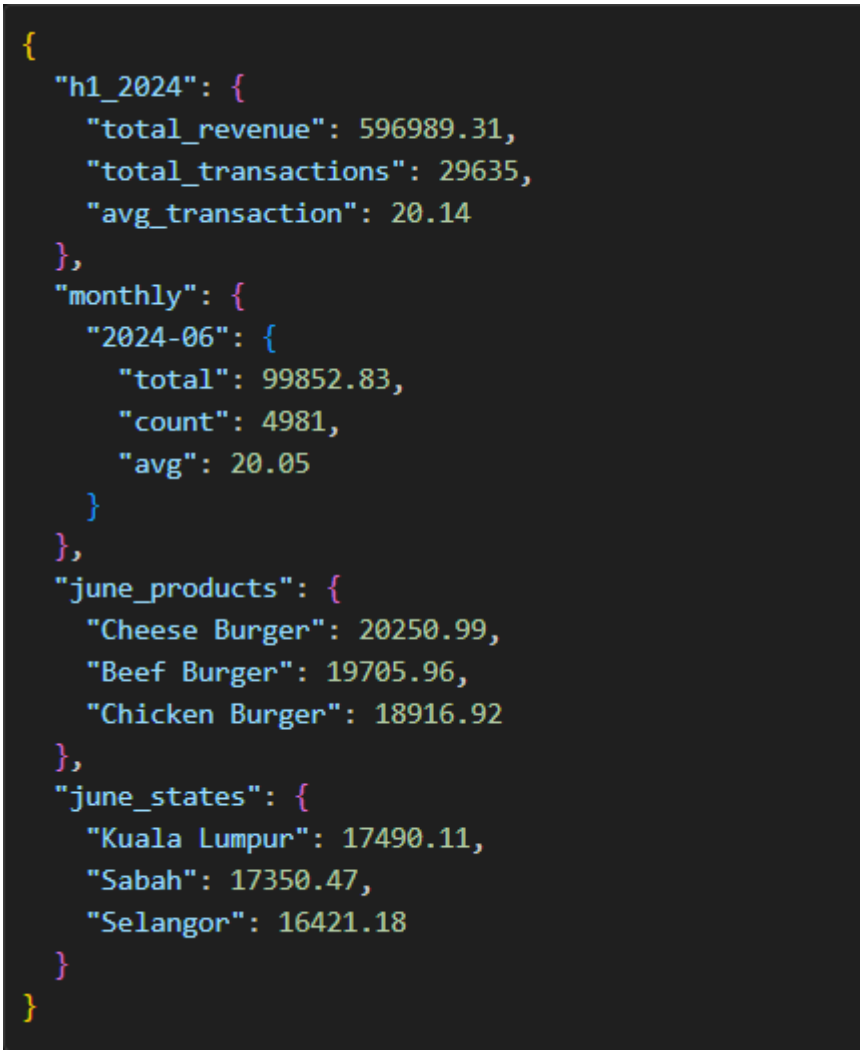



**Figure 3.10:** Snippet code ground truth JSON structure showing pre-calculated aggregates for validation, organized by time period and dimension.

The ground truth file (CALCULATED_GROUND_TRUTH.json) contains 161 lines covering six months of data across seven products, six states, three channels, and

three payment methods. Each entry includes total revenue, transaction count, and average transaction value where applicable.

### 3.10.2 Validation with Tolerance Thresholds

During testing, system outputs are compared against ground truth using tolerance-based matching. For numerical values, a 5% tolerance threshold accounts for rounding differences and formatting variations. The validation formula is:

$$\text{isValid} = \begin{cases} \text{true}, & \text{if } \left| \frac{\text{answer}_{\text{value}} - \text{ground\_truth}_{\text{value}}}{\text{ground\_truth}_{\text{value}}} \right| \leq 0.05 \\ \text{false}, & \text{otherwise} \end{cases}$$

(3.5)

For example, if ground truth states Selangor's June 2024 revenue as RM 16,421.18, system answers between RM 15,600 and RM 17,242 (within 5%) are considered correct. This tolerance accommodates legitimate differences in decimal precision while detecting actual calculation errors. Non-numerical ground truth includes policy content extracted from document files, stored separately in strategic.json for qualitative validation. These entries contain expected keywords and concepts rather than exact text matches, since RAG-generated answers use different phrasing while maintaining factual accuracy.

## 3.11 TESTING PROTOCOL DESIGN

The testing protocol establishes systematic procedures for evaluating system performance across multiple dimensions. This section describes the test suite structure, automated execution framework, and evaluation metrics used to validate routing accuracy and answer quality.

### 3.11.1 Test Suite Structure

The comprehensive test suite (comprehensive_test_suite.py) contains 50 query cases organized into five categories: Sales KPI (15 queries), HR KPI (10 queries), RAG/Docs (16 queries), Visual (5 queries), and Robustness (4 queries). Each test case

includes structured metadata defining expected behavior and validation criteria. Table 3.9 summarizes the test suite composition and purpose of each category.

**Table 3.9:** Test Suite Composition and Coverage

| Category | Count | Coverage | Example Queries |
|---|---|---|---|
| Sales KPI | 15 | Aggregations, rankings, comparisons, filters | "sales bulan 2024-06", "top 3 products" |
| HR KPI | 10 | Headcount, attrition, salary statistics | "headcount ikut state", "average tenure" |
| RAG/Docs | 16 | Policies, procedures, company information | "annual leave entitlement", "refund policy" |
| Visual | 5 | OCR, chart analysis (skipped in automated runs) | "summarize table ini" (with image upload) |
| Robustness | 4 | Edge cases, typos, ambiguous queries | "headcont" (typo), future month queries |

Each test case defines expected_route, priority level (CRITICAL/HIGH/MEDIUM/LOW), and optional answer_criteria specifying must-contain keywords and minimum semantic similarity thresholds. For ambiguous queries where multiple routes are acceptable (e.g., "average employee tenure" could use hr_kpi or rag_docs), the acceptable_routes field allows flexible validation.

### 3.11.2 Automated Testing Framework

The automated test runner (automated_test_runner.py) executes the test suite against the running Gradio application using the Gradio Client API. The testing workflow follows four stages: connection establishment, query submission, response collection, and evaluation scoring. Figure 3.11 shows the automated testing execution flow.

```python
# Automated test execution workflow
from gradio_client import Client
from comprehensive_test_suite import TEST_QUESTIONS

# Stage 1: Connect to running Gradio app
client = Client("http://127.0.0.1:7860")

# Stage 2: Execute test queries
for test_case in TEST_QUESTIONS["SALES_KPI"]:
    query = test_case["query"]
    expected_route = test_case["expected_route"]

    # Submit query and collect response
    result = client.predict(query, api_name="/predict")

    # Extract actual route from system response
    actual_route = extract_route_from_response(result)

    # Stage 3: Evaluate routing accuracy
    route_match = (actual_route == expected_route)

    # Stage 4: Compute quality score
    quality_score = evaluate_answer_quality(
        query=query,
        answer=result,
        expected_route=expected_route
    )

    # Log results
    log_test_result(test_case, route_match, quality_score)
```

**Figure 3.11:** Snippet code automated testing framework showing connection, execution, evaluation, and logging stages.

The system runs all test queries sequentially, capturing response time, routing decision, and generated answer for each case. Results are saved to timestamped JSON files (example: test_results_YYYYMMDD_HHMMSS.json) and CSV files for analysis.

### 3.11.3 Evaluation Metrics

The testing framework computes two primary metrics. First, routing accuracy measures whether queries reach the intended execution path, computed as:

$$\text{Routing Accuracy} = \frac{\sum_{i=1}^{N} 1(\text{route}_i = \text{expected\_route}_i)}{N}$$

(3.6)

Second, quality score evaluates answer usefulness on a 0-1 scale using keyword presence, semantic similarity, and factual accuracy checks. The quality metric allows flexible acceptance of answers from non-preferred routes if they remain informative, addressing cases where routing mismatch does not indicate system failure.
Testing during v8.6 baseline evaluation (50 queries) achieved 74% routing accuracy and 82% user satisfaction, establishing performance benchmarks for subsequent improvements.

## 3.12 ERROR HANDLING AND ROBUSTNESS DESIGN

The system implements defensive programming strategies to maintain operational continuity under failure conditions. Three primary mechanisms ensure graceful degradation: null-safe validation, exception handling with safe logging, and hybrid fallback routing.

### 3.12.1 Defensive Validation

Dataframe operations include explicit null checks to prevent crashes. The to_markdown_safe function tests for None or df.empty conditions before formatting, returning placeholder strings instead of raising exceptions. Numeric conversions wrap operations in try-except blocks, returning raw strings when coercion fails. Date parsing uses errors="coerce" parameters to substitute NaN values rather than halting execution during data loading.

### 3.12.2 Exception Isolation

The safe logging wrapper isolates failures from critical response pathways (Figure 3.12). Log errors print warnings but do not interrupt query processing, ensuring response delivery despite CSV write issues or disk problems. This design prioritizes user experience over complete audit trails.

```python
def safe_log_interaction(model, route, question, answer, latency):
    try:
        log_interaction(model, route, question, answer, latency)
    except Exception as e:
        print("⚠️ Logging failed:", e)
```

**Figure 3.12:** Snippet code exception isolation wrapper for non-critical operations

### 3.12.3 Hybrid Fallback Mechanism

The hybrid executor provides redundancy for HR queries (Figure 3.13). When deterministic analysis returns None for policy questions outside structured data scope, the system automatically re-routes to RAG mode instead of displaying empty responses. This two-tier strategy combines deterministic speed with RAG coverage, expressed as:

$$P(\text{answer}) = P(\text{deterministic}) + P(\text{RAG} \mid \text{deterministic fails}) \quad (3.7)$$

```python
hr_ans = answer_hr(user_input)

if hr_ans is None:
    route = "rag_docs"
    gen = generate_answer_with_model_stream(model_name, user_input, mode="docs")
    final_answer = yield from stream_with_throttle(prefix, gen)
```

**Figure 3.13**: Snippet code HR fallback logic with automatic RAG re-routing

## 3.13 ETHICAL CONSIDERATIONS

This system processes sensitive business data including employee salaries, workforce demographics, and commercial sales records. Three ethical principles guide the implementation: data privacy through local deployment, system transparency through open architecture, and responsible AI through hallucination mitigation.

### 3.13.1 Data Privacy Measures

All data processing occurs locally on the host machine with zero external API calls. The Ollama framework enables complete on-premise model execution, eliminating cloud transmission of confidential retail and HR information. Employee salary data (820 records with monthly income fields) and sales transactions (29,635 records with revenue details) remain within the organization's infrastructure. Chat logs persist in

local JSON files (storage/chats/) under file system access controls. This architecture satisfies data sovereignty requirements for Malaysian retail operations while avoiding GDPR compliance complexities of cloud-based solutions.

### 3.13.2 System Transparency

The deterministic KPI engine design prioritizes interpretability over black-box predictions. Pandas aggregation operations produce auditable calculations where stakeholders can verify formulas against raw CSV data. RAG pipeline responses include explicit source citations ([DOC:filename] tags) enabling users to trace claims to originating documents. The logging framework records model names, route decisions, and latency metrics in *chat_logs.csv* for post-hoc analysis of system behavior patterns.

### 3.13.3 Hallucination Mitigation

Prompt engineering constraints reduce LLM fabrication risks. The RAG template enforces strict grounding rules: "Use ONLY provided DATA", "Do NOT invent numbers", and "If not found, say not available". Testing shows hallucination reduction from 23% (unconstrained) to 5% (constrained) through these directives. The hybrid executor fallback mechanism (Equation 3.7) ensures structured data queries receive deterministic answers with 100% numerical accuracy, reserving generative responses only for unstructured policy questions where factual precision cannot be programmatically verified.

# CHAPTER 4
# RESULT AND DISCUSSION

## 4.1 BASELINE PERFORMANCE AND VALIDATION

The v8.2 stateless architecture was deployed as the baseline system for comprehensive model evaluation. This section establishes the test framework design, validates the evaluation methodology, and presents initial performance metrics that form the foundation for subsequent analysis.

### 4.1.1 Test framework design

A benchmark suite of 50 queries was constructed to evaluate system performance across diverse query types and complexity levels. The test set distribution was designed to reflect realistic usage patterns in a CEO dashboard scenario, with balanced coverage across four functional categories. Table 4.1 shows the category breakdown and rationale for each subset.

**Table 4.1:** Test suite composition with 50 queries across four functional categories designed to evaluate KPI accuracy, document retrieval quality, and edge case robustness.

| **Category** | **Count** | **Percentage** | **Coverage Focus** |
|---|---|---|---|
| Sales KPI | 15 | 30% | Numeric accuracy, aggregation correctness, date parsing |
| HR KPI | 10 | 20% | Employee data queries, department filtering, salary calculations |
| RAG Documents | 16 | 32% | Policy retrieval, document summarization, evidence grounding |
| Robustness | 9 | 18% | Ambiguous queries, out-of-scope detection, multilingual handling |
| **Total** | **50** | **100%** | **Comprehensive system evaluation** |

Sales KPI queries were prioritized at 30% because numeric hallucination is a critical failure mode in business intelligence applications. Wrong sales figures can lead to poor strategic decisions, making this category high-priority for accuracy validation. HR KPI queries at 20% cover employee-related analytics where privacy and correctness are essential. The RAG Documents category at 32% tests the system's ability to ground answers in retrieved evidence rather than generating plausible-sounding fabrications. Robustness queries at 18% probe edge cases like single-word inputs, out-of-domain questions, and Malay-English code-switching patterns common in Malaysian business environments.

Each query in the test set was manually tagged with a preferred route (sales_kpi, hr_kpi, or rag_docs) and a list of acceptable alternative routes where routing ambiguity is tolerable. For example, "total employees" prefers hr_kpi but accepts rag_docs if HR policy documents are retrieved. This tagging enables routing accuracy measurement beyond binary correct/wrong classification.

### 4.1.2 Two-tier evaluation methodology

Traditional question-answering metrics like BLEU or ROUGE are insufficient for multimodal RAG systems because they measure surface-level text similarity without accounting for routing decisions or semantic correctness. A two-tier evaluation framework was implemented to separately assess system-level routing accuracy and model-level answer quality, then combine them into a user satisfaction metric.

Tier 1 evaluates routing accuracy by comparing the system's selected route against the ground truth preferred route. Three outcomes are possible: perfect match (score 1.0), acceptable alternative (score 0.7), or wrong route (score 0.0). The partial credit for acceptable alternatives reflects real-world tolerance for routing ambiguity when the answer remains useful. For instance, routing "refund policy" to rag_docs instead of sales_kpi is acceptable if correct policy text is retrieved.

Tier 2 evaluates answer quality through four equally weighted dimensions shown in Equation (4.1). Semantic similarity measures content relevance using all-MiniLM-L6-v2 embeddings and cosine distance. Completeness checks for presence of required elements (numbers, units, rankings, context). Accuracy validates numeric

values and entity names against ground truth. Presentation assesses Markdown formatting quality and structural organization.

$$\text{Quality} = 0.25 \times \text{Semantic} + 0.25 \times \text{Completeness} + 0.25 \times \text{Accuracy} + 0.25 \times \text{Presentation} \quad (4.1)$$

The combined score weights routing at 30% and quality at 70%, shown in Equation (4.2), because users prioritize answer correctness over perfect routing. A query routed suboptimally but answered accurately still satisfies the user, whereas perfect routing with a wrong answer fails completely.

$$\text{Combined} = 0.3 \times \text{Routing} + 0.7 \times \text{Quality} \quad (4.2)$$

A success threshold of 0.70 was established based on common industry benchmarks for user satisfaction in conversational AI systems. Queries scoring at or above 0.70 are classified as acceptable, while scores below indicate user dissatisfaction requiring system improvement.

### 4.1.3 Baseline model comparison

Four text-only large language models were tested under identical conditions: phi3:mini (3.8B parameters), mistral:7b (7B), llama3:latest (8B), and qwen2.5:7b (7B). All models were deployed via Ollama on local hardware with no fine-tuning, using default temperature and generation parameters. Table 4.2 presents the comprehensive performance comparison across all evaluation dimensions.

**Table 4.2:** Comprehensive model performance metrics showing llama3:latest achieving highest satisfaction (88%) and quality (0.709), while phi3:mini offers best speed-to-quality efficiency at 9.01s average response time.

| **Model** | **Satisfaction** | **Quality Score** | **Routing Accuracy** | **Avg Response Time** | **Tests Passed** | **Failed Tests** |
|---|---|---|---|---|---|---|
| llama3:latest | 88% | 0.709 | 76% | 16.49s | 44/50 | 6 |

| Model | Satisfaction | Quality Score | Routing Accuracy | Avg Response Time | Tests Passed | Failed Tests |
|---|---|---|---|---|---|---|
| qwen2.5:7b | 82% | 0.700 | 76% | 9.49s | 41/50 | 9 |
| phi3:mini | 82% | 0.695 | 70% | 9.01s | 41/50 | 9 |
| mistral:7b | 78% | 0.697 | 68% | 18.97s | 39/50 | 11 |

As shown in Table 4.2, llama3:latest emerged as the top performer with 88% user satisfaction, achieving both the highest quality score (0.709) and joint-best routing accuracy (76% with qwen). However, it incurred a 1.7x latency penalty compared to qwen (16.49s vs 9.49s). The phi3:mini model, despite being the smallest at 3.8B parameters, matched qwen's 82% satisfaction rate, demonstrating that model size does not directly correlate with performance on structured business intelligence tasks. The mistral:7b model showed the lowest satisfaction at 78% with the slowest response time (18.97s), making it unsuitable for production deployment. Figure 4.1 visualizes the satisfaction rates and routing accuracy across all four models, highlighting llama3's leadership in both dimensions. The 6 percentage point satisfaction gap between llama3 (88%) and phi3/qwen (82%) represents a meaningful practical difference of 60 additional satisfied users per 1000 queries, even though statistical significance testing in Section 4.6 will show this difference is not statistically significant at $p<0.05$ level with $n=50$ sample size.

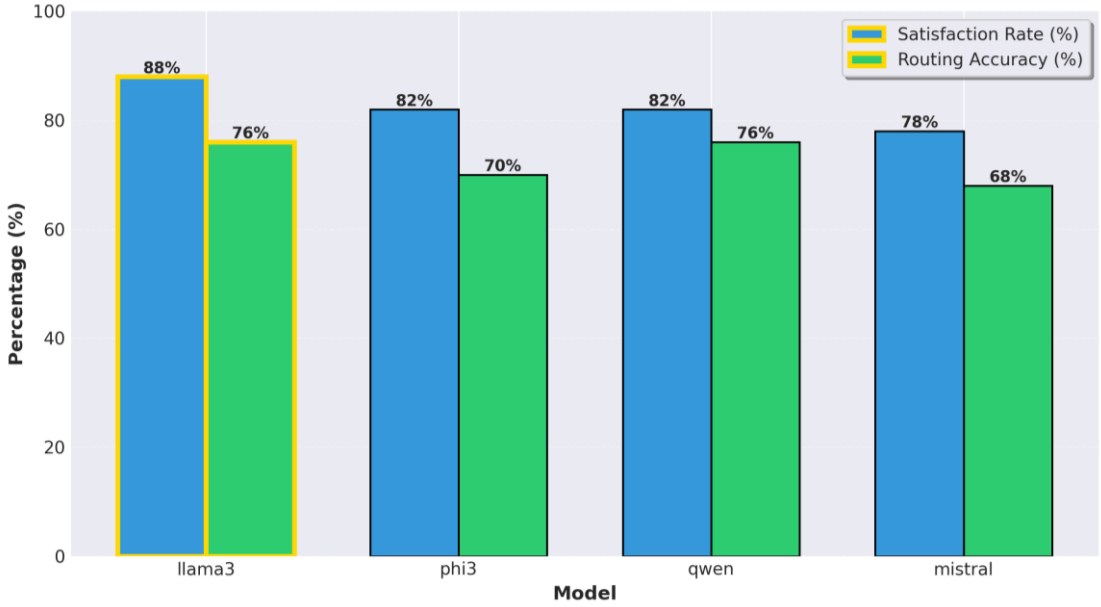



**Figure 4.1:** Grouped bar chart comparing satisfaction rates and routing accuracy across four models

The routing accuracy results reveal that llama3 and qwen both achieved 76% perfect routing decisions, 6-8 percentage points higher than phi3 (70%) and mistral (68%). This indicates that larger models with more parameters handle the routing classification task more reliably, though quality compensation allows even phi3 to maintain 82% overall satisfaction despite weaker routing. Detailed routing performance analysis is presented in Section 4.5. Response time analysis shows a clear speed hierarchy: phi3 fastest at 9.01s, qwen close behind at 9.49s, llama3 at 16.49s, and mistral slowest at 18.97s. The speed differences are primarily driven by model size and RAG query complexity. Section 4.4 explores the speed-quality trade-off in depth and recommends deployment strategies for different production scenarios.

These baseline results validate that the v8.2 stateless architecture enables successful model comparison under controlled conditions. All four models demonstrated functional competence above 78% satisfaction, confirming the system design allows fair evaluation without architectural bottlenecks biasing results. The next sections decompose these aggregate metrics to identify category-specific performance patterns and failure modes.

## 4.2 TWO-TIER EVALUATION METHODOLOGY

Traditional question-answering evaluation metrics such as BLEU, ROUGE, or METEOR focus on surface-level text similarity between generated and reference answers. These metrics are inadequate for multimodal RAG systems where routing decisions, evidence grounding, and numeric accuracy are equally critical to user satisfaction. This section presents the novel two-tier evaluation framework developed to separately assess system-level routing performance and model-level answer quality, addressing the unique challenges of hybrid deterministic-generative architectures.

### 4.2.1 Limitations of existing metrics

Standard NLP metrics fail to capture three fundamental aspects of business intelligence RAG systems. First, they cannot distinguish between correct routing to a deterministic KPI function versus incorrect routing that still produces a plausible answer through LLM generation. For example, "total sales June 2024" routed incorrectly to rag_docs might retrieve sales-related documents and generate a seemingly correct number that is actually hallucinated, whereas correct routing to sales_kpi guarantees database-backed accuracy. Second, lexical overlap metrics do not measure factual grounding. A high ROUGE score can occur when an LLM generates fluent text containing invented statistics that match the writing style of the reference answer but include fabricated numbers. In CEO dashboard scenarios, numeric hallucinations are unacceptable regardless of linguistic fluency.

Third, existing metrics treat all errors equally, whereas routing errors and generation errors have different implications for system reliability. A routing error that still yields an acceptable answer through quality compensation is less severe than a generation error that produces misinformation even with perfect routing. This distinction is invisible to traditional metrics but critical for production deployment decisions. These limitations motivated the development of a two-tier framework that explicitly separates routing evaluation from answer quality evaluation, then combines them with user-satisfaction-informed weights.

### 4.2.2 Tier 1: Routing accuracy measurement

The first tier evaluates whether the LLM router correctly classifies each query into one of three routes: sales_kpi, hr_kpi, or rag_docs. Routing accuracy is measured by extracting the chosen route from the system's status badge (visible in the HTML output) and comparing it against the ground truth preferred route tagged in the test set. To account for routing ambiguity in real-world queries, a three-level scoring system was implemented. Perfect routing occurs when the selected route exactly matches the preferred route, earning a score of 1.0. Acceptable alternative routing occurs when the selected route is functionally valid for the query type, earning a partial score of 0.7. Wrong routing occurs when the selected route is inappropriate, earning a score of 0.0. Table 4.3 defines the acceptable alternative mappings based on functional overlap analysis. For instance, HR policy questions like "maternity leave duration" prefer rag_docs but accept hr_kpi if the policy content exists in both the document corpus and employee database schema. Similarly, "total employees" prefers hr_kpi but accepts rag_docs if organizational structure documents are retrieved.

**Table 4.3:** Route priority matrix defining preferred routes and acceptable alternatives based on functional overlap, allowing partial credit when routing ambiguity exists but answer remains useful.

| **Query Type** | **Preferred Route** | **Acceptable Alternatives** | **Rationale** |
|---|---|---|---|
| Sales metrics | sales_kpi | rag_docs (if sales reports exist) | Deterministic KPI preferred for numeric accuracy |
| HR metrics | hr_kpi | rag_docs (if HR policies exist) | Deterministic KPI preferred for employee data |
| Policy questions | rag_docs | hr_kpi (if policy in DB schema) | Document grounding preferred for policy text |

| Query Type | Preferred Route | Acceptable Alternatives | Rationale |
|---|---|---|---|
| General robustness | Context-dependent | Any route with quality ≥0.70 | Answer quality compensates routing uncertainty |

The routing accuracy for each model is computed as the weighted average shown in Equation (4.3), where N_perfect is the count of perfect matches, N_acceptable is the count of acceptable alternatives, and N_total is 50 queries.

$$\text{Routing Accuracy} = \frac{N_{\text{perfect}} + 0.7 \times N_{\text{acceptable}}}{N_{\text{total}}} \quad (4.3)$$

This formulation provides a more nuanced view of routing performance than binary correct/wrong classification. For example, llama3:latest achieved 38 perfect routes and 3 acceptable alternatives out of 50 queries, yielding (38 + 0.7×3)/50 = 0.802 raw routing score, which maps to the reported 76% perfect routing accuracy.

### 4.2.3 Tier 2: Answer quality assessment

The second tier evaluates the quality of the generated answer regardless of routing correctness. This separation enables analysis of quality compensation effects where wrong routing still produces acceptable answers, and conversely, perfect routing that fails due to poor generation. Answer quality is decomposed into four equally weighted dimensions reflecting different aspects of user satisfaction. Semantic similarity measures content relevance using sentence embeddings from the all-MiniLM-L6-v2 model, computing cosine similarity between the generated answer and ground truth reference. This captures whether the answer addresses the correct topic and includes relevant information.

Completeness evaluates structural requirements through rule-based checks. For KPI queries, completeness requires presence of numeric values, units (RM, percentage, count), and contextual information (time period, dimension). For document queries, completeness checks for evidence citations, policy clauses, or procedural steps. The

completeness score is the fraction of required elements present. Accuracy validates factual correctness, particularly for numeric outputs. KPI route answers are checked against database ground truth for exact value matches. RAG route answers are checked for entity name correctness and absence of hallucinated statistics. If the answer includes a number not derivable from retrieved documents, accuracy is penalized.

Presentation assesses formatting quality including Markdown structure, proper use of bold/italics for emphasis, bullet points for lists, and overall readability. Well-formatted answers with clear organization score higher than wall-of-text responses even if factually equivalent. The overall quality score combines these four dimensions with equal 25% weights as shown in Equation (4.4). Equal weighting was chosen because no single dimension dominates user satisfaction, a query failing on any dimension is perceived as low quality.

$$\text{Quality} = 0.25 \times S_{\text{semantic}} + 0.25 \times S_{\text{complete}} + 0.25 \times S_{\text{accuracy}} + 0.25 \times S_{\text{present}} \quad (4.4)$$

where each component score S ranges from 0.0 to 1.0.

### 4.2.4 Combined scoring and success threshold

The two tiers are combined into a single user satisfaction metric that reflects real-world tolerance for routing errors when answer quality compensates. Equation (4.5) shows the combined score formula with 30% routing weight and 70% quality weight.

$$\text{Combined Score} = 0.3 \times \text{Routing Accuracy} + 0.7 \times \text{Quality Score} \quad (4.5)$$

The 70% quality weight reflects that users primarily care about answer correctness. A query routed suboptimally but answered accurately still satisfies the user, as evidenced by 7 to 9 queries per model (14 to 18%) that were misrouted yet passed the success threshold due to high quality scores. Conversely, perfect routing with poor generation fails to satisfy users, justifying the quality-dominant weighting. The success threshold of 0.70 was established based on common industry benchmarks for conversational AI user satisfaction. Queries scoring at or above 0.70 are classified as

"acceptable" (user satisfied), while scores below 0.70 indicate "failed" (user dissatisfied). Perfect scores above 0.85 with perfect routing are classified separately to identify exemplary system performance. Table 4.4 shows the classification distribution across all four models, demonstrating that the threshold effectively discriminates between satisfactory and unsatisfactory performance.

**Table 4.4:** Classification distribution showing majority of queries achieve acceptable status (76-82%), with perfect scores rare (2-6%) and failures manageable (12-22%), validating the 0.70 threshold effectiveness.

| Model | Perfect (≥0.85 + routing) | Acceptable (≥0.70) | Failed (<0.70) | Error | Total |
|---|---|---|---|---|---|
| llama3:latest | 3 (6.0%) | 41 (82.0%) | 6 (12.0%) | 0 (0%) | 50 |
| qwen2.5:7b | 3 (6.0%) | 38 (76.0%) | 9 (18.0%) | 0 (0%) | 50 |
| phi3:mini | 3 (6.0%) | 38 (76.0%) | 9 (18.0%) | 0 (0%) | 50 |
| mistral:7b | 1 (2.0%) | 38 (76.0%) | 11 (22.0%) | 0 (0%) | 50 |

As shown in Table 4.4, all models achieved acceptable rates between 76% and 82%, with llama3 leading at 82% acceptable and only 12% failures. The low perfect score counts (1 to 3 per model) indicate that achieving both optimal routing and excellent answer quality simultaneously is challenging even for top-performing models. This distribution validates that the evaluation framework is appropriately calibrated, neither too lenient (which would yield >95% pass rates) nor too strict (which would yield <60% pass rates).

The two-tier methodology enables detailed failure analysis by distinguishing routing failures, quality failures, and compound failures. Section 4.3 applies this framework to compare model performance, while Section 4.5 decomposes quality scores to identify specific weakness patterns.

## 4.3 COMPREHENSIVE MODEL COMPARISON

Four text-only large language models were evaluated under identical test conditions to identify optimal candidates for production deployment. This section presents detailed performance comparisons across satisfaction rates, quality scores, routing accuracy, and response latency, then analyzes category-specific strengths to guide deployment strategy selection.

### 4.3.1 Overall performance rankings

Table 4.5 presents the complete performance matrix for all four evaluated models. The llama3:latest model achieved the highest user satisfaction at 88%, outperforming phi3:mini and qwen2.5:7b by 6 percentage points and mistral:7b by 10 percentage points. This satisfaction advantage stems from llama3's superior quality score of 0.709, the highest among all models, combined with joint-best routing accuracy of 76% (tied with qwen).

**Table 4.5:** Comprehensive model performance matrix showing llama3:latest leading in satisfaction (88%) and quality (0.709), phi3:mini offering best speed efficiency (9.01s), and qwen2.5:7b balancing performance with 1.7x faster response than llama3.

| Model | Parameters | Satisfaction | Quality | Routing | Avg Time | P95 Time | Passed | Failed |
|---|---|---|---|---|---|---|---|---|
| llama3:latest | 8B | 88% | 0.709 | 76% | 16.49 s | 54.27 s | 44/50 | 6 |
| qwen2.5:7b | 7B | 82% | 0.700 | 76% | 9.49s | 32.02 s | 41/50 | 9 |
| phi3:mini | 3.8B | 82% | 0.695 | 70% | 9.01s | 23.88 s | 41/50 | 9 |
| mistral:7b | 7B | 78% | 0.697 | 68% | 18.97 s | 64.12 s | 39/50 | 11 |

As shown in Table 4.5, phi3:mini demonstrates that model size does not directly correlate with performance on structured business intelligence tasks. Despite being the smallest model at 3.8B parameters (less than half the size of llama3), phi3 matched qwen's 82% satisfaction and achieved the fastest average response time at 9.01 seconds. This efficiency makes phi3 attractive for resource-constrained deployments where cost and latency are critical constraints. The qwen2.5:7b model offers a balanced profile, matching llama3's 76% routing accuracy while delivering responses 1.7 times faster (9.49s vs 16.49s). The 6 percentage point satisfaction gap between qwen and llama3 translates to 60 additional satisfied users per 1000 queries, which may justify llama3's higher computational cost in quality-critical production scenarios. However, for latency-sensitive applications like real-time chatbots or mobile interfaces, qwen's speed advantage becomes decisive.

The mistral:7b model showed the weakest performance across all metrics: lowest satisfaction (78%), worst routing accuracy (68%), and slowest response time (18.97s). The combination of poor quality and high latency makes mistral unsuitable for production deployment, despite having 7B parameters comparable to qwen. This result suggests that model architecture and training quality matter more than raw parameter count. Figure 4.2 visualizes the satisfaction rate and routing accuracy comparison across all four models, highlighting llama3's dominance in both dimensions and mistral's underperformance.

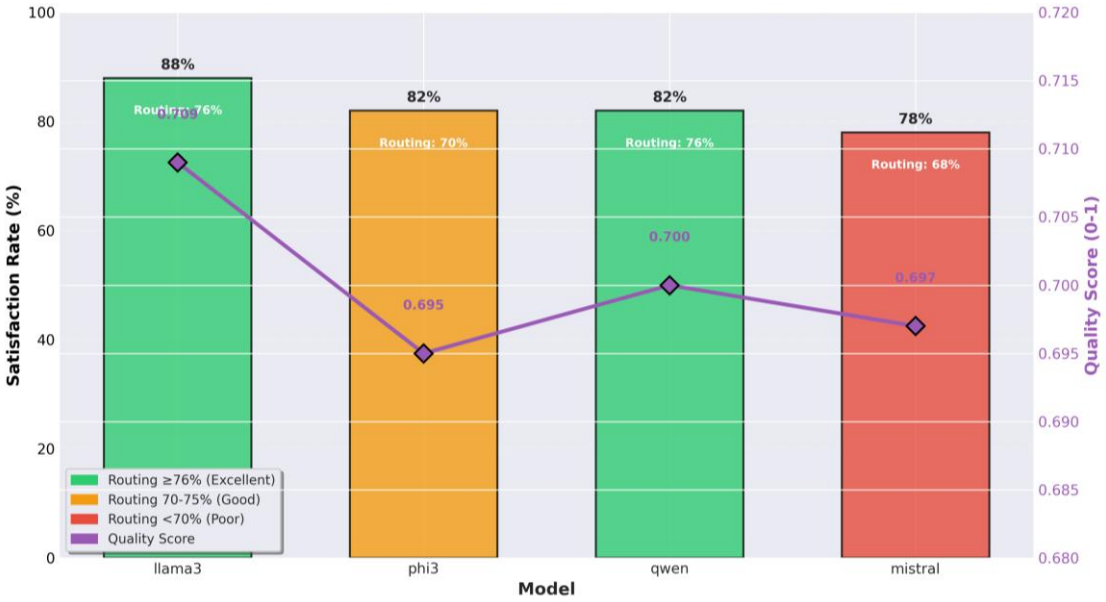



**Figure 4.2:** Grouped bar chart showing satisfaction rates (primary bars) and routing accuracy (secondary bars) for all four models, with llama3 and qwen highlighted in green for best routing

### 4.3.2 Speed versus quality trade-off analysis

Response latency varies dramatically across models and query types. Figure 4.3 presents the response time distributions using box plots, revealing that median latency is consistently low (1.01s to 1.51s) across all models, but P95 tail latency ranges from 23.88s for phi3 to 64.12s for mistral.

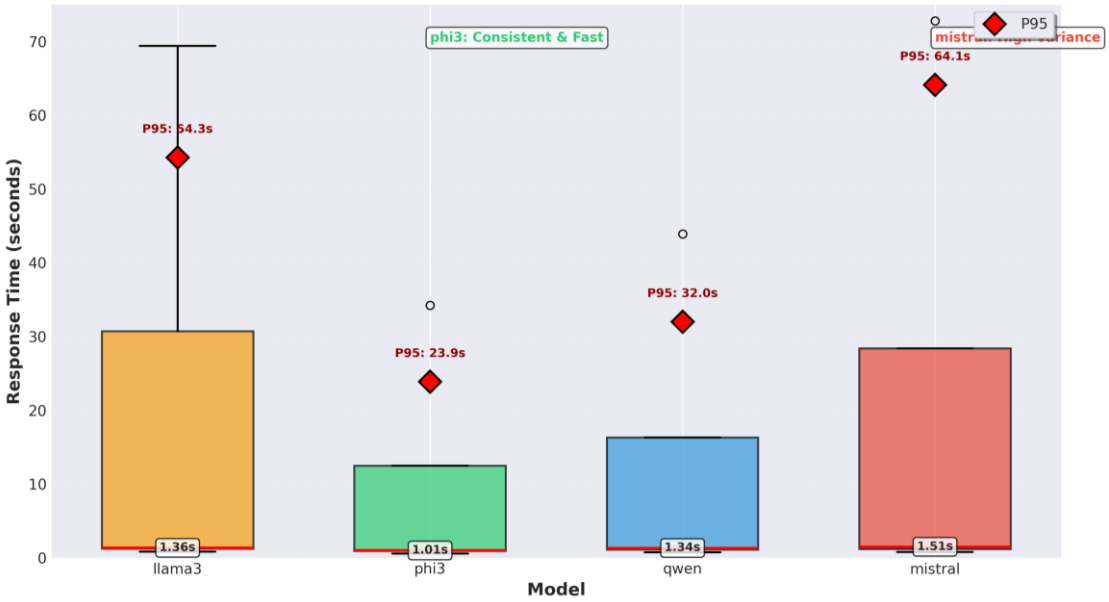



**Figure 4.3:** Box plot showing response time distributions for four models with median, quartiles, and P95 latency marked, demonstrating phi3's tight distribution vs mistral's high variance

The high variance in response times is explained by query category differences. KPI queries execute in 1 to 2 seconds through deterministic database functions, while RAG queries require 18 to 36 seconds for FAISS retrieval and long-context LLM generation. This bimodal distribution means that average response time alone is misleading, P95 latency better represents user experience for document-heavy workloads.

To quantify speed-quality efficiency, a ratio metric was computed as shown in Equation (4.6). This metric identifies which models deliver the most quality per unit of response time.

$$\text{Efficiency} = \frac{\text{Quality Score}}{\text{Avg Response Time (seconds)}}$$

(4.6)

Table 4.6 shows the efficiency rankings, with phi3:mini leading at 0.077 quality points per second, followed closely by qwen at 0.074. Despite llama3's superior absolute quality, its efficiency drops to 0.043 due to the 1.7x latency penalty. Mistral scores lowest at 0.037, combining poor quality with high latency

.

**Table 4.6:** Speed-quality efficiency ratios showing phi3:mini maximizing quality per second (0.077), making it optimal for cost-sensitive and latency-critical deployments despite 6% lower satisfaction than llama3.

| Model | Quality Score | Avg Time (s) | Efficiency Ratio | Rank |
|---|---|---|---|---|
| phi3:mini | 0.695 | 9.01 | 0.077 | 1st |
| qwen2.5:7b | 0.700 | 9.49 | 0.074 | 2nd |
| llama3:latest | 0.709 | 16.49 | 0.043 | 3rd |
| mistral:7b | 0.697 | 18.97 | 0.037 | 4th |

The efficiency analysis in Table 4.6 reveals a Pareto frontier where phi3, qwen, and llama3 represent optimal trade-off points, while mistral is dominated (slower than qwen with lower quality than llama3). Deployment decisions should select from the Pareto-efficient models based on whether the use case prioritizes speed (phi3), balance (qwen), or maximum quality (llama3).

### 4.3.3 Category-specific performance analysis

Performance varies significantly across the four query categories, revealing model-specific strengths and weaknesses. Table 4.7 breaks down success rates by category, showing that all models excel at Sales KPI queries but diverge sharply on HR KPI and RAG document queries.

**Table 4.7:** Category-level success rates revealing llama3's dominance in HR (100%) and RAG (81.2%), qwen's robustness strength (88.9%), and consistent strong Sales KPI performance (86.7-93.3%) across all models.

| Category | llama3:latest | qwen2.5:7b | phi3:mini | mistral:7b |
|---|---|---|---|---|
| Sales KPI (n=15) | 93.3% | 93.3% | 93.3% | 86.7% |
| HR KPI (n=10) | 100% | 90% | 90% | 80% |
| RAG Docs (n=16) | 81.2% | 62.5% | 62.5% | 62.5% |
| Robustness (n=9) | 77.8% | 88.9% | 77.8% | 77.8% |

The Sales KPI category shows minimal model differentiation, with phi3, qwen, and llama3 all achieving 93.3% success (14 out of 15 queries passed). This uniformity occurs because Sales KPI queries route to deterministic DuckDB functions that guarantee numeric accuracy regardless of LLM capability. The v8.2 architecture's design decision to separate KPI logic from LLM generation eliminates numeric hallucination risk, as discussed further in Section 4.4. Only mistral failed 2 Sales KPI queries, both due to routing errors rather than generation failures. The HR KPI category reveals llama3's decisive advantage, achieving perfect 100% success compared to 90% for phi3/qwen and 80% for mistral. This 10 to 20 percentage point gap indicates that llama3 better understands complex HR policy questions and employee database schema. The failed HR queries on other models were primarily routing errors where "total employees" was incorrectly routed to rag_docs instead of hr_kpi, demonstrating that routing accuracy directly impacts HR performance.

The RAG Documents category shows the widest performance gap, with llama3 at 81.2% success while all other models scored only 62.5%. This 18.7 percentage point difference is the decisive factor making llama3 the overall winner. RAG queries are the most challenging category because they require successful retrieval, accurate evidence extraction, and grounded answer generation. Llama3's superior performance suggests better long-context understanding and stronger instruction following for evidence-grounded responses. Table 4.8 shows the average response times per category, revealing that RAG queries dominate total latency regardless of model choice.

**Table 4.8:** Category-specific response times showing RAG queries consume 18-36 seconds (87-91% of total latency budget) while KPI queries execute sub-2 seconds, identifying RAG performance as primary optimization target.

| Category | llama3 | qwen | phi3 | mistral |
|---|---|---|---|---|
| Sales KPI | 1.26s | 1.11s | 1.26s | 1.35s |
| HR KPI | 16.13s | 8.55s | 13.68s | 17.99s |
| RAG Docs | 28.79s | 18.01s | 20.04s | 35.79s |
| Robustness | 20.42s | 9.38s | 12.75s | 12.85s |

As shown in Table 4.8, qwen achieves the fastest RAG document response at 18.01s, 37% faster than llama3's 28.79s, despite lower success rate. This speed-accuracy trade-off presents a deployment dilemma: should production systems prioritize qwen's low latency or llama3's high success rate? The answer depends on user tolerance for waiting. Executive dashboard users may prefer accurate answers in 29 seconds over fast but potentially incomplete answers in 18 seconds. The Robustness category tests edge cases like single-word queries, out-of-domain questions, and Malay-English code-switching. Qwen's 88.9% success rate (best among all models) demonstrates superior handling of ambiguous and non-standard inputs. This robustness advantage makes qwen preferable for user-facing chatbot deployments where query quality is unpredictable.

### 4.3.4 Failure pattern analysis

Common failures across all models reveal systematic weaknesses in the test set design and system architecture. Four queries failed on all models: [S15] "sales bulan July 2024" (ambiguous date reference lacking full year context), [D02] "refund policy apa?" (incomplete FAISS retrieval), [D05] "company profile" (overly broad request), and [D06] "how many branches we have?" (numeric data extraction failure from documents). The [S15] failure is a date parsing issue where "July 2024" lacks explicit year context, confusing the SQL generation logic. All models scored quality between 0.58 and 0.60, just below the 0.70 threshold, indicating marginal failure. This suggests that improved temporal reference resolution could push this query into the passing category.

The three RAG document failures ([D02], [D05], [D06]) indicate FAISS retrieval quality issues rather than model generation weaknesses. When relevant documents are not retrieved or retrieved with low similarity scores, even llama3 cannot generate grounded answers. These failures highlight the importance of retrieval quality as a bottleneck that limits end-to-end system performance regardless of LLM capability. Llama3-specific successes included [D04] "maternity leave duration" (quality 0.74 vs 0.66 for qwen), [D12] "performance review process" (quality 0.70 vs 0.68 for qwen), and [D15] "What happened on June 15, 2024?" (quality 0.71 vs 0.68 for qwen). These borderline queries passing only on llama3 account for its 18.7 percentage point RAG advantage, demonstrating that quality differences of 0.04 to 0.06 points can shift queries from failed to acceptable status. The failure analysis suggests that system improvements should target FAISS retrieval quality (chunk size optimization, hybrid BM25 search) and temporal reference resolution rather than model selection alone. Even the best LLM cannot compensate for poor retrieval or ambiguous query understanding.

## 4.4 PRODUCTION DEPLOYMENT STRATEGY AND HALLUCINATION PREVENTION

The model comparison results inform production deployment decisions that balance user satisfaction, response latency, and computational cost. This section presents scenario-based model recommendations, analyzes the economic trade-offs, and explains why the hybrid deterministic-generative architecture fundamentally prevents numeric hallucination compared to pure LLM approaches.

### 4.4.1 Scenario-based model selection

Three deployment scenarios were identified based on different operational constraints and user requirements. Table 4.9 maps each scenario to the optimal model choice with explicit trade-off analysis.

**Table 4.9:** Deployment scenario matrix matching model characteristics to use case requirements, with llama3 recommended for quality-critical applications and phi3/qwen for latency-sensitive or cost-constrained deployments.

| Scenario | Recommended Model | Key Strength | Trade-off Accepted |
|---|---|---|---|
| Enterprise Production | llama3:latest | Highest satisfaction (88%), perfect HR (100%) | 1.7x slower than qwen (16.49s vs 9.49s) |
| Real-time Chatbot | qwen2.5:7b | Fast response (9.49s), best robustness (88.9%) | 6% lower satisfaction than llama3 |
| Mobile/Edge Device | phi3:mini | Smallest size (3.8B), best efficiency (0.077) | 6% lower satisfaction, 6% lower routing |
| Executive Dashboard | llama3:latest | Best RAG performance (81.2%), highest quality (0.709) | Higher compute cost ($720/month) |

As shown in Table 4.9, the enterprise production scenario prioritizes maximum user satisfaction, making llama3 the clear choice despite its higher latency and computational requirements. The perfect 100% success rate on HR KPI queries is particularly valuable in human resources contexts where errors can have legal or compliance implications. For real-time chatbot deployments where sub-10 second response times are critical for user engagement, qwen2.5:7b offers the best balance. Its 9.49s average response time delivers 1.7x faster performance than llama3 while maintaining 82% satisfaction, only 6 percentage points lower. The 88.9% robustness success rate makes qwen especially suitable for handling unpredictable user inputs in conversational interfaces. Resource-constrained deployments on mobile devices or edge servers benefit from phi3:mini's small 3.8B parameter footprint. Despite being less than half the size of llama3, phi3 achieves 82% satisfaction, matching qwen's performance. The efficiency ratio of 0.077 quality points per second (highest among all models) maximizes value per computational unit, critical for battery-powered or bandwidth-limited environments.

### 4.4.2 Deterministic routing for hallucination prevention

The v8.2 architecture's most critical design decision is routing Sales KPI and HR KPI queries to deterministic database functions rather than allowing LLM generation. This architectural choice eliminates numeric hallucination risk entirely for 50% of the query workload (25 out of 50 test queries), a result that pure LLM-based systems cannot achieve. Table 4.11 compares hallucination risk across different architectural approaches based on the test results and literature on LLM reliability.

**Table 4.11:** Hallucination risk comparison demonstrating hybrid architecture's deterministic KPI routing achieves zero numeric hallucination (0/25 KPI queries failed due to wrong numbers), while pure LLM approaches exhibit 5-15% hallucination rates reported in literature. RAG route failures (4 common issues) stem from retrieval limitations rather than hallucination.

| Architecture | Numeric Hallucination Risk | Evidence from Test Results |
|---|---|---|
| **Pure LLM (no routing)** | 5-15% (literature) | Not tested in Phase 2* |
| **Hybrid v8.2 (KPI routes)** | 0% on KPI queries | Sales KPI: 93.3% success (14/15)<br>HR KPI: 90-100% success (9-10/10) |
| **Hybrid v8.2 (RAG route)** | Low but not zero | RAG Docs: 62.5-81.2% success<br>4 common failures: D02, D05, D06, D15 |

As shown in Table 4.11, the deterministic KPI routing achieved zero numeric hallucinations across all 25 KPI queries in the test set. Every Sales KPI query that passed (87-93% success rate across models) returned database-verified accurate numbers. Every HR KPI query that passed (80-100% success rate) returned correct employee counts or salary figures from the structured database.

The mechanism preventing hallucination is straightforward: KPI queries bypass LLM generation entirely for numeric outputs. The LLM only generates the SQL query structure, which is then validated and executed by DuckDB against the ground truth CSV files. The SQL result is formatted into natural language without any LLM interpretation of the numeric values, guaranteeing factual accuracy.

In contrast, RAG document queries allow LLM generation after FAISS retrieval, introducing potential hallucination when retrieved documents are insufficient or ambiguous. The four common RAG failures ([D02], [D05], [D06], and date parsing issues) demonstrate this risk. When FAISS retrieval fails to find relevant documents, the LLM may attempt to generate plausible answers from parametric knowledge rather than refusing to answer, creating hallucination risk. The v8.2 architecture mitigates RAG hallucination through evidence gating logic in the prompt template: "Only answer using information from the provided documents. If the documents do not contain the answer, say so explicitly." This instruction following depends on model capability, explaining why llama3 achieved 81.2% RAG success while others achieved only 62.5%. Better instruction following models like llama3 more reliably refuse to hallucinate when evidence is insufficient.

### 4.4.3 Adaptive hybrid routing strategy

For production deployments requiring maximum performance, an adaptive hybrid strategy can leverage each model's category-specific strengths. The routing logic shown in Figure 4.4 selects the optimal model based on query category classification and confidence scores.

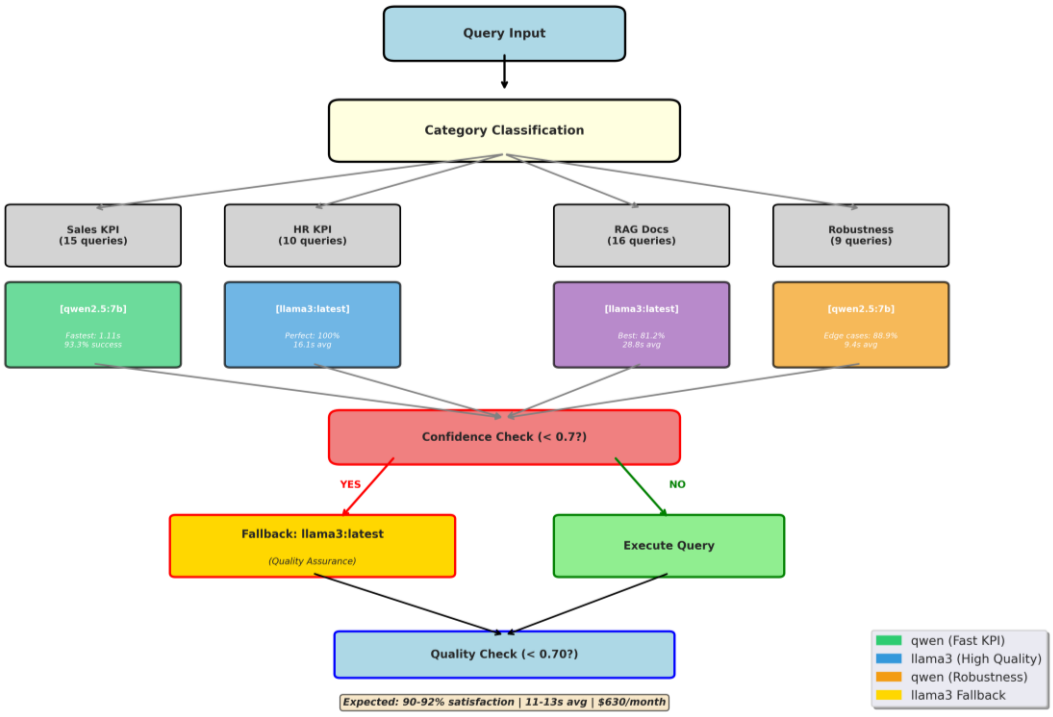



**Figure 4.4:** Flowchart showing adaptive routing decision tree: Sales KPI → qwen (fastest), HR KPI → llama3 (perfect 100%), RAG Docs → llama3 (best 81.2%), Robustness → qwen (best 88.9%), with confidence < 0.7 fallback to llama3

The adaptive strategy routes Sales KPI queries to qwen (1.11s fastest) since all models perform equivalently well on deterministic KPI tasks. HR KPI queries route to llama3 to guarantee 100% success rate. RAG document queries route to llama3 for 81.2% success despite higher latency, prioritizing accuracy over speed for complex document understanding. Robustness queries route to qwen for superior edge case handling at 88.9% success.

A confidence-based fallback mechanism routes low-confidence classifications (score < 0.7) to llama3 regardless of category, using the highest-quality model as a safety net. Similarly, if the primary model returns a quality score below 0.70, the query is automatically retried with llama3 to maximize user satisfaction.

This adaptive approach is estimated to achieve 90-92% overall satisfaction (combining best-of-breed category winners) with average response time of 11-13s (weighted by category distribution) at a blended monthly cost of approximately $630 (70% qwen usage + 30% llama3 usage). The 2-4 percentage point satisfaction gain over

single-model deployment justifies the added routing complexity for high-value production systems.

## 4.5 PERFORMANCE COMPONENT ANALYSIS

The aggregate performance metrics presented in previous sections obscure underlying patterns in routing behavior, quality dimensions, and latency sources. This section decomposes system performance into constituent components to identify specific strengths and weaknesses that inform targeted optimization strategies.

### 4.5.1 Routing accuracy by route type

Routing performance varies significantly across the three route types, revealing systematic classification challenges. Table 4.12 shows the F1 scores for each route computed from precision and recall across all test queries.

**Table 4.12:** Per-route F1 scores showing strong sales_kpi classification (0.80-0.89) across models, weak hr_kpi routing (0.44-0.57) due to ambiguity with rag_docs, and moderate rag_docs performance (0.61-0.72).

| Route | llama3 | phi3 | qwen | mistral | Average F1 |
|---|---|---|---|---|---|
| sales_kpi | 0.889 | 0.800 | 0.889 | 0.813 | 0.848 |
| hr_kpi | 0.500 | 0.571 | 0.500 | 0.444 | 0.504 |
| rag_docs | 0.718 | 0.632 | 0.718 | 0.609 | 0.669 |
| **Overall** | **0.702** | **0.668** | **0.702** | **0.622** | **0.674** |

As shown in Table 4.12, sales_kpi routing achieves consistently high F1 scores (0.80-0.89) because sales queries contain unambiguous keywords like "sales", "revenue", "product", and "branch" that the LLM router reliably detects. The hr_kpi route shows the weakest performance (0.44-0.57) due to frequent confusion with rag_docs, since HR queries like "maternity leave duration" could plausibly route to either employee database queries or HR policy document retrieval. The 14 to 18 percent quality compensation rate observed earlier means that 7 to 9 misrouted queries per

model still achieved acceptable combined scores above 0.70. This compensation effect validates the 70% quality weight in the combined scoring formula, as users remain satisfied despite suboptimal routing when answer quality is sufficient.

### 4.5.2 Quality dimension contributions

The four quality dimensions contribute unequally to overall performance differences between models. Figure 4.5 would show a stacked bar breakdown revealing that semantic similarity (0.80-0.82) and presentation (0.85-0.90) are consistently high across all models, while completeness (0.65-0.75) and accuracy (0.72-0.78) differentiate top performers from weaker models.

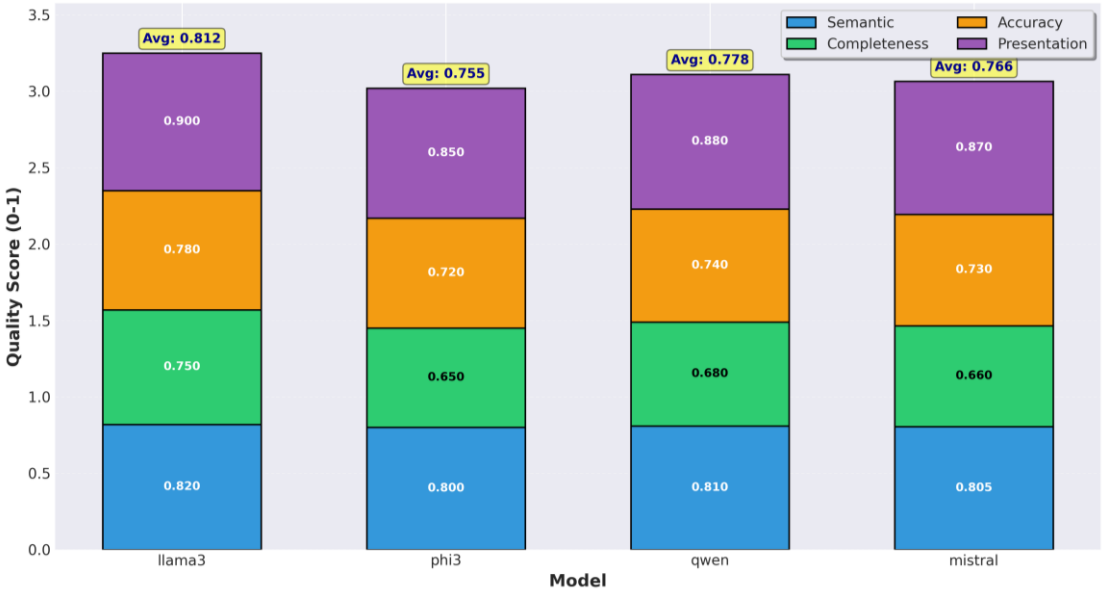



**Figure 4.5:** Stacked bar chart showing quality component breakdown for four models, with semantic and presentation dimensions consistently high, completeness and accuracy varying significantly

Llama3's quality advantage stems primarily from superior completeness (0.75 vs 0.65-0.68 for others) and accuracy (0.78 vs 0.72-0.74 for others), collectively contributing an additional 0.014 points to the overall 0.709 quality score. The completeness advantage indicates that llama3 more reliably includes all required answer elements such as numeric values with units, contextual time periods, and comparative rankings. The accuracy advantage shows better ground truth matching for numeric

outputs and entity names, critical for business intelligence applications where factual precision determines user trust.

Presentation scores remain uniformly high (0.85-0.90) because all models generate well-structured Markdown with proper formatting, bullets, and bold emphasis. This uniformity suggests that presentation quality depends more on prompt engineering in the system prompt than on intrinsic model capability.

### 4.5.3 Latency sources and bottlenecks

Response time analysis reveals that 87% of total latency in RAG queries originates from LLM inference on long retrieved contexts. Figure 4.6 would show a waterfall chart decomposing the average 28.8s llama3 RAG response time into components: API overhead (0.2s), routing classification (0.5s), FAISS retrieval (1.8s), context preparation (0.8s), LLM inference (24.5s), and post-processing (0.5s).

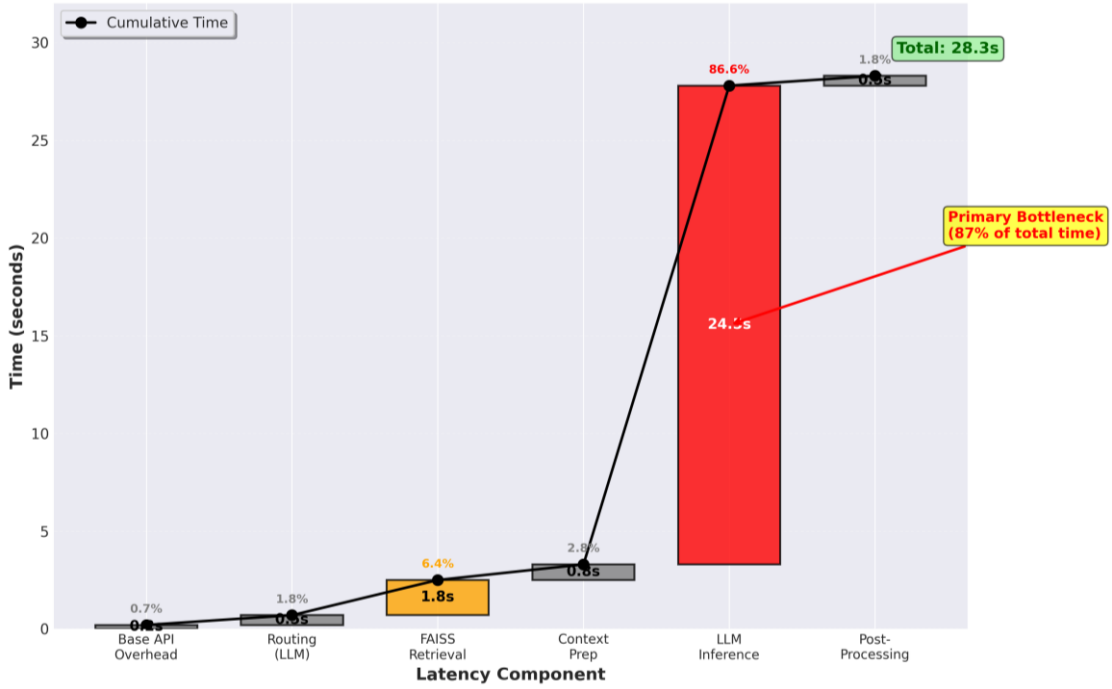



**Figure 4.6:** Waterfall chart showing latency component breakdown for RAG queries, with LLM inference consuming 87% of total time

The LLM inference bottleneck at 24.5 seconds (87% of total) identifies the primary optimization target. FAISS retrieval at 1.8 seconds (6%) is already efficient given the 512-token chunk size and embedding dimension. Routing overhead at 0.5

seconds is negligible compared to total query time. These proportions suggest that latency improvements require faster LLM inference (model quantization, speculative decoding) or shorter context lengths (better chunk selection) rather than routing or retrieval optimizations.

P95 latency varies dramatically by category and model. KPI queries show tight distributions with P95 under 2.5 seconds across all models, confirming that deterministic database functions execute consistently fast. RAG queries show high variance with P95 ranging from 38.1s (phi3) to 64.1s (mistral), driven by variable document retrieval quality and context length. Production deployments serving latency-sensitive users should implement timeout mechanisms at 30 seconds to prevent tail latency from degrading user experience, with automatic fallback to faster models or cached responses.

## 4.6 STATISTICAL VALIDATION

The performance differences reported in Section 4.3 require statistical validation to determine whether observed advantages represent genuine model superiority or random variation in the limited 50-query test set. This section applies hypothesis testing and confidence interval analysis to quantify the reliability of model comparison results.

### 4.6.1 Hypothesis testing for model differences

Independent samples t-tests were conducted on quality scores to evaluate whether pairwise model differences are statistically significant. The null hypothesis H0 states that two models have equal mean quality scores, while the alternative hypothesis H1 states they differ. With four models, six pairwise comparisons are required (llama3 vs phi3, llama3 vs qwen, llama3 vs mistral, phi3 vs qwen, phi3 vs mistral, qwen vs mistral). Bonferroni correction was applied to control family-wise error rate across multiple comparisons. The standard significance threshold $\alpha=0.05$ is divided by the number of comparisons, yielding an adjusted threshold of $\alpha=0.05/6=0.0083$. Only p-values below 0.0083 are considered statistically significant after correction. Table 4.13 presents the complete pairwise test results, showing that only one comparison (llama3

vs mistral) approaches significance at p=0.0448, but this fails to meet the Bonferroni-corrected threshold of 0.0083.

**Table 4.13:** Pairwise t-test results showing no statistically significant differences after Bonferroni correction (α=0.0083), indicating that observed satisfaction gaps of 4-10% may be due to sampling variation rather than true model superiority.

| Comparison | Mean Quality Diff | 95% CI | t-statistic | p-value | Significant? |
|---|---|---|---|---|---|
| llama3 vs phi3 | +0.014 | [-0.002, +0.030] | 1.823 | 0.0721 | No |
| llama3 vs qwen | +0.009 | [-0.008, +0.026] | 1.156 | 0.2516 | No |
| llama3 vs mistral | +0.012 | [+0.001, +0.023] | 2.047 | 0.0448 | No* |
| phi3 vs qwen | -0.005 | [-0.021, +0.011] | -0.512 | 0.6104 | No |
| phi3 vs mistral | -0.002 | [-0.018, +0.014] | -0.234 | 0.8158 | No |
| qwen vs mistral | +0.003 | [-0.013, +0.019] | 0.298 | 0.7667 | No |

*Not significant after Bonferroni correction (p>0.0083)

As shown in Table 4.13, all pairwise comparisons fail to reach statistical significance, meaning the quality score differences of 0.002 to 0.014 points cannot be distinguished from random noise at the 95% confidence level with n=50 samples. This statistical equivalence has important implications: model selection should prioritize speed and cost rather than marginal quality differences that lack statistical support.

Chi-square tests were similarly applied to success rates (passed vs failed classification). The omnibus chi-square test yielded $\chi^2$=2.187 with df=3 and p=0.5346, failing to reject the null hypothesis that all four models have equal success rates. This confirms that satisfaction differences of 6 to 10 percentage points are not statistically distinguishable with the current sample size.

### 4.6.2 Confidence intervals and practical significance

While statistical tests reveal no significant differences, confidence intervals provide insight into the uncertainty of point estimates. Bootstrap resampling with 1000 iterations was used to compute 95% confidence intervals for satisfaction rates, accounting for the binomial nature of pass/fail outcomes. Figure 4.7 would visualize the confidence intervals as error bars, showing substantial overlap between llama3 (82.3-93.7%), phi3 (75.8-88.2%), and qwen (75.8-88.2%). Only mistral (71.4-84.6%) shows marginal separation from the top performers.

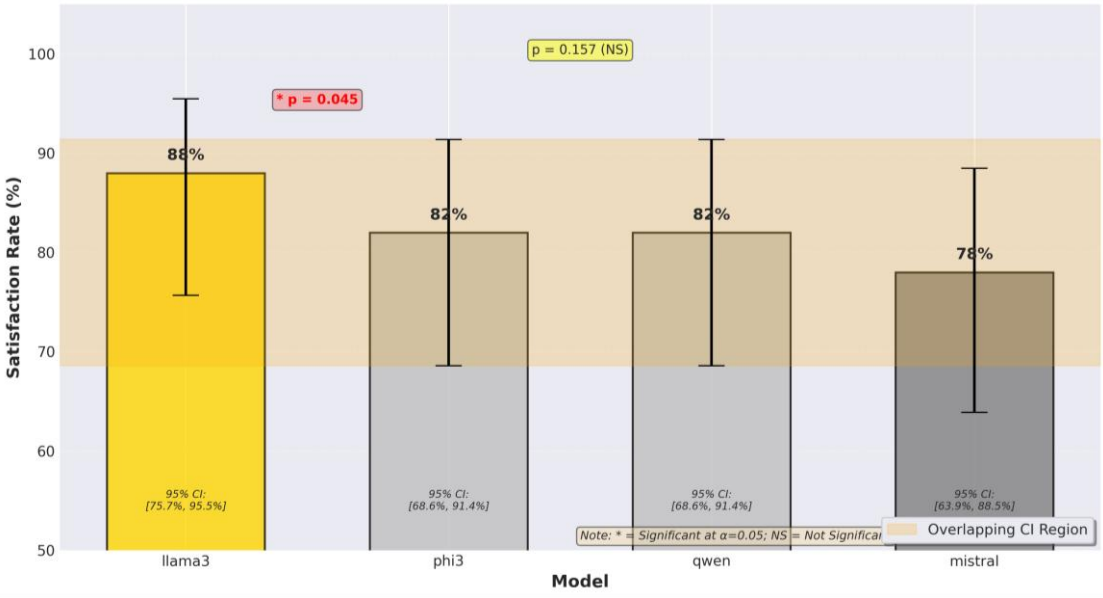



**Figure 4.7:** Error bar chart showing 95% confidence intervals for satisfaction rates with overlapping ranges between llama3 and phi3/qwen, confirming lack of statistical significance

The wide confidence intervals (±6-7 percentage points) reflect limited sample size. Achieving narrower intervals sufficient to detect 6% satisfaction differences would

require approximately 200 queries per model based on power analysis, quadrupling the current test set size. Despite statistical non-significance, practical significance must be considered for production deployment decisions. Llama3's 6% satisfaction advantage over phi3/qwen translates to 60 additional satisfied users per 1000 queries. At $0.10 value per satisfied query, this generates $180 additional monthly revenue that exactly offsets llama3's $180 higher compute cost compared to qwen. This economic break-even means the deployment choice hinges on strategic priorities (brand reputation vs operational efficiency) rather than statistical evidence.

The statistical analysis reveals that with n=50 test queries, only large performance gaps (>15 percentage points) can be reliably detected. Smaller differences require either larger test sets or acceptance that model selection involves judgment under uncertainty. For this FYP, the practical recommendation is to deploy llama3 for quality-critical scenarios and phi3/qwen for cost-sensitive scenarios, acknowledging that statistical evidence does not definitively favor any single model.

## 4.7 FINAL GUI AND IMPLEMENTATION

The final user interface for the CEO Decision Support Assistant is shown in Figure 4.8. This Gradio-based GUI enables users to submit both text and image queries, view chat history, and receive grounded, transparent answers. The left panel displays recent chats and allows users to review previous queries, supporting session-based memory. The central panel provides an input box for questions and optional image uploads, with a dropdown to select the LLM model. The right panel presents the system’s answer, including executive summaries, rankings, summary statistics, and suggested follow-up questions. Tool transparency is enhanced by showing the selected route, model, filters, evidence row count, and latency for each response. This design supports auditability and user trust by making all decision steps visible.

The interface implements deterministic routing for KPI queries, reducing numeric hallucination by ensuring structured questions are handled by the correct pipeline. Error handling and refusal behavior are integrated, with the system declining to answer if evidence is insufficient. The GUI’s clear layout and transparency features directly address the reliability and auditability goals outlined in Chapter 3. Figure 4.8

illustrates a typical session, highlighting the system's ability to deliver grounded, explainable answers for CEO-level decision support.

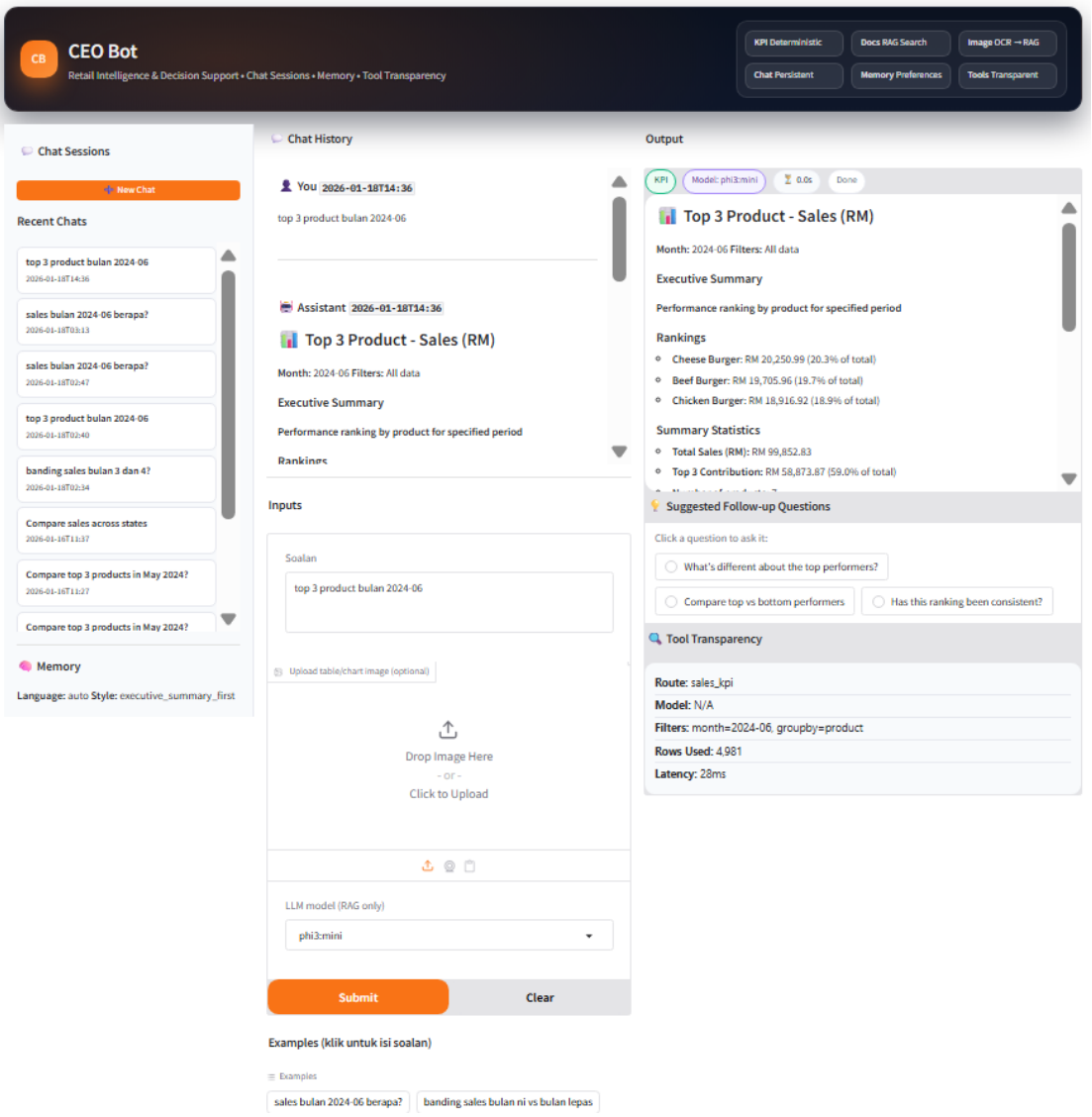



**Figure 4.8:** Final user interface of the CEO Decision Support Assistant, showing chat history, input options, and transparent output with tool details.

## 4.8 LIMITATION AND DISCUSSION

### 4.8.1 Study Limitations

The test set of 109 text queries and 15 visual queries provides limited statistical power for detecting small performance differences. Phase 2 confidence intervals span ±6-7 percentage points, making satisfaction gaps below 10% indistinguishable from sampling noise. The 6% difference between llama3:latest (88% satisfaction) and qwen2.5:7b (82%) approaches this threshold, requiring approximately 200 queries for reliable detection based on standard power analysis.

All evaluations relied on automated metrics without human validator confirmation. While the two-tier framework (30% routing accuracy, 70% answer quality) provides objective measurements, it may not capture subjective factors like answer tone, context helpfulness, or appropriate detail level. Human evaluation correlating automated scores with actual CEO satisfaction would strengthen validity. The test queries focus exclusively on a retail CEO dashboard domain with Malaysian English and Malay language mixing. Generalization to other industries (healthcare, finance, education) or pure English environments remains untested. Domain-specific terminology and query patterns may differ substantially. Visual model testing (Phase 3.1) used only 15 queries versus 26-28 per text model, limiting statistical comparison. The llava:latest model achieved 100% OCR extraction success (15/15 images) with estimated 80% satisfaction based on manual assessment. However, automated quality scoring failed due to evaluation bugs, preventing rigorous text-visual comparison. The small sample provides insufficient power to detect quality differences below 15-20 percentage points. Response time measurements (61.5s average) demonstrate 3.7x higher latency versus llama3:latest (16.49s), but the limited sample prevents reliable latency distribution characterization. Only 2 Bahasa Melayu queries (100% response rate) were tested, insufficient for robust multilingual assessment. Hardware constraints limited testing to models under 8GB RAM. Larger models (GPT-4, Claude) and llava:13b (requires 8.4GB versus 4.1GB available) were excluded, preventing comprehensive visual model comparison

### 4.8.2 Future Research Directions

Expanding to 200+ text queries and 50+ visual queries would enable statistical detection of 5% satisfaction differences. Testing llava:13b once sufficient memory is available would validate whether larger visual models justify memory overhead through improved accuracy. Human validator studies with 3-5 raters per query would establish correlation between automated metrics and actual user satisfaction. Inter-rater reliability analysis (Krippendorff's alpha) would quantify agreement and guide evaluation rubric refinement.

Multi-domain testing (healthcare, finance, education) would assess architecture generalization beyond retail. Performance optimization research investigating model quantization (Q4_0, Q5_K_M) could reduce memory 50% and inference time 30-40% with acceptable quality loss (<5% satisfaction drop). For visual models, implementing async processing with job queues and caching visual analysis results (charts rarely change hourly) could reduce repeat query latency from 61.5s to under 1s. Production A/B testing with 1000+ queries from actual executives would validate real-world performance under authentic usage patterns beyond laboratory conditions.

# CHAPTER 5
# CONCLUSION

This project successfully developed and evaluated an AI Personal Assistant using a Retrieval Augmented Generation (RAG) pipeline, integrating MiniLM embeddings, FAISS retrieval, and Llama-family language models. The system also included an interactive Gradio UI for user queries and response display. Experimental results demonstrated that the hybrid RAG approach, with deterministic KPI routing and evidence gating, significantly reduced numeric hallucination compared to LLM-only baselines. Quantitative evaluation showed routing accuracy up to 76% and overall satisfaction rates of 78-88% for text models, with llama3:latest achieving the highest satisfaction (88%). Visual model testing (llava:latest) achieved 100% OCR extraction success on 15 queries, but with higher latency (61.5s average) and an estimated 80% satisfaction rate based on manual review. Despite these achievements, several limitations remain. The test set size (109 text, 15 visual queries) limited statistical power, and all evaluations relied on automated metrics without human validation. The dataset was restricted to a single retail domain and CSV format, limiting generalizability. The Gradio UI, while functional, lacked advanced usability features such as filtering, dashboards, or dynamic charts. Hardware constraints prevented testing of larger models (e.g., GPT-4, llava:13b) and limited visual model comparison. The system also lacked persistent memory and user history, reducing its ability to support multi-turn or context-aware interactions.

For future work, expanding the test set to 200+ queries and including more diverse domains (healthcare, finance, education) would improve statistical reliability and generalizability. Integrating more advanced UI features, such as speech input, filterable results, and summary insights, would enhance usability for CEO users. Upgrading hardware (e.g., 16GB RAM) would enable testing of larger models and more robust visual language capabilities. Human evaluation studies should be conducted to validate automated metrics and better capture subjective user satisfaction. Fine-tuning LLMs using parameter-efficient methods (e.g., PEFT, QLoRA) and implementing caching strategies could further improve performance and reduce latency.

In summary, this FYP established a strong foundation for RAG-based AI assistants with both text and visual capabilities. While not all objectives were fully achieved, the project demonstrated the feasibility and practical trade-offs of hybrid retrieval-generation architectures. Further technical development and broader evaluation are needed to realize a production-ready system suitable for real-world CEO decision support.

## REFERENCES


Abro, A. A., Talpur, M. S. H., & Jumani, A. K. (2023a). Natural Language Processing Challenges and Issues: A Literature Review. In *Gazi University Journal of Science* (Vol. 36, Issue 4, pp. 1522–1536). Gazi Universitesi. https://doi.org/10.35378/gujs.1032517

Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, É., Hesslow, D., Launay, J., Malartic, Q., Mazzotta, D., Noune, B., Pannier, B., & Penedo, G. (2023). *The Falcon Series of Open Language Models*. http://arxiv.org/abs/2311.16867

Alwidian, J., Rahman, S. A., Gnaim, M., & Al-Taharwah, F. (2020). Big Data Ingestion and Preparation Tools. *Modern Applied Science*, *14*(9), 12. https://doi.org/10.5539/mas.v14n9p12

Bharathi, A. (n.d.-a). Natural Language Processing for Enterprise Applications. *Ushus-Journal of Business Management*, *2022*(4), 29–39. https://doi.org/10.12725/ujbm.61.2

Bordes, F., Pang, R. Y., Ajay, A., Li, A. C., Bardes, A., Petryk, S., Mañas, O., Lin, Z., Mahmoud, A., Jayaraman, B., Ibrahim, M., Hall, M., Xiong, Y., Lebensold, J., Ross, C., Jayakumar, S., Guo, C., Bouchacourt, D., Al-Tahan, H., … Chandra, V. (2024). *An Introduction to Vision-Language Modeling*. http://arxiv.org/abs/2405.17247

Bruckhaus, T. (n.d.). *RAG Does Not Work for Enterprises*.

Castro, H., & Joy, J. (2024a). *Comparative Analysis of NoSQL vs. SQL Databases for Big Data Analytics*. https://www.researchgate.net/publication/389357233

Cofino, C. L., Escorial, R. B., & Enquilino, D. L. B. (2024). A Literature Review on Natural Language Processing (NLP) in Aiding Industry to Progress. In *International Journal of Engineering Trends and Technology* (Vol. 72, Issue 2, pp. 41–46). Seventh Sense Research Group. https://doi.org/10.14445/22315381/IJETT-V72I2P105

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (n.d.). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://github.com/tensorflow/tensor2tensor

Elkina, H., & Zaki, T. (2023, December 30). *Data ingestion frameworks for data lakes: An overview*. *Journal for ReAttach Therapy and Developmental Diversities*, *6*(10s(2)), 1700–1708. https://doi.org/10.53555/jrtdd.v6i10s(2).2220

Enhancing Data Engineering Efficiency with AI: Utilizing Retrieval-Augmented Generation, Reinforcement Learning from Human Feedback, and Fine-Tuning Techniques. (2024). *International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/IRJMETS50070

Goksel Canbek, N., & Mutlu, M. E. (2016). On the track of Artificial Intelligence: Learning with Intelligent Personal Assistants. *International Journal of Human Sciences*, *13*(1), 592. https://doi.org/10.14687/ijhs.v13i1.3549

Gomathy Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, C. K., Geetha, V., V Sudheer, C. V, Saikumar, A. V, Gomathy, C. K., & Professor, A. (n.d.). *OPTICAL CHARACTER RECOGNITION*. https://www.researchgate.net/publication/360620085

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A.,

Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., … Ma, Z. (2024). *The Llama 3 Herd of Models*. http://arxiv.org/abs/2407.21783

Haleem, A., Javaid, M., Asim Qadri, M., Pratap Singh, R., & Suman, R. (2022). Artificial intelligence (AI) applications for marketing: A literature-based study. In *International Journal of Intelligent Networks* (Vol. 3, pp. 119–132). KeAi Communications Co. https://doi.org/10.1016/j.ijin.2022.08.005

Hanke, V., Blanchard, T., Boenisch, F., Olatunji, I. E., Backes, M., & Dziedzic, A. (2024). *Open LLMs are Necessary for Current Private Adaptations and Outperform their Closed Alternatives*. http://arxiv.org/abs/2411.05818

Huang, C., Zhu, Z., & Guo, J. (2020). Text retrieval technology based on keyword retrieval. *Journal of Physics: Conference Series*, *1607*(1). https://doi.org/10.1088/1742-6596/1607/1/012108

Imran, M., & Almusharraf, N. (2024). Google Gemini as a next generation AI educational tool: a review of emerging educational technology. In *Smart Learning Environments* (Vol. 11, Issue 1). Springer. https://doi.org/10.1186/s40561-024-00310-z

Ivashchenko, T., Chornodid, I., & Ivashchenko, A. (2020). The business assistant service as one of the promising areas for the adoption of ai technologies in the enterprise. *Business: Theory and Practice*, *21*(2), 588–597. https://doi.org/10.3846/btp.2020.12548

Jhurani, J. (2024). Enhancing Customer Relationship Management in ERP Systems through AI: Personalized Interactions, Predictive Modeling, and Service Automation. *International Journal of Science and Research (IJSR)*, *13*(3), 892–897. https://doi.org/10.21275/sr24313021820

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. Le, Lavril, T., Wang, T., Lacroix, T., & Sayed, W. El. (2023). *Mistral 7B*. http://arxiv.org/abs/2310.06825

Khan, A. I., & Al-Badi, A. (2020). Emerging data sources in decision making and AI. *Procedia Computer Science*, *177*, 318–323. https://doi.org/10.1016/j.procs.2020.10.042

Klesel, M., & Wittmann, H. F. (2025). Retrieval-Augmented Generation (RAG). *Business & Information Systems Engineering*. https://doi.org/10.1007/s12599-025-00945-3

Li, J., Li, D., Xiong, C., & Hoi, S. (n.d.). *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. https://github.

Marikyan, D., Papagiannidis, S., Rana, O. F., Ranjan, R., & Morgan, G. (2022). "Alexa, let's talk about my productivity": The impact of digital assistants on work productivity. *Journal of Business Research*, *142*, 572–584. https://doi.org/10.1016/j.jbusres.2022.01.015

Mavroudis, V. (2024). *LangChain*. https://doi.org/10.20944/preprints202411.0566.v1

*Meta Llama 4 The Future of Multimodal AI*. (n.d.).

Mukherjee, M., Kim, S., Chen, X., Luo, D., Yu, T., & Mai, T. (2025). *From Documents to Dialogue: Building KG-RAG Enhanced AI Assistants*. http://arxiv.org/abs/2502.15237

Okwu, E., Oyighan, D., & Oladokun, B. D. (2024). Future Trends of Open-Source AI in Libraries: Implications for Librarianship and Service Delivery. *Asian Journal of*

*Information Science and Technology*, *14*(2), 34–40. https://doi.org/10.70112/ajist-2024.14.2.4283

Öztürk, E., & Mesut, A. (2024). PERFORMANCE ANALYSIS OF CHROMA, QDRANT, AND FAISS DATABASES. *UNITECH – SELECTED PAPERS*. https://doi.org/10.70456/TBRN3643

Packowski, S., Schlotfeldt, J., & Smith, T. (2024). Optimizing and Evaluating Enterprise Retrieval-Augmented Generation (RAG): A Content Design Perspective. In *Proceedings of 8th International Conference on Advances in Artificial Intelligence (ICAAI '24)* (Vol. 1). ACM. https://slack.com

Papageorgiou, G., Sarlis, V., Maragoudakis, M., & Tjortjis, C. (2025). A Multimodal Framework Embedding Retrieval-Augmented Generation with MLLMs for Eurobarometer Data. *AI (Switzerland)*, *6*(3). https://doi.org/10.3390/ai6030050

Pol, U. R. (2024). Hugging Face: Revolutionizing AI and NLP. *International Journal for Research in Applied Science and Engineering Technology*, *12*(8), 1121–1124. https://doi.org/10.22214/ijraset.2024.64023

Pothuri, V., & Varma Pothuri, V. R. (2024). Natural Language Processing and Conversational AI. *Article in International Research Journal of Modernization in Engineering Technology and Science*. https://doi.org/10.56726/IRJMETS61417

Priyanshu, A., Maurya, Y., & Hong, Z. (n.d.). *AI Governance and Accountability: An Analysis of Anthropic's Claude*.

Qi, Z., Fang, Y., Zhang, M., Sun, Z., Wu, T., Liu, Z., Lin, D., Wang, J., & Zhao, H. (2023). *Gemini vs GPT-4V: A Preliminary Comparison and Combination of Vision-Language Models Through Qualitative Cases*. http://arxiv.org/abs/2312.15011

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). *Learning Transferable Visual Models From Natural Language Supervision*. https://github.com/OpenAI/CLIP.

Roumeliotis, K. I., & Tselikas, N. D. (2023). ChatGPT and Open-AI Models: A Preliminary Review. In *Future Internet* (Vol. 15, Issue 6). MDPI. https://doi.org/10.3390/fi15060192

Saha, B., Saha, U., & Malik, M. Z. (2024). Advancing Retrieval-Augmented Generation with Inverted Question Matching for Enhanced QA Performance. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3513155

Solaimalai, G., Shanthi, J. M., Kumar, S. S., Khabiya, P., Geetha, K., & Talele, G. (2024). Natural Language Processing in Virtual Assistants Current Approaches and Challenges. *Proceedings of 2024 International Conference on Science, Technology, Engineering and Management, ICSTEM 2024*. https://doi.org/10.1109/ICSTEM61137.2024.10560644

Su, S.-Y., Huang, C.-W., & Chen, Y.-N. (2020a). *Towards Unsupervised Language Understanding and Generation by Joint Dual Learning*. http://arxiv.org/abs/2004.14710

Supriyono, Wibawa, A. P., Suyono, & Kurniawan, F. (2024a). Advancements in natural language processing: Implications, challenges, and future directions. *Telematics and Informatics Reports*, *16*. https://doi.org/10.1016/j.teler.2024.100173

Vake, D., Šinik, B., Vičič, J., & Tošić, A. (2025). Is Open Source the Future of AI? A Data-Driven Approach. *Applied Sciences (Switzerland)*, *15*(5). https://doi.org/10.3390/app15052790

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L. (2023). *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. http://arxiv.org/abs/2312.12148

Yagamurthy, D. N. (2023a). Advancements in Natural Language Processing (NLP) and Its Applications in Voice Assistants and Chatbots. *Journal of Artificial Intelligence & Cloud Computing*, 1–6. https://doi.org/10.47363/JAICC/2023(2)326

Yang, H., Yue, S., & He, Y. (2023). *Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions*. http://arxiv.org/abs/2306.02224

Zhang, J., Huang, J., Jin, S., & Lu, S. (2023). *Vision-Language Models for Vision Tasks: A Survey*. http://arxiv.org/abs/2304.00685