Retrieval Augmented Generation for Automating Enterprise Technical Queries

Nyabuti, Sharon Kemunto

168403

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Data Science and Analytics at Strathmore University

Strathmore Institute of Mathematical Sciences

Strathmore University

Nairobi, Kenya

June, 2025

Declaration and Approval

I declare that this work has not been previously submitted and approved for award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Name: **Nyabuti Sharon Kemunto**

Signature: Sharon K

Date: May 23, 2025

Approval

The dissertation of Nyabuti Sharon Kemunto was reviewed and approved by:

Allan Omondi, Ph.D,
Lecturer, Strathmore School of Computing,
Strathmore University.

Dr. Godfrey Achono Madigu,
Dean, Strathmore Institute of Mathematical Sciences,
Strathmore University.

Prof. Bernard Shibwabo Kasamani,
Director of Graduate Studies,
Strathmore University.

Abstract

The growing complexity of enterprise IT systems, coupled with an increasing volume of technical support queries, continues to burden support teams, especially during high-demand periods. A significant portion of these queries are repetitive, leading to agent fatigue, delayed resolutions, and reduced customer satisfaction.

This study presents a Retrieval-Augmented Generation (RAG) system for automating the resolution of enterprise technical queries. The system integrates a retrieval module that identifies relevant content from a domain-specific knowledge base and a generative module that produces contextually appropriate responses using pre-trained language models.

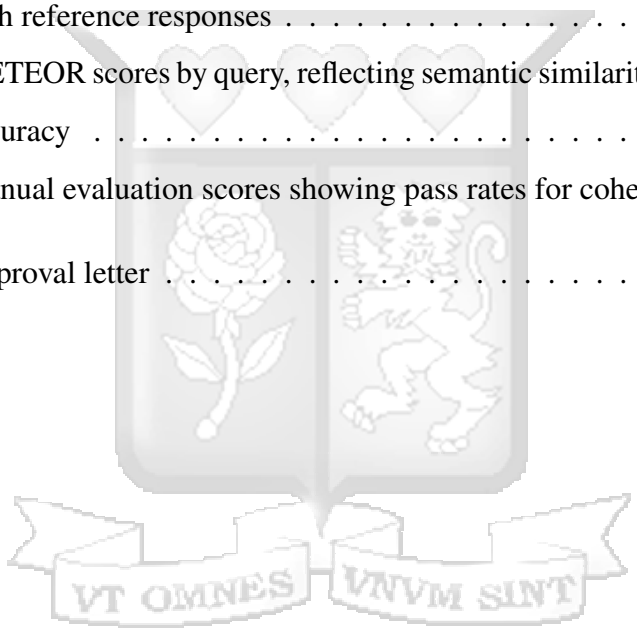
Its performance was evaluated using Bilingual Evaluation Understudy (BLEU) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE-L) metrics, alongside expert qualitative assessments. Results demonstrate that the RAG-based approach improves response quality and fluency, reduces manual workload, and accelerates resolution times.

This study demonstrates the practical value of using RAG systems to automate repetitive support tasks in enterprise environments.

Keywords: Natural Language Processing (NLP), Information Retrieval (IR), Retrieval-Augmented Generation (RAG), IT Support Systems, Generative AI.

Table of contents

List of figures

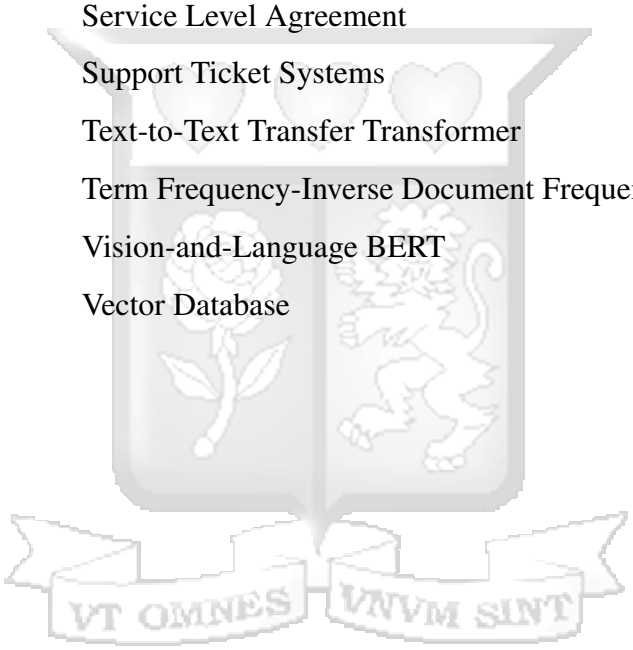


List of tables

List of Abbreviations and Acronyms

AI	Artificial Intelligence
AIML	Artificial Intelligence Mark-up Language
ANN	Approximate Nearest Neighbor
ANCE	Approximate Closest Neighbor Contrast Learning
BERT	Bidirectional Encoder Representations Transformers
BM25	Best Match 25
ChatGPT	Chat Generative Pre-Trained Transformer
CLERC	Case Law Evaluation and Retrieval Corpus
DBMS	Database Management System
DIRAS	Domain-specific Information Retrieval Annotation with Scalability
DocIDs	Document Identifiers
DPO	Dialogue Processing
DPR	Dense Passage Retrieval
DST	Dialogue State Tracking
E5	Efficient and Effective Embedding for Dense Retrieval
FAQ	Frequently Asked Questions
Gen AI	Generative AI
GIR	Generative Information Retrieval
GPT	Generative Pre-Trained Transformers
IDF	Inverse Document Frequency
IR	Information Retrieval
KL	Kullback-Leibler Divergence
LLM	Large Language Model
MML	Multimodal Machine Learning
MRR	Mean Reciprocal Rank
NDCG	Normalized Discounted Cumulative Gain
NLG	Natural Language Generation

NLP	Natural Language Processing
NLU	Natural Language Understanding
QA	Question Answering
RAG	Retrieval-Augmented Generation
RL	Reinforcement Learning
RNN	Recurrent Neural Networks
RRF	Reciprocal Rank Fusion
Seq2Seq	Sequence-to-Sequence
SimLM	Similarity Learning with Language Models
SLA	Service Level Agreement
STS	Support Ticket Systems
T5	Text-to-Text Transfer Transformer
TF-IDF	Term Frequency-Inverse Document Frequency
ViLBERT	Vision-and-Language BERT
VecDB	Vector Database



Definition of Terms

Artificial Intelligence (AI) – The simulation of human intelligence processes by machines. In this thesis, it includes the use of machine learning models such as large language models and generative AI to enhance automated technical support (Fuchs et al., 2022).

BM25 (Best Match 25) – A sparse retrieval algorithm that ranks documents based on term frequency and document relevance. Effective for handling routine queries efficiently (Thakur et al., 2021).

Dense Retrieval – A method that transforms queries and documents into dense vectors for semantic similarity comparison, improving retrieval of complex or paraphrased queries (Karpukhin et al., 2020).

Generative AI – AI systems that produce new content such as text or code using large-scale models. Enhances response generation in technical support applications (**?**).

Generative Information Retrieval (GIR) – Combines retrieval and generation by using models like T5 and BART to generate documents or responses directly from queries (Lewis et al., 2020).

Large Language Model (LLM) – Deep learning models trained on large text corpora capable of understanding and generating natural language, such as GPT-4 and Mistral (Radford et al., 2021).

Query Augmentation – The expansion of queries by appending contextually relevant terms, improving document recall and retrieval accuracy (Khan et al., 2019).

Retrieval-Augmented Generation (RAG) – A hybrid framework combining document retrieval and text generation to produce accurate and context-aware answers (Lewis, 2019).

Sparse Retrieval – Traditional keyword-based document retrieval using models like TF-IDF and BM25. Fast but limited in understanding query intent (Paulin et al., 2024).

Technical Support Ticket – A formal request for assistance logged by users, containing issue descriptions and timestamps, and managed within support systems (Deshai et al., 2024).

Acknowledgment

First and above all, I am grateful to the Almighty God for His provision, grace, and for granting me good health throughout the study period.

I am grateful to my parents, brothers and sisters for their immeasurable support throughout the study period.

I am indebted to my supervisors, Dr. Allan Omondi and Dr. Kennedy Senagi, for their availability, kind guidance, and support without which this dissertation would not have been a success.

I am grateful to my classmates for their support and encouragement.

Dedication

This dissertation is dedicated to **God Almighty** for giving me wisdom and good health.

Chapter 1

Introduction

1.1 Background to the Study

Technical support is critical in shaping the customer experience in enterprise software development (Al-Hawari et al., 2021). Many organizations have implemented Support Ticket Systems (STSs) to facilitate communication between users and incident-resolving personnel. These systems enable customers to report problems or requests, providing a structured channel to resolve issues efficiently. are widely used across various industries and are the primary communication tool to ensure service continuity and rapid problem resolution.

However, large IT organizations increasingly struggle with the growing volume and diversity of support tickets. During peak service periods or outages, these companies often become overwhelmed by the influx of numerous tickets, leading to delays in resolution. Such delays negatively impact customer satisfaction and can even result in business losses (Gupta and Sengupta, 2012). Efficient ticket handling is crucial, as many of these issues are time-sensitive and demand immediate attention.

Support tickets often contain valuable insights for resolving individual issues and improving service delivery. Over time, these tickets accumulate into large datasets that, when properly analyzed, can reveal trends, recurring problems, and customer behavior patterns. Unfortunately, the manual extraction of relevant information from these datasets is time-consuming, hampers the timely resolution of issues. Research has shown that many support tickets consist of repetitive and straightforward inquiries; for example, Mercedes-Benz Consulting found that 80% of inquiries in the automotive sector are repetitive (Gentsch, 2018). This pattern of repetitive queries is not unique to the automotive industry but is shared across various sectors, representing a significant inefficiency in many support systems.

This repetition in support tasks also has a detrimental effect on employee engagement. According to a report by (Nguyen, 2022), 63% of employees worldwide report low engagement with their work, mainly due to the monotonous nature of tasks like ticket resolution. This lack of engagement stifles creativity and innovation in the workplace, as employees become demotivated by the repetitive aspects of their roles.

Table 1.1 illustrates employee engagement data from China, highlighting the scale of this issue.

Table 1.1: Employee engagement (Marin, 2021)

Category	**2009-2010**	**2011-2012**
Actively disengaged	27%	24%
Not engaged	62%	63%
Engaged	11%	13%

AI-driven support systems present a promising solution to these challenges. By automating routine tasks and handling repetitive queries, AI-powered support systems allow human agents to focus on more complex and creative tasks. This approach improves ticket resolution efficiency and employee engagement by reducing their participation in monotonous activities. In addition, it significantly shortens ticket turnaround times, helping departments meet their service level agreements (SLA) more consistently (Fuchs et al., 2022).

1.2 Statement of the Problem

The growing complexity of IT systems and the increasing reliance on digital services have led to a significant increase in support ticket volume in IT companies, increasing by approximately 15% per year over the last five years (Nguyen, 2022). Current IT support systems are struggling to handle this continuous growth, particularly during periods of high demand, when the repetitive nature of many queries adds further strain on support teams. As a result, human agents experience fatigue, response times are delayed, and ticket resolutions vary in quality, factors that ultimately erode customer satisfaction.

The core issue arises from the limitations of existing IT support systems that inefficiently retrieve relevant information to resolve tickets, especially those involving complex technical queries. Despite implementing knowledge bases and automated support features, IT teams face delays and inconsistencies in ticket resolution due to inadequate retrieval of contextually appropriate information on demand (Deshai et al., 2024). This challenge is particularly severe within large IT organizations, where complicated system environments require rapid and precise responses to ensure continuous and reliable service.

Companies with complex infrastructures and high support volumes, especially those involved in large-scale software and technical services, are most affected by this problem. Since 2018, the rapid expansion of digital services has intensified the need for advanced support solutions that can keep up with increasingly diverse user demands and complex queries. Attempts to address this problem through knowledge management systems, static FAQs, and machine learning-based ticket classification have provided partial relief Crosley and Wasito (2023). However, these methods still fail to allow efficient retrieval of precise information needed for complex issues, often resulting in slower response times and increased stress on support staff.

These limitations impact primary stakeholders, including IT support engineers, who need rapid access to relevant information to provide timely and accurate assistance, and customers, who rely on prompt resolutions. Secondary stakeholders, such as business clients and organizational leaders, face operational disruptions and reduced customer satisfaction when technical issues are not resolved promptly.

1.3 Research Objectives

1.3.1 General Objective

The main objective of this study is to develop a Retrieval-Augmented Generation (RAG) system that provides accurate, timely, and contextually relevant responses to user inquiries in an enterprise customer support system.

1.3.2 Specific Objectives

i) Review current customer support approaches to identify how an automated RAG-based solution can improve existing support channels.

ii) Develop a RAG system using a pre-trained large language model (LLM), GPT-4, to retrieve relevant content and generate accurate, context-aware responses.

iii) Evaluate the system's performance based on accuracy, relevance, and effectiveness in enterprise technical support scenarios.

1.4 Research Questions

i) How can current customer support approaches be improved using a RAG-based solution?

ii) How can a RAG system using GPT-4 be implemented to retrieve relevant content and generate accurate technical support responses?

iii) How effective is the RAG system in improving accuracy, relevance, and user experience in enterprise technical support scenarios?

1.5 Justification of the Study

This study is essential for improving the efficiency, accuracy, of technical support in an enterprise environment.

Support Teams: Automating response generation reduces workload, improves ticket resolution times, and enables teams to handle more queries with greater consistency.

Customers: Faster and more accurate responses enhance user experience and satisfaction by resolving issues promptly and contextually.

Business Leaders: Automation lowers support costs and provides actionable insights that inform product development and service improvement.

System Integrity: Unlike general-purpose tools like ChatGPT, the proposed RAG system operates securely within organizational boundaries, reflecting the latest internal documentation, offering accurate, real-time responses that public models cannot guarantee.

1.6 Scope and Limitations

1.6.1 Scope

The research focuses on developing and implementing an automated support system within an enterprise IT environment to handle diverse support issues. This includes:

i. **Knowledge Extraction:** Automated extraction of relevant information from an enterprise's internal technical documentation.

ii. **Response Generation:** Dynamic generation of context-aware responses to improve resolution times and accuracy.

iii. **Performance Evaluation:** The study will assess the system's impact on accuracy and user satisfaction.

1.6.2 Limitations

Ambiguous User Queries: The system may face challenges when handling vague, unclear, or poorly structured queries.

Scalability: Although designed for real-time operation, system performance may decline under high query volumes without appropriate infrastructure scaling.

System Performance and Maintenance Overhead: Response accuracy is highly dependent on retrieval quality. Additionally, managing multiple integrated components introduces technical debt and requires continuous maintenance and optimization.

Chapter 2

Literature Review

2.1 Introduction

Information retrieval (IR) is a critical field dedicated to organizing, retrieving, and presenting information from large datasets, serving as the backbone of modern search engines, digital libraries, and AI-based applications such as chatbots and virtual assistants (Mitra and Craswell, 2019). Recent developments in IR have expanded traditional keyword-based techniques to incorporate advanced neural ranking models, improving the relevance and accuracy of search results through machine learning and deep learning (Guo et al., 2019).

Integrating sparse and dense retrieval methods is fundamental in contemporary IR systems, particularly within Retrieval Augmented Generation (RAG) frameworks. RAG systems combine retrieval capabilities with generative models ensuring the responses are accurate and relevant.(Lewis et al., 2020). Section 2.2.1, Sparse Retrieval, covers classical techniques such as rule-based and statistical methods, which leverage keyword matching and frequency-based models to retrieve relevant content efficiently. Section 2.2.2, Dense Retrieval, explores neural embeddings for semantic search, where dense vector representations map queries and documents into a shared semantic space, allowing for more nuanced and context-aware retrieval. Finally, Section 2.2.3, Generative Information Retrieval, introduces advanced techniques like query augmentation and response generation, enabling IR systems to go beyond traditional retrieval by producing contextually enriched and highly relevant responses, advancing the capabilities of AI-driven applications.

2.2 Information Retrieval Theory

Information Retrieval (IR) theory is centered on identifying and retrieving relevant information from vast datasets, with a primary goal of enhancing both the effectiveness and efficiency of retrieval systems. Modern IR systems extend beyond essential keyword matching, incorporating techniques that interpret the context and intent behind user queries to produce results more closely aligned with user expectations (Meng et al., 2023). As data volumes grow exponentially, there is an increasing need for IR systems that adapt to varying user needs and query types, utilizing advanced models and algorithms to handle the diversity and complexity of modern data landscapes (Zhang et al., 2021).

Many Information Retrieval (IR) systems have traditionally relied on sparse retrieval methods, which focus on exact keyword matches between user queries and documents. A commonly used technique in this domain is the vector space model, which employs metrics such as term frequency-inverse document frequency (TF-IDF) to rank documents based on keyword relevance (Prabhu and Anand, 2024). Although sparse retrieval methods can be effective for straightforward queries, they frequently struggle to interpret complex language, including synonyms, paraphrases, and variations in phrasing that express the same intent.

Researchers have started using more advanced methods that incorporate semantic understanding and machine learning to overcome these limitations. Dense retrieval techniques, for example, create vector representations of words and documents, allowing the system to recognize relationships between words and understand their meanings more profoundly (Karpukhin et al., 2020). These improvements help systems better grasp what users intend with their queries, even when the language is ambiguous. As the field of IR continues to evolve, understanding these foundational principles is crucial for building future systems that effectively combine retrieval and generation capabilities, ultimately enhancing user experiences across various applications.

2.2.1 Sparse Retrieval

Sparse retrieval refers to traditional information retrieval methods that rely on specific keywords or terms in documents and user queries to determine relevance. These methods often utilize high-dimensional vectors, where each dimension corresponds to a unique term in the dataset, to represent the textual data. One of the fundamental techniques in sparse retrieval is the inverted index, which efficiently maps each term to a list of documents containing that term, enabling quick lookup in extensive document collections.

Among the various sparse retrieval techniques, Term Frequency-Inverse Document Frequency (TF-IDF) is widely used to assess the importance of words in documents relative to a user's query. The relevance score $R(q,d)$ between a query q and a document d is calculated by aggregating the TF-IDF weights of terms standard to both the query and the document, as shown in the equation below.

$$R(q,d) = \sum_{t \in q \cap d} \text{tf-idf}(t,d) \cdot \text{tf-idf}(t,q), \tag{2.1}$$

where:

- t represents the terms common to both the query q and the document d.
- $\text{tf-idf}(t,d)$ is the TF-IDF weight of the term t in the document d.
- $\text{tf-idf}(t,q)$ is the TF-IDF weight of the term t in the query q.

The weight of TF-IDF is calculated as the product of two components: Term Frequency (TF) and Inverse Document Frequency (IDF), defined as follows:

$$\text{tf}(t,d) = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in document } d}, \tag{2.2}$$

$$\text{idf}(t) = \log\left(\frac{N}{n_t}\right), \tag{2.3}$$

where:

- N is the total number of documents in the collection.
- n_t is the number of documents in which the term t appears.

The product of these two components gives the TF-IDF score:

$$\text{tf-idf}(t,d) = \text{tf}(t,d) \cdot \text{idf}(t). \tag{2.4}$$

This formula ensures that terms that appear frequently in a specific document but not in many others (high TF and low DF) are given greater importance in determining relevance.

Although sparse retrieval methods like TF-IDF and Best Match 25 (BM25) excel at quickly retrieving documents, they often struggle to understand the full context and intent behind a user's query, especially when dealing with synonyms, specialized terminology, or complex queries. This limitation stems from their reliance on exact term matching, which may not fully capture the meaning of user language(Luan et al., 2021; Thakur et al., 2021)

Rule-Based Retrieval

Rule-based retrieval is a method commonly used in information retrieval systems that relies on predefined rules or patterns to identify relevant documents based on user queries. These rules are often crafted based on domain knowledge, allowing systems to map queries to specific terms or document attributes that match those rules. For example, if a user searches for "technical support," a rule-based system might prioritize documents containing terms like "helpdesk," "assistance," or "troubleshooting," ensuring a more precise match (Sansone and Sperlí, 2022).

Rule-based retrieval is particularly effective in structured environments, such as legal databases, customer support systems, or specialized knowledge repositories, where vocabulary and terminology are controlled and consistent. These systems can efficiently deliver accurate results when queries and documents follow predictable patterns(Zhang et al., 2021).

However, the effectiveness of rule-based retrieval diminishes in dynamic and diverse contexts. When user queries become more varied, relying solely on fixed rules may lead to gaps in performance, as the system might not recognize synonyms or paraphrases that convey similar meanings. For instance, a rule designed to respond to the term "repair" might not catch variations like "fix" or "restore," resulting in missing relevant documents. As language evolves, maintaining and updating the rules becomes increasingly resource-intensive, requiring frequent updates to remain effective (Wu et al., 2022). This limitation highlights the need for integrating more adaptive and context-aware approaches that leverage natural language processing (NLP) and machine learning techniques to complement or replace traditional rule-based methods.

Transaction-Based Retrieval

Rule-based retrieval is a method within sparse retrieval that connects user queries to relevant documents based on predefined rules. These rules often rely on fixed patterns from expert knowledge or historical data to determine document relevance according to specific keywords. For example, when a user enters the term “repair,” the system prioritizes documents containing related terms like “fix,” “maintenance,” or “service.” This approach is efficient in structured environments such as technical support databases or specialized knowledge bases where the vocabulary is consistent and well-defined (Sansone and Sperlí, 2022).

Rule-based retrieval often struggles in dynamic or diverse contexts despite its effectiveness in structured settings. As user queries become more varied, relying solely on fixed rules can lead to gaps in performance. For instance, if a user rephrases a query, like asking “ways to fix a broken appliance,” instead of simply using the word “repair,” the rule-based system might miss relevant documents that don’t match the exact keywords. Additionally, as language evolves and users adopt new terms or expressions, maintaining and updating the rule set becomes resource-intensive, underlining the need for more adaptive retrieval methods that handle the nuances of natural language more effectively (Gupta et al., 2024).

2.2.2 Dense Retrieval

Dense retrieval is a modern approach to information retrieval that uses advanced techniques to enhance the accuracy and relevance of search results. Unlike traditional sparse retrieval methods, which rely on exact keyword matches, dense retrieval focuses on understanding word meanings and relationships. This semantic approach is particularly beneficial for handling complex queries and natural language, as it captures semantic similarities rather than surface-level matches. By employing machine learning and deep learning techniques, dense retrieval systems can analyze vast amounts of data to identify relevant documents based on contextual understanding (Karpukhin et al., 2020).

The advent of pre-trained language models such as BERT (Devlin, 2018) has revolutionized information retrieval, paving the way for the development of various dense retrieval methods, including dense passage retrieval (DPR) (Karpukhin et al., 2020), approximate closest neighbor contrast learning (ANCE) (Xiong et al., 2020), efficient and effective embeddings for dense retrieval (E5) (Wang et al., 2022b), and similarity learning with language models (SimLM) (Wang et al., 2022a). These methods leverage Transformer-based encoders to generate dense vector representations for queries and documents, enhancing the ability to grasp underlying semantics and improving retrieval accuracy.

The core of dense retrieval lies in transforming documents and queries into vector representations. Given a document d and a query q, we denote their vector representations as $\mathbf{v}_d$ and $\mathbf{v}_q$, respectively. Each document d is transformed into a dense vector $\mathbf{v}_d$ through a pre-trained language model, and the same process is applied to the query q to obtain vector $\mathbf{v}_q$. The encoding process can be represented using the following functions:

$$\mathbf{v}_d = E_d(d), \quad \mathbf{v}_q = E_q(q), \tag{2.5}$$

Where $E_d(\cdot)$ and $E_q(\cdot)$ represent the encoder functions for documents and queries, respectively. These functions can utilize the same or different models optimized for specific tasks.

Dense retrieval methods evaluate relevance by calculating the similarity between the query vector $\mathbf{v}_q$ and the document vector $\mathbf{v}_d$. This similarity is commonly measured using cosine similarity, expressed as follows:

$$R(q,d) = \cos(\mathbf{v}_q, \mathbf{v}_d) = \frac{\mathbf{v}_q \cdot \mathbf{v}_d}{\|\mathbf{v}_q\| \|\mathbf{v}_d\|}, \tag{2.6}$$

where $\mathbf{v}_q \cdot \mathbf{v}_d$ represents the dot product of the query vector $\mathbf{v}_q$ and the document vector $\mathbf{v}_d$, and $\|\mathbf{v}_q\|$ and $\|\mathbf{v}_d\|$ denote the magnitudes of the query and document vectors, respectively.

Finally, documents are ranked based on these similarity scores to identify the most relevant ones for the user, facilitating a more accurate retrieval process.

Semantic-Based Retrieval

Semantic-based retrieval is a subset of dense retrieval that emphasizes the importance of meaning in the retrieval process. This method employs Natural Language Processing (NLP) techniques to analyze the semantics of both the query and the documents, enabling the system to understand the underlying intent of the user. For example, if a user searches for "automobile repair," a semantic-based retrieval system might also identify and prioritize documents containing related phrases like "car maintenance" or "vehicle service," even if those exact terms do not appear in the query. By capturing the context and meaning, semantic-based retrieval enhances the accuracy and relevance of search results, aligning closely with user intent and improving the overall search experience(Martinez-Rodriguez et al., 2020; Zhu et al., 2021)

Embedding (Vector Representation) Embedding, or vector representation, is a fundamental technique in dense retrieval systems that transforms words or phrases into numerical vectors, allowing the system to capture the semantic relationships between terms. In a vector space, words or phrases with similar meanings are positioned closer together, enabling the retrieval system to identify relevant documents based on their proximity. Modern techniques like BERT (Bidirectional Encoder Representations from Transformers) and Sentence-BERT have

advanced the creation of embeddings by leveraging deep learning models trained on large text corpora, capturing more nuanced aspects of language and context than earlier methods like Word2Vec and GloVe (Bölücü et al., 2023). These embeddings form the foundation for dense retrieval systems, enhancing their ability to assess the semantic similarity between user queries and documents, thereby improving retrieval performance and relevance (Luan et al., 2021; Xiong et al., 2020).

Vector Databases Vector databases (VecDBs) play a crucial role in dense retrieval by efficiently storing and managing embeddings generated for various documents and queries. Unlike traditional database management systems (DBMS), developed over the past 60 years to handle structured data, vector databases are specifically designed for high-dimensional vector data produced by deep learning models. These embeddings represent the semantic features of unstructured multi-modal data, such as images and text, allowing for effective representation and retrieval of relevant information.

Fast Similarity Searches VecDBs enable fast similarity searches through embedding indexing. This allows quickly identifying the most similar vectors to a user's query vector. This capability significantly speeds up retrieval, especially in applications involving large datasets, such as image and text retrieval, where traditional databases may struggle to deliver timely results (Pan et al., 2024). The architecture of vector databases is optimized for approximate nearest neighbor (ANN) searches, which are essential for efficiently handling the high dimensionality of vector data.

One key feature distinguishing vector databases from traditional DBMS is their ability to perform approximate searches instead of exact matches. In VecDBs, the semantic similarity between vectors is represented by their distances. This enables searches that identify the top-k nearest neighbors in a high-dimensional space. This is critical for applications where understanding context and meaning is more important than exact data retrieval.

2.2.3 Generative Information Retrieval

Generative Information Retrieval (GIR) is an advanced approach that combines traditional retrieval methods with generative techniques to enhance the relevance and quality of search results. Unlike standard retrieval systems that primarily fetch documents based on matching terms, generative retrieval generates new content or modifies existing content to address user queries better. This method leverages large language models (LLMs) to produce more coherent, context-aware responses, and tailored to user needs (Lewis et al., 2020). By integrating generative capabilities, GIR can improve the overall effectiveness of information retrieval systems, making them more adaptable to diverse user requirements and complex queries. This approach marks a significant shift in how information is accessed and utilized, moving towards more dynamic and interactive search experiences.

Generative Retrieval A formal representation of the generative process can be defined as follows:

$$R(q,d) = P(d'|q;\theta) \tag{2.7}$$

Where:

- $R(q,d)$ is the relevance score of document d given the query q,
- d' is the document identifier generated in response to the query,
- θ represents the model parameters.

The emergence of generative retrieval represents a shift from traditional index-based retrieval methods. Generative retrieval relies on pre-trained generative models, such as Text-to-Text Transfer Transformer(T5) (Raffel et al., 2020) and BART (Lewis, 2019), to directly generate document identifiers (DocIDs) related to the query, achieving end-to-end retrieval without relying on large-scale pre-built document indices.

A key technique in generative information retrieval is RAG. RAG first uses either sparse or dense retrieval methods to find relevant information. It then utilizes this information to create accurate and context-aware responses. The RAG framework can be expressed as:

$$R(q,d) = \sum_{i=1}^{T} P(d'_i | d'_{<i}, q; \theta) \tag{2.8}$$

Where:

- T is the length of the generated document identifier d',
- d'_i represents the token at position i,
- $d'_{<i}$ is the sequence of tokens generated before position i.

This process ensures end-to-end optimization by directly generating document identifiers, reducing reliance on external indexing, and lowering the system's demand for storage resources.

Query Augmentation Query augmentation is a crucial technique that enhances the original user query. Formally, if we denote the original query as q and the augmented query as q', we can represent the augmentation process as:

$$q' = \text{Augment}(q, \mathscr{T}) \tag{2.9}$$

Where $\mathscr{T}$ is the set of relevant terms added to the original query. This enables the system to retrieve a more comprehensive set of documents, thereby enhancing the relevance and accuracy of the results (Khan et al., 2019).

Response Generation Response generation involves producing accurate answers based on the retrieved documents and the user's query. Formally, given a set of retrieved documents D and a query q, the task can be modeled as:

$$r = \text{Generate}(D, q; \theta) \tag{2.10}$$

Where r represents the generated response, D is the retrieved context, and θ denotes the parameters of the generative model.

Various generative models have been integrated into Retrieval-Augmented Generation (RAG) systems to produce accurate and coherent responses. Prominent models include GPT-4 (OpenAI, 2023), BART (Lewis, 2019), T5 (Raffel et al., 2020) and LLaMA (Touvron et al., 2023), Falcon (Penedo et al., 2023). Each model varies in terms of complexity, training approach, computational requirements, and inference efficiency.

2.3 Multimodal Learning and Knowledge Fusion Theory

Multimodal learning is a foundational area in AI focused on integrating diverse data types, such as text, images, audio, and video, to create enriched representations that enhance model performance (Baltrušaitis et al., 2018). This approach enables applications requiring unified data understanding, including image captioning, visual question answering, and cross-modal retrieval. As data from varied sources increases, knowledge fusion techniques approaches that align and unify multimodal information become crucial for developing systems that interpret and reason over complex, multimodal data (Gou et al., 2021).

Section 2.3.1 introduces Multimodal Machine Learning, detailing methods that enable models to process and integrate information from different modalities. Section 2.3.2, Representation Learning for Multimodal Data, explores approaches to encoding and unifying multimodal information, covering techniques such as joint and coordinated representations that enhance alignment across modalities. Section 2.3.3 discusses Attention Mechanisms, examining how attention techniques like self-attention and cross-attention enable models to selectively focus on relevant features across modalities, which is critical for tasks requiring cross-modal relationships.

Section 2.3.4 delves into Knowledge Fusion, which integrates models and data sources through methods like model fusion and knowledge distillation, creating cohesive representations that strengthen multimodal understanding. Finally, Section 2.3.5 examines Ontologies: Organizing and Representing Knowledge, highlighting how structured frameworks help organize and standardize information across modalities, thereby supporting interoperable and contextually aware AI systems.

2.3.1 Multimodal Machine Learning

Multimodal machine learning (MML) is the field of study focused on designing models that learn from data across multiple modalities, such as text, images, audio, and video, to capture more prosperous and more comprehensive data representations than possible from any single modality. This approach leverages the unique strengths of each modality, combining complementary information to enhance model performance across various tasks (Baltrušaitis et al., 2018). MML enables models to achieve deeper insights and improve their robustness, as demonstrated in applications ranging from emotion recognition to autonomous driving and medical diagnostics, where each modality contributes unique context, spatial, and semantic information critical to accurate interpretation (Tsai et al., 2018).

2.3.2 Representation learning for multimodal data

Representation learning for multimodal data focuses on developing methods that integrate diverse data sources text, images, audio, and video into meaningful, machine-interpretable forms. By capturing complementary information from different modalities, multimodal representation learning enables models to achieve more robust performance across complex tasks. Recent advances show that integrating diverse data sources can lead to more comprehensive representations, improving tasks such as sentiment analysis, visual question answering, and emotion recognition, where single-modality data may lack necessary context or depth (Baltrušaitis et al., 2018; Guo et al., 2022).

In multimodal learning, two primary approaches are commonly used for creating representations: **joint representation** and **coordinated representation**. Joint representation combines multiple modalities into a shared space, simplifying model architecture by allowing the model to operate in a unified vector space (Yao et al., 2020). In contrast, coordinated representation maintains separate embeddings for each modality. Still, it aligns them to ensure comparable or related meanings across modalities, which is beneficial when preserving modality-specific features is essential (Tsai et al., 2018).

Joint Representation

Joint representation fuses features from different modalities into one combined embedding, capturing interactions and dependencies. Given two modalities M_1 and M_2 with feature representations $X_1 \in \mathbb{R}^{d_1}$ and $X_2 \in \mathbb{R}^{d_2}$, where d_1 and d_2 are the dimensions of each modality, we define the joint representation Z as:

$$Z = f(X_1, X_2; \theta) \tag{2.11}$$

Where:

- f is a fusion function parameterized by θ, often a neural network that learns to combine X_1 and X_2 into a unified vector space $Z \in \mathbb{R}^{d_z}$.
- d_z is the dimensionality of the joint embedding space, which can vary depending on the fusion technique used.

Concatenation is a common fusion technique:

$$Z = [X_1; X_2] \tag{2.12}$$

Where $[;]$ denotes Concatenation of the feature vectors.

Attention-based fusion is also widely used, especially in transformer-based models, where each modality's features are weighted dynamically. Attention scores $\alpha_{i,j}$ between features X_i from modality M_1 and X_j from modality M_2 are calculated as:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_k \exp(e_{i,k})} \tag{2.13}$$

Where:

$$e_{i,j} = Q_i \cdot K_j \tag{2.14}$$

Here, Q and K are query and key vectors derived from the embeddings, enhancing inter-modal alignment through attention mechanisms (Kim et al., 2021).

Coordinated Representation

Coordinated representations keep the embeddings for each modality separate but align them to be semantically consistent. This approach is practical in applications like cross-modal retrieval, where preserving unique modality-specific details is crucial. For embeddings $Z_1 \in \mathbb{R}^{d_1}$ and $Z_2 \in \mathbb{R}^{d_2}$ for M_1 and M_2, a typical goal is to maximize the similarity between semantically related data points.

One method for alignment is **cosine similarity**, which is computed as follows:

$$S(Z_1, Z_2) = \frac{Z_1 \cdot Z_2}{\|Z_1\| \|Z_2\|} \tag{2.15}$$

Where $S(Z_1, Z_2)$ is maximized for pairs of embeddings that are semantically aligned.

Another widely-used approach is **contrastive loss**. In this method, positive pairs (related samples) are pulled closer together, while negative pairs (unrelated samples) are pushed apart. The contrastive loss $L_{contrast}$ is defined as:

$$L_{contrast} = \sum_{(i,j)\in P} d(Z_1^{(i)}, Z_2^{(j)})^2 + \sum_{(i,j)\in N} \max(0, m - d(Z_1^{(i)}, Z_2^{(j)}))^2 \tag{2.16}$$

Where:

- P and N represent the sets of positive and negative pairs, respectively,
- $d(Z_1^{(i)}, Z_2^{(j)})$ is the Euclidean distance between embeddings,
- m is a margin that defines the minimum distance for negative pairs, ensuring they are adequately separated (Chen et al., 2020; Radford et al., 2021)

2.3.3 Attention mechanisms

In multimodal learning, attention mechanisms play a pivotal role, allowing models to focus on the most relevant parts of each modality selectively. This selective weighting aligns different modalities and captures cross-modal dependencies, enabling more accurate and contextually informed representations (Tsai et al., 2018). Attention mechanisms are beneficial in scenarios where information from one modality needs to inform or complement another, such as aligning visual content in an image with textual descriptions in natural language (Chen et al., 2020; Tan and Bansal, 2019). For instance, vision-language models use cross-attention layers to dynamically relate parts of an image to relevant words in a sentence, a feature critical for tasks like image captioning and visual question answering (Radford et al., 2021).

Self-Attention Mechanism

The self-attention mechanism, popularized by transformer architecture, has become foundational in modern multimodal and LLMs due to its capacity to capture relationships within and between data sequences. In self-attention, each feature in an input sequence is transformed into **query (Q)**, **key (K)**, and **value (V)** matrices, enabling the model to assess the relevance of different features across modalities by computing attention scores (Alexey, 2020).

Given an input sequence of feature vectors $X = \{x_1, x_2, \ldots, x_n\}$ from a modality M, where each $x_i \in \mathbb{R}^d$, the attention score $\alpha_{i,j}$ between features x_i and x_j is calculated as follows:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_k \exp(e_{i,k})} \tag{2.17}$$

Where:

$$e_{i,j} = \frac{Q_i \cdot K_j}{\sqrt{d}} \tag{2.18}$$

Here:

- $Q = XW_Q$, $K = XW_K$, and $V = XW_V$ are the query, key, and value matrices, with W_Q, W_K, and W_V being learned weight matrices.
- d is the dimensionality of the query and key vectors, and the scaling factor $\sqrt{d}$ stabilizes the attention scores Alexey (2020).

The output for each feature vector is then computed as a weighted sum of the values:

$$\text{Attention}(Q, K, V) = \sum_j \alpha_{i,j} \tag{2.19}$$

This self-attention mechanism allows each feature from one modality to focus on important features across the entire input, supporting effective cross-modal alignment (Jaegle et al., 2021; Kim et al., 2021).

Cross-Attention in Transformers for Multimodal Alignment

In multimodal transformers, *cross-attention layers* align features across different modalities by using the query from one modality and the key-value pairs from another modality. This process allows the model to focus on specific regions or parts of one modality (e.g., visual features in images) in response to another modality (e.g., text descriptions). Models

like **ViLBERT** and **LXMERT** employ cross-attention layers to align image features with corresponding textual tokens, ensuring that each modality's context is effectively integrated into a joint representation for downstream tasks (Lu et al., 2019; Tan and Bansal, 2019).

For cross-attention, the attention score $\alpha_{i,j}^{(m,n)}$ between a feature $x_i^{(m)}$ from modality M and a feature $x_j^{(n)}$ from modality N is calculated similarly:

$$\alpha_{i,j}^{(m,n)} = \frac{\exp\left(\frac{Q_i^{(m)} \cdot K_j^{(n)}}{\sqrt{d}}\right)}{\sum_k \exp\left(\frac{Q_i^{(m)} \cdot K_k^{(n)}}{\sqrt{d}}\right)} \tag{2.20}$$

Where:

- $Q^{(m)}$ and $K^{(n)}$ are query and key matrices derived from modalities M and N, respectively.

This cross-modal attention mechanism enables transformers to attend across different modalities in a structured manner, learning robust and aligned representations that are particularly useful for multimodal applications such as image captioning, video understanding, and visual question answering (Chen et al., 2020; Radford et al., 2021)

2.3.4 Knowledge Fusion

Knowledge fusion in multimodal learning refers to combining and aligning knowledge from diverse data sources to create a unified, contextually enriched representation. This fusion process integrates information from different modalities, models, and domains, resulting in a comprehensive and cohesive knowledge structure (Poole and Mackworth, 2023; Zhang et al., 2021). Below are critical methods in knowledge fusion that contribute to the alignment and organization of multimodal information, each providing unique advantages for effective integration and representation across diverse sources.

2.3.5 Ontologies Organizing and Representing Knowledge

Ontologies are structured frameworks that represent and organize knowledge across various domains by defining relationships between entities, concepts, and their attributes. In multimodal learning, ontologies play a pivotal role in providing a standardized schema, which aligns data from disparate sources such as text, images, and audio, fostering semantic interoperability across modalities (Taglino et al., 2023). Ontologies define categories and relationships, facilitating interoperability by ensuring that related entities in different modalities correspond to the same concept. For instance, in medical imaging, ontologies can align textual descriptions of diseases with visual features in diagnostic images, enabling accurate cross-modal retrieval and reasoning (Dunnmon et al., 2020).

Mathematically, an ontology O can be represented as a set of concepts $C = \{c_1, c_2, \ldots, c_n\}$, relationships $R = \{r_1, r_2, \ldots, r_m\}$, and axioms that define the hierarchical structure among concepts. For two entities c_i and c_j, a relationship $r \in R$ signifies that c_i and c_j share a specific connection, facilitating cross-modal alignment through shared ontological structures (Luschi et al., 2023). This structured approach enables models to reason across different types of information and enriches multimodal datasets with contextual depth and accuracy.

- **Concepts (C)**: Represented as $C = \{c_1, c_2, \ldots, c_n\}$, where each c_i is a distinct concept or entity within the ontology.
- **Relationships (R)**: Represented as $R = \{r_1, r_2, \ldots, r_m\}$, where each r_j is a relationship type, such as "is-a" or "part-of," which defines the connection between two or more concepts.
- **Axioms**: Logical statements that define hierarchical structures or constraints among concepts and relationships within the ontology.

For example, given two concepts c_i and c_j, a relationship $r \in R$ might signify a hierarchical or associative connection, enabling the system to interpret c_i as related to c_j in a manner that facilitates cross-modal alignment.

$$O = \{C, R, \text{Axioms}\} \tag{2.21}$$

$$C = \{c_1, c_2, \ldots, c_n\} \tag{2.22}$$

$$R = \{r_1, r_2, \ldots, r_m\} \tag{2.23}$$

This formalism enables robust knowledge representation and alignment across different modalities, ensuring that shared ontological structures enhance multimodal data integration and reasoning.

Model Fusion

Model fusion is a technique used in multimodal learning to integrate multiple models that process distinct data modalities, resulting in a unified representation that leverages the strengths of each model. By combining insights from different models, model fusion can enhance performance, particularly in tasks where complementary information from separate modalities (e.g., text, image, and audio) is essential (Gao et al., 2020). Model fusion approaches vary in complexity, from simple ensemble methods to more sophisticated attention-based fusion layers that dynamically weight model outputs based on contextual relevance (CHEN et al., 2020).

A common approach in model fusion is the *weighted averaging* of model outputs. Given two models, f_1 and f_2, which produce predictions $f_1(x)$ and $f_2(y)$ for two modalities x and y, respectively, the fused output z can be computed as:

$$z = \alpha f_1(x) + (1 - \alpha) f_2(y) \tag{2.24}$$

Where:

- α is a weighting factor, $0 \leq \alpha \leq one$, that adjusts the contribution of each model's output based on their relative importance (Chen et al., 2020).

Another approach to model fusion, particularly in deep learning, involves *attention-based fusion layers*. In this approach, the attention mechanism dynamically assigns weights to the outputs of each model depending on their relevance to the task. For instance, if the attention score β represents the importance of modality M relative to another modality N, the fused representation z can be formulated as:

$$z = \beta f_1(x) + (1-\beta) f_2 \tag{2.25}$$

Where β is determined by an attention function that takes into account the context of the input data, such as:

$$\beta = \text{softmax}(W[f_1(x); f_2(y)]) \tag{2.26}$$

where W is a learned weight matrix, and $[f_1(x); f_2(y)]$ denotes the concatenation of the model outputs (Tsai et al., 2018).

In ensemble-based model fusion, individual model predictions are often aggregated through *voting* or *averaging*, making it particularly useful in applications like image captioning, sentiment analysis, and multimodal sentiment recognition (Gandhi et al., 2023). The fusion of predictions increases robustness by reducing the likelihood of errors in individual models.

Overall, model fusion provides a powerful framework for enhancing multimodal understanding. It allows for integrating diverse sources of information and optimizing the relevance of each modality to the task at hand.

Knowledge Distillation and Knowledge Fusion in LLMs

Knowledge Distillation

Knowledge distillation is a model compression technique where a more extensive, complex model (the "teacher") transfers its knowledge to a smaller, simpler model (the "student"). This approach is beneficial in multimodal learning, where distilling multimodal expertise from a large model into a compact one maintains model performance while reducing computational demands (Gou et al., 2021). The distillation process minimizes the difference between the teacher model's output probabilities P_T and the student model's output probabilities P_S, typically using *Kullback-Leibler (KL) divergence*, which measures similarity between the probability distributions:

$$L_{\text{KD}} = \text{KL}(P_T || P_S) = \sum_i P_T(i) \log \frac{P_T(i)}{P_S(i)} \tag{2.27}$$

Where:

- P_T and P_S are the probability distributions of the teacher and student models, respectively.
- i represents each class or output unit in the model's final layer.

This loss function allows the student model to learn to mimic the output behavior of the teacher, thereby capturing essential patterns and relationships that the teacher model encodes (Gou et al., 2021).

Knowledge Fusion in Large Language Models (LLMs)

In multimodal contexts, *Large Language Models (LLMs)* such as CLIP and GPT-4 are adapted to fuse knowledge across different data types (e.g., text, image, and audio) to create cohesive and versatile models. LLMs achieve knowledge fusion through specialized architectures and cross-modal training objectives that align embeddings across modalities, creating a shared multimodal space for integrated understanding (Radford et al., 2021).

For instance, in models like CLIP, a joint training objective aligns image and text embeddings within a common space, enabling cross-modal retrieval where images can retrieve corresponding text and vice versa. The *fusion objective* for aligning modality-specific embeddings z_t (text) and z_i (image) is given by:

$$L_{\text{fusion}} = \sum_{(z_t, z_i)} \|z_t - z_i\|^2 \tag{2.28}$$

Where:

- z_t and z_i are the embeddings of corresponding text and image data points.
- The objective minimizes the Euclidean distance between matched pairs, encouraging the embeddings to occupy a shared semantic space.

Furthermore, *multimodal adapters* are sometimes added to LLMs, allowing them to process new modalities by adjusting existing weights rather than retraining the model from scratch (Ebrahimi et al., 2024). This enables LLMs to incorporate additional modalities into their knowledge structure efficiently, enhancing their utility in multimodal learning tasks.

2.4 Theoretical Framework

The theoretical framework for this research integrates key concepts from information retrieval and multimodal learning to support the development of effective technical support systems. The framework is structured around core theories, starting with sparse and dense retrieval techniques that focus on enhancing response accuracy through rule-based, semantic, and vector-based approaches. Additionally, it explores generative information retrieval methods, which leverage techniques like query augmentation and response generation to improve the relevance of automated support responses. Further, the framework incorporates multimodal learning and knowledge fusion, employing attention mechanisms and knowledge distillation to enhance data representation and alignment across modalities. This comprehensive

integration aims to optimize the retrieval and generation processes, ensuring efficient and accurate information delivery, especially in complex, knowledge-intensive domains.

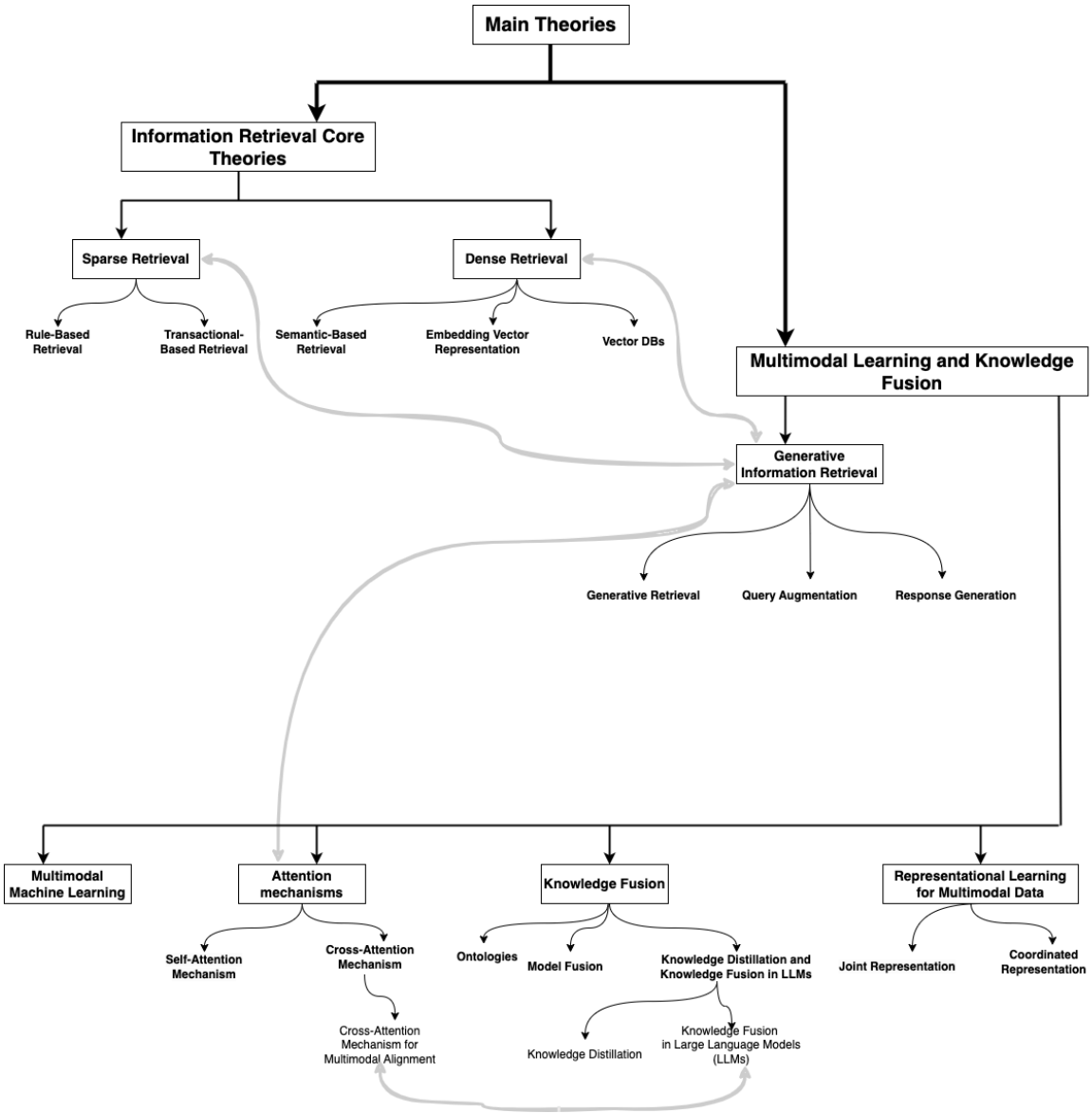


Figure 2.1: Theoretical Framework

2.5 Empirical Review

The empirical framework explores advancements in technical support systems by leveraging information retrieval techniques and Retrieval-Augmented Generation (RAG) models. Section

2.5.1 delves into the application of sparse and dense retrieval techniques, demonstrating how hybrid models enhance response efficiency and accuracy by combining keyword-based and semantic retrieval methods (Qu et al., 2024) (Paulin et al., 2024). Section 2.5.2 focuses on integrating RAG models to generate context-rich responses, reducing hallucinations, and improving user satisfaction through real-time data retrieval (rin Lee and Kim, 2024). Finally, section 2.5.3 examines evaluation metrics for information retrieval and RAG, emphasizing traditional metrics like Mean Reciprocal Rank (MRR) and innovative frameworks such as RAGElo and DIRAS to optimize response accuracy and relevance (Ni et al., 2024).

2.5.1 Application of Sparse and Dense Retrieval Techniques in Technical Support

Recent research has demonstrated significant advancements in the application of both sparse and dense retrieval techniques to optimize technical support systems. Sparse retrieval methods like BM25 have traditionally been used for handling routine inquiries due to their efficiency in retrieving answers based on keyword matching. These methods are particularly beneficial for technical support systems focused on resolving repetitive questions and standard troubleshooting scenarios (Zhou et al., 2024). In a comparative study, using binary token indexes in sparse retrieval reduced system preparation time and storage requirements while still achieving competitive accuracy, demonstrating how sparse techniques can enhance the cost-efficiency of support systems (Zhou et al., 2024).

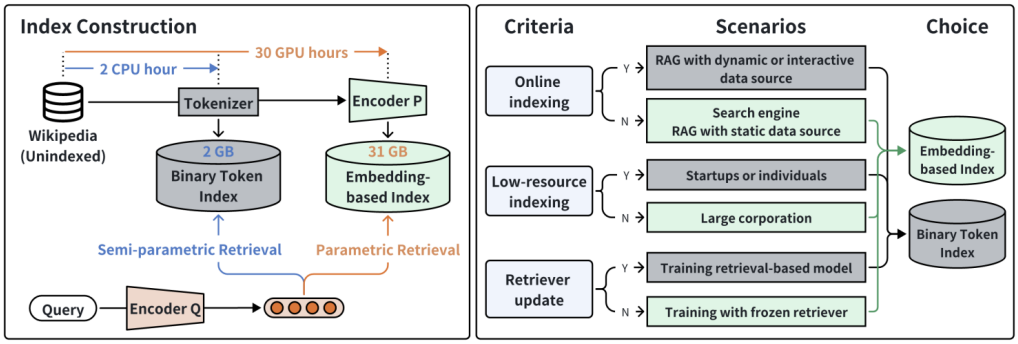


Figure 2.2: Left: a comparison of storage requirements (2 GB vs. 31 GB) and time/resource costs (2 CPU hours vs. 30 GPU hours) for setting up a semi-parametric retrieval pipeline with a binary token index versus a conventional neural retrieval pipeline with an embedding-based index. Right: decision-making criteria and scenarios guiding the selection between the two indexing types.

However, dense retrieval methods, particularly those based on deep learning models like BERT, have gained traction for handling more complex, context-sensitive queries. These models use dense embeddings to capture semantic nuances, enabling more accurate retrieval of support documents even when queries involve synonyms or specialized terminology (Qu et al., 2024). In practical applications, integrating dense retrieval has improved the relevance of results, particularly in domains where a precise understanding of the query context is essential.

Interestingly, hybrid models that combine both sparse and dense retrieval techniques are emerging as a promising solution to maximize the strengths of each method. For example, systems that utilize a sparse index for initial filtering, followed by dense retrieval for fine-tuned ranking, have shown significant improvements in retrieval accuracy and efficiency (Paulin et al., 2024). This layered approach ensures that support systems can efficiently handle both routine and complex inquiries, thereby enhancing user satisfaction and reducing response times.

The ongoing evolution of retrieval methods in technical support highlights the need for continuous refinement, particularly as customer expectations for quick and accurate resolutions increase. By leveraging advancements in sparse and dense retrieval, organizations can build robust support systems that effectively balance speed and accuracy in customer interactions.

2.5.2 Comparative Analysis of Generative Models

When evaluating generative models for Retrieval Augmented Generation (RAG) systems, it is essential to consider output quality, inference cost, and processing speed. Tables 2.1, 2.2, and 2.3, together with Figures 2.3, 2.4, 2.5, and 2.6, present a comparative analysis of several leading models based on Azure AI leaderboard benchmarks (May 2025).

Table 2.1 shows that the o3 and o4 mini models lead in quality.Table 2.2 highlights Ministral 3B as the most cost effective model, while Figure 2.4 visually confirms the relationship between quality and cost. In terms of throughput, Table 2.3 and Figure 2.5 demonstrate that Llama 3.2-1B Instruct and o3 mini achieve the highest processing speeds.

GPT-4 stands out for its strong reasoning capabilities, reliable understanding of technical prompts, and overall stability—qualities that are essential in enterprise support settings. As shown in Figure 2.6, GPT-4 maintains a high quality index despite its higher cost. Based on these factors, GPT-4 was selected for this study to support the RAG system's ability to deliver consistently accurate and well-informed responses in complex support scenarios.

Table 2.1: Model Comparison by Quality Index (↑)

Model	**Quality Index**
o3	**0.90**
o4-mini	0.89
DeepSeek-R1	0.87
o1	0.87
MAI-DS-R1	0.87

Table 2.2: Model Comparison by Cost (↓, USD per 1M Tokens)

Model	Cost (USD)
Mistral-3B	**0.04**
Phi-4-mini-instruct	0.13
Mistral-Nemo	0.15
Mistral-small-2503	0.15
gpt-4.1-nano	0.17

Table 2.3: Model Comparison by Throughput (↑, Tokens per Second)

Model	Throughput (tokens/sec)
Llama-3.2-1B-Instruct	**131.16**
o3-mini	127.75
gpt-4.1-mini	125.04
gpt-4.1-nano	115.82
o1-mini	105.88

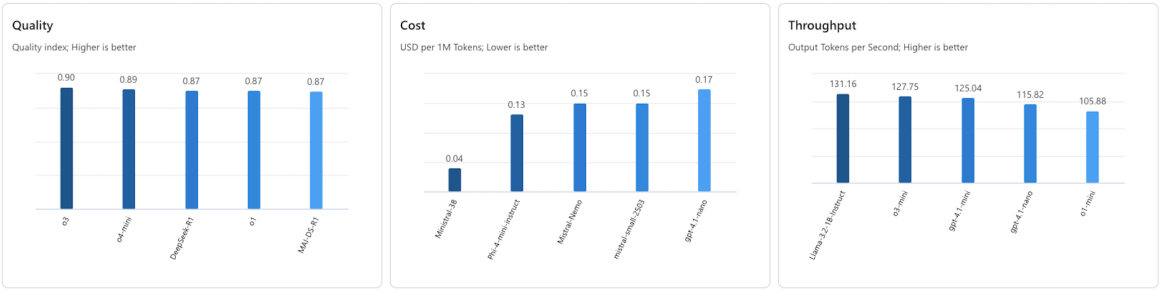


Figure 2.3: Overall Benchmark Comparison by Quality, Cost, and Throughput (Azure AI Leaderboard, May 2025)

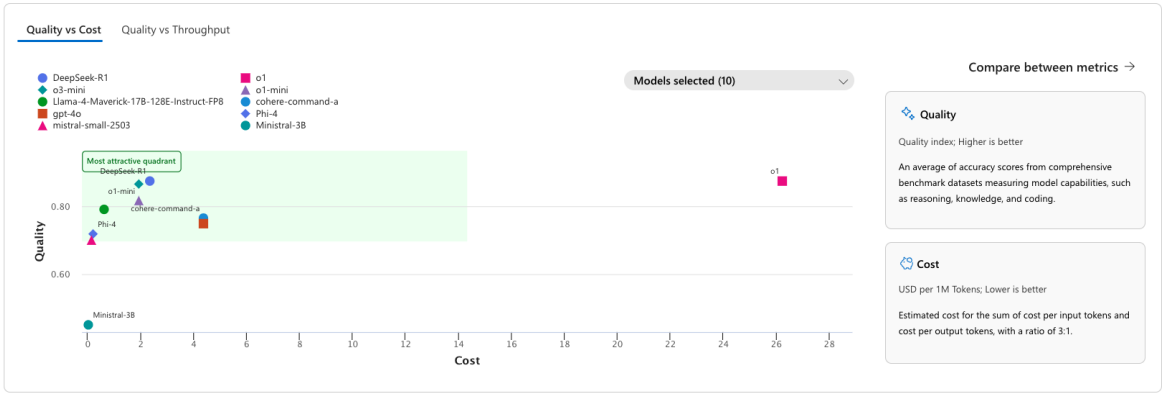


Figure 2.4: Quality vs Cost Comparison of Selected Generative Models

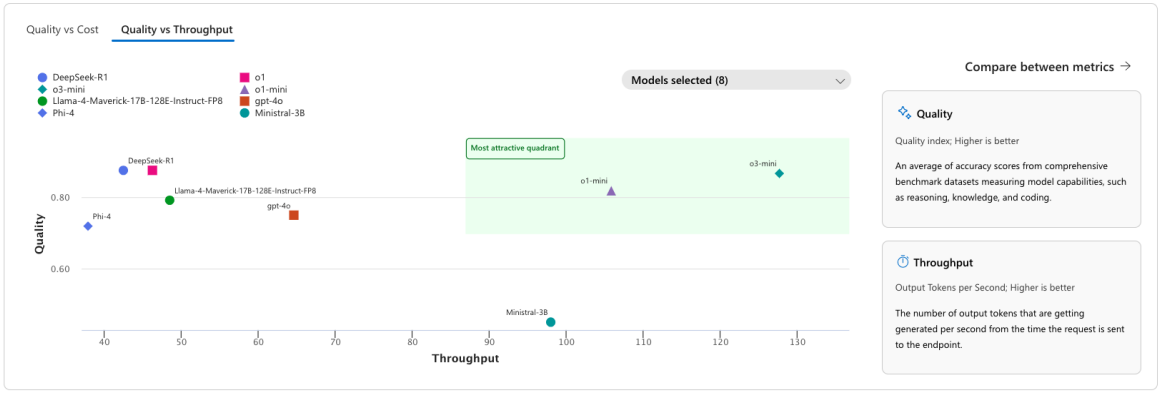


Figure 2.5: Quality vs Throughput Comparison of Selected Generative Models

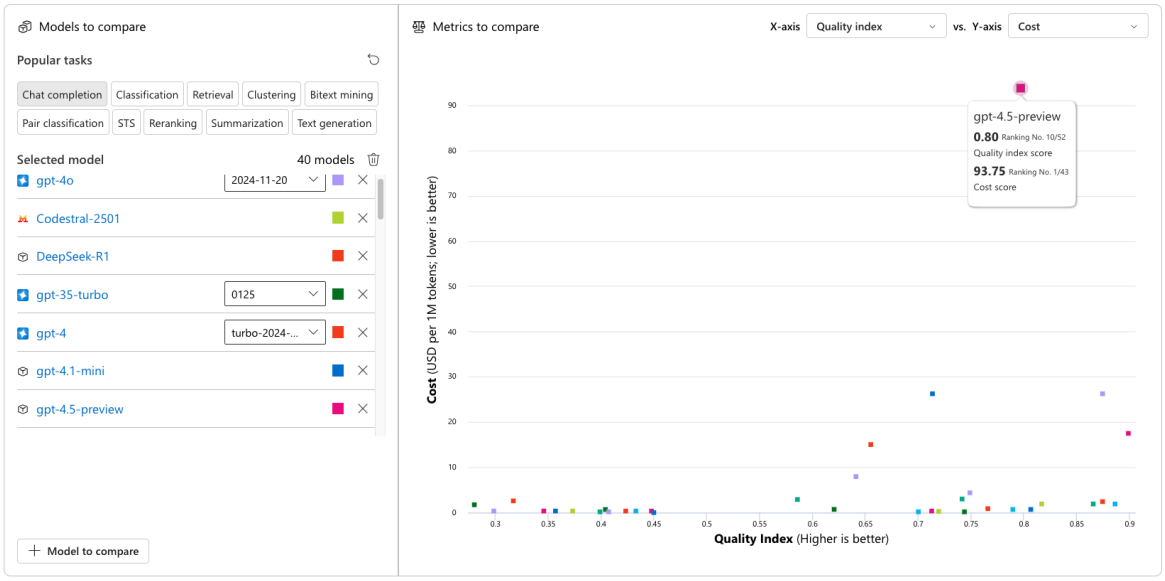


Figure 2.6: GPT-4 Model Benchmarking: Quality vs Cost in Comparison to Other Leading Models

2.5.3 Integrating RAG for Enhanced Technical Support

Recent empirical research emphasizes the growing impact of Retrieval-Augmented Generation (RAG) models in technical support systems, particularly in improving response accuracy and user satisfaction. RAG models combine generative capabilities with real-time retrieval from a knowledge source, enabling the production of context-rich, factually grounded responses.

The Case Law Evaluation and Retrieval Corpus (CLERC), designed for legal case analysis, demonstrated the effectiveness of RAG models in generating accurate and evidence-supported responses in complex domains. While models like GPT-4 perform well in fluency, RAG-enhanced systems surpassed them in factual accuracy by grounding outputs in retrieved documents (Hou et al., 2024). This highlights RAG's advantage in reducing hallucinations and improving trustworthiness.

Further advancements include Model Internals-based RAG Explanations (MIRAGE), which improves answer attribution by linking generated outputs to model internals and retrieved sources. This approach has shown strong alignment with human assessments, particularly

in multilingual settings, reinforcing the reliability of RAG systems in technical support (Qi et al., 2024).

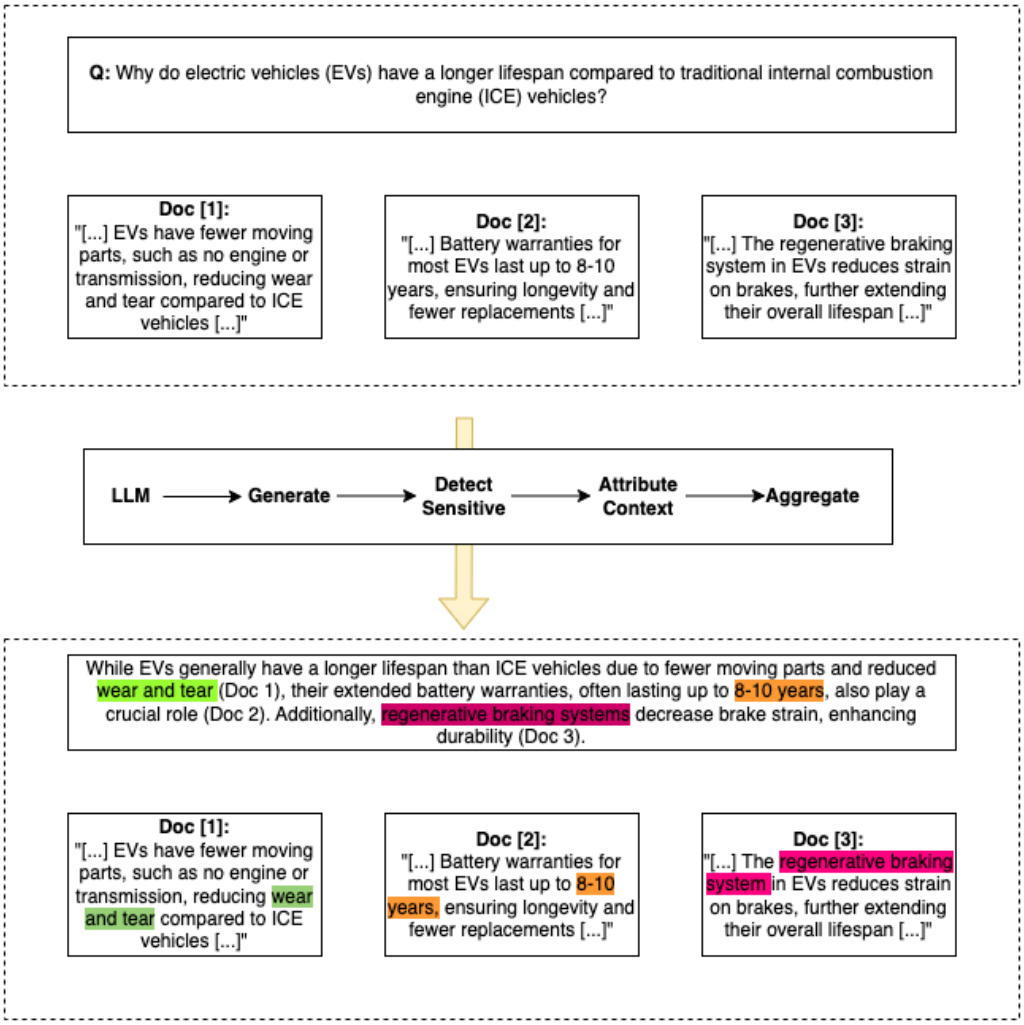


Figure 2.7: MIRAGE: A model internals-based attribution framework for RAG. Colored spans denote context-sensitive matches traced to supporting sources.

In addition, the Satyrn platform integrates RAG with structured data analytics to generate accurate, coherent reports from both structured and unstructured inputs. This demonstrates the flexibility of RAG in enterprise support applications where data sources vary in format (Sterbentz et al., 2024).

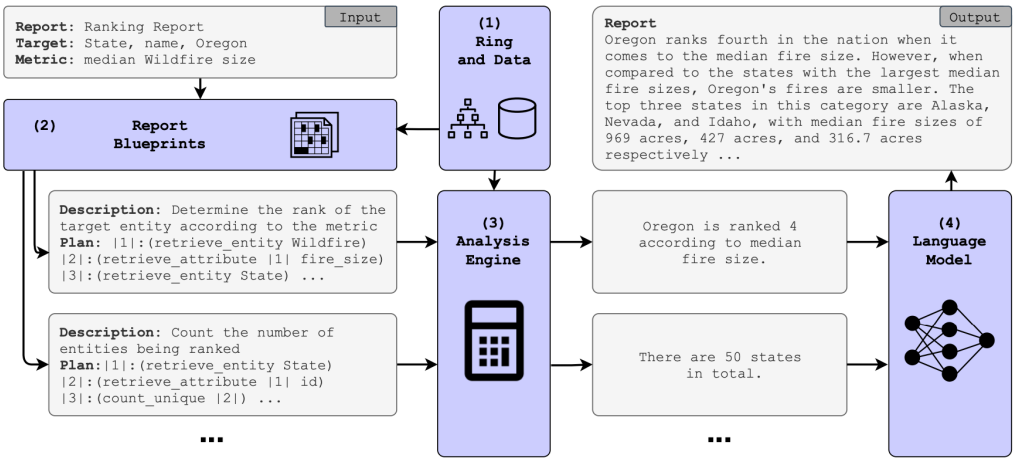


Figure 2.8: High-level overview of SATYRN's analytics-augmented generation framework.

These studies underscore the transformative potential of RAG in automating technical support by producing responses that are accurate, explainable, and aligned with user intent. As RAG systems continue to advance, they are poised to become a foundational technology in next-generation customer service platforms.

2.5.4 Evaluation of Retrieval Augmented Generation

Evaluating the effectiveness of a Retrieval Augmented Generation (RAG) system requires metrics that assess both the quality and relevance of generated responses. This study focuses on three widely adopted evaluation metrics: F1 Score, BLEU, ROUGE-L, and METEOR.

F1 Score: F1 Score measures the balance between precision and recall by evaluating the number of shared words between the generated response and the reference text. In the context of natural language generation, it provides a useful approximation of how accurately and completely the system captures the essential content of the ground truth. It is defined as the harmonic mean of precision and recall:

$$\text{F1 Score} = \frac{2 \cdot \mathit{Precision} \cdot \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}} \tag{2.29}$$

This metric is particularly valuable for open-ended queries where partial lexical overlap is meaningful, even if the phrasing differs from the reference response.

BLEU (Bilingual Evaluation Understudy): BLEU measures the n-gram precision of the generated response against one or more reference texts, with a brevity penalty to penalize overly short outputs. It is computed as:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{2.30}$$

where p_n is the modified precision for n-grams, w_n is the weight assigned to each n-gram level, and BP is the brevity penalty.

ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation): ROUGE-L evaluates the overlap of the longest common subsequence (LCS) between the generated and reference responses. It emphasizes content recall and is given by:

$$\text{ROUGE-L} = \frac{(1+\beta^2) \cdot R \cdot P}{R + \beta^2 \cdot P} \tag{2.31}$$

where R is recall, P is precision, and β controls the relative importance of recall versus precision (typically set to 1.2).

METEOR (Metric for Evaluation of Translation with Explicit Ordering): METEOR improves on BLEU and ROUGE by incorporating synonym and stem matches, as well as word order. It is defined as:

$$\text{METEOR} = F_{mean} \cdot (1 - Penalty) \tag{2.32}$$

where F_{mean} is the harmonic mean of unigram precision and recall, and $Penalty$ penalizes fragmented matches.

These metrics collectively provide a robust framework for evaluating the accuracy, fluency, and content alignment of RAG-generated responses. This study applies BLEU, ROUGE-L, and METEOR to assess the quality of the RAG outputs.

2.6 Research Gap

Enterprise technical support systems often face persistent challenges in delivering fast, accurate, and context-aware responses to user queries. These environments are characterized by complex infrastructure, evolving documentation, and a high volume of repetitive, ambiguous, or domain-specific issues. Traditional rule-based systems, although structured and explainable, are rigid and require constant manual updates. On the other hand, purely generative models may produce fluent but ungrounded responses, lacking alignment with internal procedures, technical constraints, or documentation standards. This disconnect affects the reliability, traceability, and trustworthiness of AI-powered support tools.

Recent advances in Retrieval Augmented Generation (RAG) offer a promising alternative by combining document retrieval with generative language models, thereby enhancing response relevance and grounding. However, the adoption of RAG systems in enterprise technical support remains limited. Most existing RAG implementations target open-domain question answering or customer-facing conversational agents that rely on publicly available data. These models are not designed to operate within the constraints of enterprise environments that require integration with internal documentation, access control, domain-specific vocabularies, and compliance considerations.

In particular, there is a lack of RAG systems designed to work with proprietary knowledge sources such as system configuration guides, troubleshooting playbooks, historical support tickets, and integration manuals. Few existing systems address the challenges of retrieving from internal content at scale, handling long-form queries with vague or incomplete descriptions, and generating responses that are both contextually accurate and explainable to support staff and end users.

This research addresses the gap by designing and implementing a Retrieval Augmented Generation system specifically tailored for enterprise technical support workflows. The proposed solution integrates semantic retrieval using vector embeddings with generative response synthesis, grounded in internal documentation indexed via Azure AI Search. The system is designed to improve support quality by surfacing relevant knowledge, reducing dependency on manual escalation, and delivering consistent, traceable responses aligned with organizational practices.

2.7 Conceptual Framework

This study adopts a Retrieval-Augmented Generation (RAG) framework using Azure AI Foundry to improve enterprise technical support. The system integrates internal document retrieval with a Large Language Model (LLM) to generate accurate, context-aware responses.

Section 2.7.1 describes the architecture: a retriever searches indexed data, an orchestrator manages the flow, and an LLM that generates responses using the query and retrieved content.

Section 2.7.2 outlines the workflow: a user submits a prompt, relevant documents are retrieved, and the LLM generates a grounded response based on both inputs.

2.7.1 RAG Workflow

Document and Query Encoding

Enterprise documents were indexed using Azure AI Search, which converted them into vector representations for semantic retrieval. User queries were similarly encoded at runtime.

Similarity Search

Azure AI Search compared the query vector to indexed document vectors and retrieved the most relevant matches based on semantic similarity.

Context Retrieval

The top matching document segments were combined with the user query to form an enriched input. This provided necessary context from internal resources.

Response Generation

The enriched input was sent to a GPT-4 model hosted on Azure AI Foundry. The model generated a complete response using both the query and retrieved context.

Final Output

The system returned a response that was accurate, relevant, and grounded in enterprise knowledge. This workflow improved clarity, precision, and response consistency in technical support.

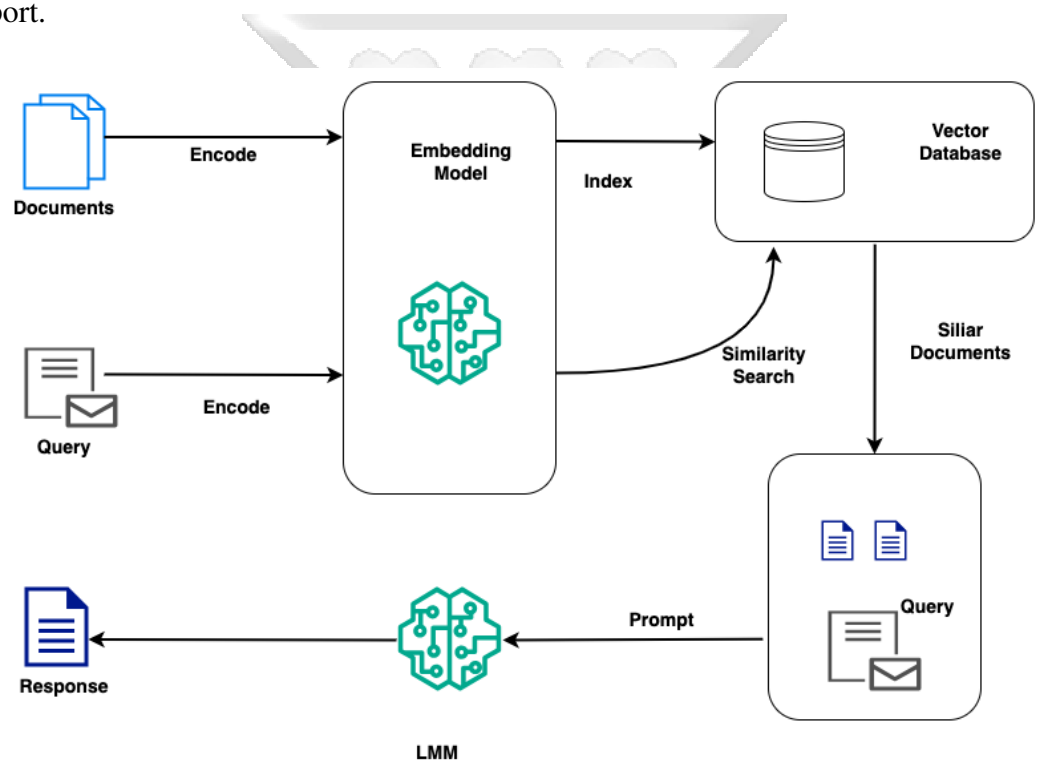


Figure 2.9: RAG Workflow implemented using Azure AI Studio

2.7.2 Technology Stack

Data Processing and Management

Internal support documentation was ingested and prepared using Azure AI Studio tools. The content was segmented into manageable sections to support context aware retrieval during query processing.

Storage and Indexing

Azure AI Search was used to store and index the processed content. The system converted the input into vector representations and enabled semantic search across the enterprise knowledge base.

Embedding Generation and Retrieval

Azure AI Search handled the creation of embeddings and performed semantic matching. When a query was received, it was encoded and compared with document vectors to identify the most relevant content.

Language Model and Response Generation

GPT-4, hosted through Azure OpenAI, was used to generate responses. The retrieved content was combined with the original query and submitted to the model to produce accurate and context rich answers.

Deployment and Infrastructure

The solution was developed and deployed entirely within Azure AI Studio. The retrieval and generation stages of the pipeline were fully managed within the Azure platform, ensuring seamless integration, high availability, and enterprise-grade performance.

Chapter 3

Methodology

3.1 Introduction

This chapter described the methodology used to design, develop, and evaluate the Retrieval Augmented Generation (RAG) system. The methodology followed the Conceptual Framework presented in Section 2.7 and the CRISP DM framework. It included business understanding, data preparation, modeling, evaluation, and deployment. Each phase provided clear justifications for the methods chosen and explained their alignment with the system implementation detailed in Chapter 5.

3.2 Research Design

The study adopted an applied, explanatory, and deductive research design to develop and evaluate a Retrieval Augmented Generation (RAG) system for enterprise technical support. The design focused on addressing inefficiencies in accessing internal support documentation by integrating semantic search with generative response modeling. The applied component involved building a working RAG system using Azure AI Studio. The explanatory component focused on evaluating how embedding generation, document retrieval, and language model generation influenced response accuracy and relevance. The deductive approach was based on theories from natural language processing and generative AI, guiding both the system architecture and evaluation framework.

3.3 CRISP DM Framework

3.3.1 Business Understanding

The study was guided by the goal of improving the accuracy and efficiency of enterprise technical support. It focused on developing a Retrieval Augmented Generation (RAG) system capable of delivering timely and relevant responses by combining semantic retrieval with generative language modeling. The specific objectives were:

i) Analyze existing customer support methods to identify limitations in response accuracy, relevance, and efficiency.

ii) Develop a RAG system that retrieves and generates accurate and context specific responses using internal documentation.

iii) Evaluate the system based on response quality, relevance, and user experience in a simulated technical support environment.

3.3.2 Data Understanding

The dataset used in this study was sourced from internal enterprise documentation specifically curated to support technical operations. It included a diverse collection of structured and semi-structured content such as API integration guides, troubleshooting manuals, error resolution workflows, and frequently asked questions (FAQs). These documents were authored by internal engineering and support teams and reflected the organization's real-world knowledge base used by agents during customer support interactions.

The core focus was on the internal guide documentation, which captured repetitive, high-volume queries related to authentication issues, API connectivity, transaction failures, and configuration steps. The inclusion of FAQs provided valuable insights into recurring support topics, while the troubleshooting content offered procedural knowledge critical for issue resolution. This combination ensured that the dataset was both domain-specific and operationally relevant, simulating realistic support scenarios encountered by enterprise teams.

3.3.3 Data Preparation

To enable accurate and efficient semantic retrieval, the internal documentation was prepared through a structured process. The key steps were as follows:

i) **Document Chunking:** The cleaned documents were segmented into smaller, logically coherent sections based on structural cues such as headings, subheadings, and paragraph breaks. This chunking process ensured that each unit contained a focused piece of information, improving retrieval precision by allowing the system to match user queries with specific content blocks rather than entire documents.

ii) **Indexing in Azure AI Search:** The resulting chunks were indexed using Azure AI Search, which generated semantic vector representations for each chunk using an embedded language model. These embeddings enabled vector-based similarity search, allowing the system to retrieve the most relevant content based on the semantic meaning of the user's query rather than keyword overlap.

3.4 Modeling

The modeling phase operationalized the Retrieval Augmented Generation (RAG) workflow by integrating semantic retrieval with generative response synthesis. This pipeline was built using services within the Azure ecosystem, including Azure AI Search and Azure OpenAI, to ensure secure, scalable, and enterprise-compliant deployment. The modeling process consisted of three main components: embedding generation, semantic retrieval, and response generation.

3.4.1 Semantic Embedding Generation

During the modeling phase, each document chunk prepared in the previous stage was transformed into a high-dimensional semantic vector using the `text-embedding-ada-002`

model, provided by Azure OpenAI. This model is based on transformer architecture and produces dense vector representations that capture the meaning of a given text.

The embedding process was executed via Azure OpenAI's REST API endpoint, ensuring seamless integration into the system's pipeline. Each call to the endpoint returned a vector embedding of the input text, which was then stored along with associated metadata. These embeddings were indexed into Azure AI Search, enabling downstream semantic similarity comparisons between queries and document content.

3.4.2 Semantic Retrieval

At runtime, when a user submits a query through the interface, the query is first processed using the same `text-embedding-ada-002` model to generate a corresponding semantic embedding. This query vector is then passed to Azure AI Search, which performs a similarity search against the pre-indexed document embeddings.

Azure AI Search uses cosine similarity to rank the relevance of each document chunk with respect to the user query. The top-ranked chunks are retrieved as the most semantically aligned content. This retrieval mechanism ensures that the system selects not just keyword-matching responses, but contextually appropriate content grounded in the enterprise's internal documentation.

3.4.3 Retrieval Augmented Generation

The final stage involves combining the retrieved document chunks with the original user query and passing them to the generative language model for response synthesis. The system uses GPT-4, hosted on Azure OpenAI, to generate a contextually grounded response based on both the input query and the retrieved supporting content.

The query and context are formatted into a structured prompt and sent to Azure OpenAI's GPT-4 endpoint via an authenticated API call. The model processes this input and returns a fluent, context-aware response that integrates the retrieved information. This method ensures

factual consistency, reduces hallucinations, and aligns the response with the organization's support standards and knowledge base.

3.4.4 User Interface and Interaction

A web based interface allowed users to input queries and view retrieved content alongside generated responses. This interface supported usability testing and user interaction.

3.4.5 Evaluation Strategy

The system was evaluated using a dual approach that combined automated performance metrics with qualitative expert assessment to provide a comprehensive understanding of its effectiveness in a technical support setting.

Automated evaluation focused on four key natural language generation (NLG) metrics:

- **F1 Score:** Assessed the balance between precision and recall by comparing word-level overlap between generated responses and expert-curated reference answers. This metric was particularly useful in determining the system's ability to retain essential content while minimizing irrelevant output.
- **BLEU (Bilingual Evaluation Understudy):** Measured n-gram precision, capturing the degree of phrase-level similarity between the generated response and reference output. While BLEU is known to penalize paraphrasing, it provided useful insight into syntactic overlap.
- **ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation):** Focused on the longest common subsequence between generated and reference text, offering a recall-oriented perspective on content overlap. ROUGE-L was especially relevant for evaluating responses to multi-part or multi-step queries.
- **METEOR (Metric for Evaluation of Translation with Explicit Ordering):** Evaluated semantic similarity by accounting for synonyms, stemming, and word order. It offered

a more linguistically informed assessment and aligned more closely with human judgment than BLEU or ROUGE alone.

In addition to quantitative evaluation, a manual review was conducted using Azure AI Studio's built-in evaluation dashboard. Domain experts independently reviewed a stratified sample of system outputs. Each response was scored based on three dimensions:

- **Clarity:** The linguistic quality and grammatical correctness of the response.
- **Fluency:** Naturalness and coherence of expression.
- **Contextual Relevance:** The degree to which the response directly addressed the user's query using appropriate technical knowledge.

The combined results from automated scoring and human review provided both quantitative performance benchmarks and qualitative insights into real-world applicability, guiding further refinement of the system.

3.5 Deployment

The entire system was developed, tested, and deployed within the Azure AI Studio environment. Azure AI Studio served as a unified platform for orchestrating model access, storage, and integration workflows, allowing seamless connection between embedding generation, semantic retrieval, and generative modeling.

Key components of the deployment infrastructure included:

- **Azure OpenAI:** Hosted the embedding model (`text-embedding-ada-002`) and GPT-4 for generation. These services were accessed via authenticated RESTful APIs.
- **Azure AI Search:** Managed indexing of vector embeddings and provided real-time similarity-based retrieval capabilities using semantic ranking.
- **Azure Blob Storage :** Served as a storage layer for raw and preprocessed documents during data preparation.

- **Azure AI Studio interface:** Provided an interactive, browser-based environment for testing, prompting, and visualizing results.

The use of Azure-native components ensured enterprise-grade scalability, security, and availability. The deployment environment also supported version control, prompt testing, and automated evaluation workflows.

3.6 System Integration

The system was designed with a modular architecture, making it easily adaptable for integration into existing enterprise platforms such as internal support dashboards, developer portals, or service desks.

Each core module—semantic retrieval, response generation, and user interaction was encapsulated as an independent service, accessible via REST APIs. This design allowed for:

- **Reusability:** Components could be reused in related support tasks, such as documentation search or automated ticket triage.
- **Extendibility:** New data sources (e.g., ticket history or monitoring logs) could be incorporated with minimal structural changes.
- **Scalability:** The system could scale horizontally by replicating services behind load balancers, or vertically through performance tuning of the language model endpoints.

This modular approach ensures that the RAG-based support system can evolve as organizational needs change, supporting broader applications beyond technical documentation lookup.

3.7 Reliability and Validity

3.7.1 Reliability

Reliability was ensured by following a consistent and structured methodology across all phases of system development. Core processes such as document chunking, embedding generation using Azure AI Search, and retrieval were applied uniformly across the dataset. The use of Azure AI Studio and reproducible workflows ensured consistency and traceability. Evaluation was repeated across multiple test cycles using standardized metrics such as BLEU and ROUGE L to verify the stability and accuracy of the generated responses.

3.7.2 Validity

Validity was established by aligning the system design with the defined research objectives. The dataset represented authentic enterprise support scenarios based on internal technical documentation. System testing used realistic user queries, and evaluation employed recognized metrics in language modeling to confirm that the system addressed real support needs. The design and implementation reflected practical enterprise environments, supporting external validity.

3.8 Ethical Considerations

The study adhered to ethical research principles to promote fairness, integrity, and responsible system development.

3.8.1 Objectivity

Objective evaluation was maintained by using automatic performance metrics without manual tuning or intervention to influence outcomes. This ensured a fair assessment of the system's performance.

3.8.2 Transparency

All tools, configurations, models, and data sources were documented in detail. The methodology and findings were reported clearly, including limitations encountered during development and evaluation.

3.8.3 Intellectual Property

All third party tools, libraries, and resources were properly cited. The design and implementation of the Retrieval Augmented Generation system were original and contributed to academic knowledge in technical support automation.

3.8.4 Data Responsibility

No personal or sensitive user data was involved. The system was built and tested using technical documentation. The model outputs were evaluated to ensure fairness and avoid biased or exclusionary responses.

3.8.5 Legal Compliance

The study followed relevant data usage and licensing policies. All data used for development and testing was drawn from authorized internal available sources.

3.9 Conclusion

This chapter detailed the methodology used to develop and evaluate the Retrieval Augmented Generation system. The CRISP DM framework structured each phase, from problem definition to system deployment. Methodological consistency ensured reliability, and system validity was supported by realistic data and evaluation strategies. Ethical considerations were addressed through responsible use of data, transparent reporting, and legal compliance. This foundation supported the practical implementation of a system designed to improve the effectiveness of enterprise technical support.

Chapter 4

System Analysis, Design, and Architecture

4.1 Introduction

This chapter presents the design and architecture of the Retrieval Augmented Generation (RAG) system developed to enhance the accuracy and efficiency of enterprise technical support. The system combines semantic retrieval with generative language modeling to provide responses that are both context aware and grounded in internal documentation.

Section 4.2 outlines the system analysis, beginning with an overview of common user challenges observed on support channels (Subsection 4.2.1). Subsection 4.2.2 defines the system requirements, covering both functional and technical specifications. Subsection 4.2.3 explains the system architecture, including embedding generation, vector based semantic retrieval, system components, and the user interface.

These sections form the basis for system implementation by demonstrating how user queries are retrieved and addressed with contextual accuracy and minimal latency.

4.2 System Analysis

4.2.1 Thematic Analysis

An analysis of user feedback from support channels revealed four key challenges. First, users reported delayed responses, especially for time sensitive issues. Second, repetitive queries

placed a burden on support teams and slowed response times. Third, many users expressed dissatisfaction with generic responses that lacked context or personalization. Finally, static resources such as FAQs and documentation were often underutilized due to poor navigation and limited search capabilities.

Users showed a strong preference for interactive support tools and highlighted that a chatbot integrated with internal documentation would improve access to relevant information and streamline issue resolution.

4.2.2 System Requirements

The system was designed to support accurate query processing, semantic retrieval, and response generation for technical support. By combining vector based search with a generative model, it enables users to submit questions and receive context aware responses.

The following were the key system requirements:

i. Query Processing (REQ 001): Enables accurate interpretation of user queries and converts them into vector embeddings for semantic search.

ii. Document Retrieval (REQ 002): Uses Azure AI Search to perform similarity matching against indexed support documentation and retrieve the most relevant content based on semantic similarity.

iii. Response Generation (REQ 003): Uses GPT 4, hosted on Azure OpenAI, to generate responses based on the retrieved content and original query. The output is expected to be coherent, relevant, and grounded in internal documentation.

4.2.3 System Architecture

The system was designed to support efficient query processing, semantic retrieval, and response generation for technical support. It consisted of modular components working together to provide accurate and timely assistance to users.

At the core of the system was a user interface built using Azure AI Studio. This interface allowed users to submit queries, view retrieved documentation, and receive responses generated through a Retrieval Augmented Generation approach. Unlike traditional keyword search, the system used semantic retrieval to surface relevant content based on meaning rather than exact terms. The overall system architecture is shown in Figure 4.4, illustrating the interaction between the user interface, retrieval components, and response generation module.

For document retrieval, the system used Azure AI Search to index and store internal documentation. The *text-embedding-ada-002* model from Azure OpenAI was used to convert documents and queries into vector representations. Azure AI Search then performed similarity matching to retrieve the most relevant content.

The retrieved content and user query were passed to GPT 4, hosted on Azure OpenAI. The model generated a response that was context aware and grounded in the retrieved content, ensuring accuracy and relevance.

By combining vector based retrieval with large language model generation, the system delivered accurate responses with reduced latency and improved user experience in technical support.

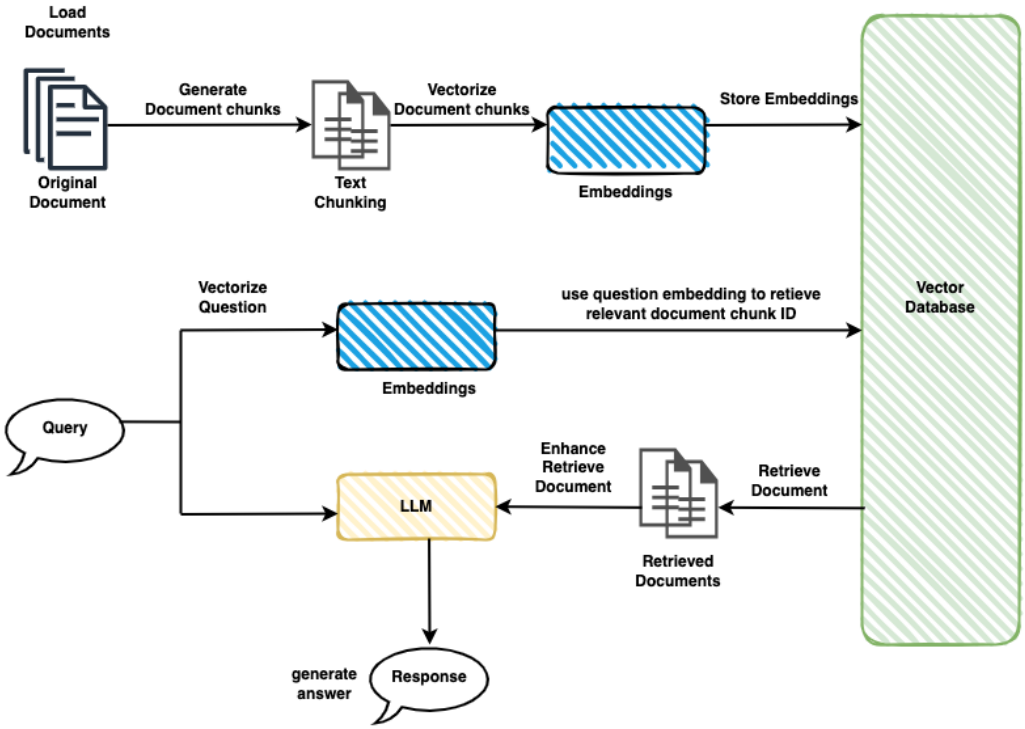


Figure 4.1: System Architecture

Embedding and Vectorization

The system uses Azure AI Search to embed and index internal support documentation. Text is processed using the *text-embedding-ada-002* model from Azure OpenAI, which converts both user queries and document chunks into vector representations for semantic retrieval.

Internal documentation, including the internal guide documentation and frequently asked questions, was split into smaller, meaningful sections before embedding. This improves retrieval accuracy by enabling fine grained matching between queries and content.

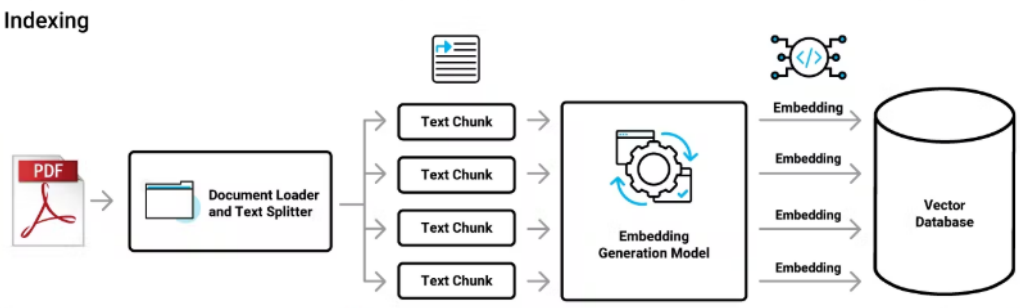


Figure 4.2: Embedding and Indexing Process

Indexing and Similarity Search

Embedded document chunks are indexed in Azure AI Search, which supports vector based similarity matching. At runtime, a user query is embedded using the same model and compared against the stored vectors. Azure AI Search performs a fast similarity search to retrieve the most relevant document chunks.

The system ranks retrieved results using **cosine similarity**, which measures the closeness between the query and document vectors in high dimensional space. Given two vectors A and B, cosine similarity is computed as:

$$\text{cosine\_similarity}(A,B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{4.1}$$

A higher score indicates stronger semantic similarity, allowing the system to select the most relevant content for response generation.

Retrieval and Generation Integration

The top retrieved document chunks are combined with the user query and passed to GPT 4 via Azure OpenAI. The model generates a context aware response grounded in the retrieved content. This end to end pipeline ensures that responses are accurate, relevant, and aligned with enterprise documentation.

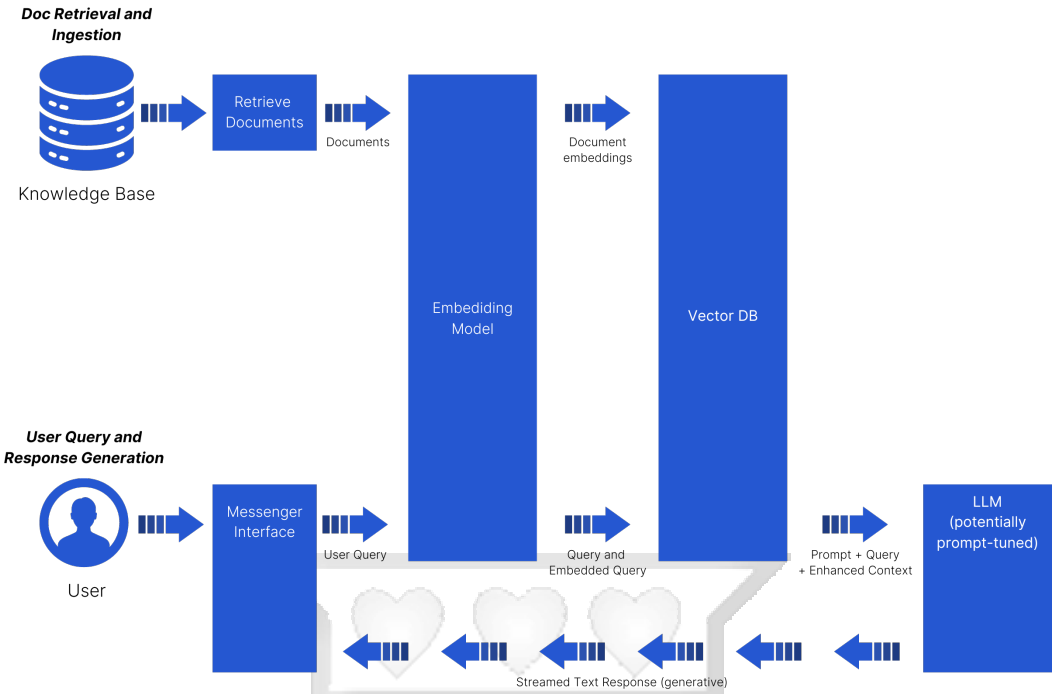


Figure 4.3: Retrieval and Generation Process

User Interface Design

The user interface was built using Azure AI Studio to support seamless interaction with the RAG system. It allows users to submit queries, receive generated responses, and view previous interactions in a clear and accessible format.

As shown in Figure 4.4, the interface includes a text input area for submitting queries, a response display panel, and a query history section. Submitted queries trigger document retrieval and response generation using the embedded knowledge base and GPT 4.

The query history panel enables users to revisit previous responses, improving continuity and reducing repeated queries. The interface was designed to be simple, responsive, and focused on enhancing the support experience.

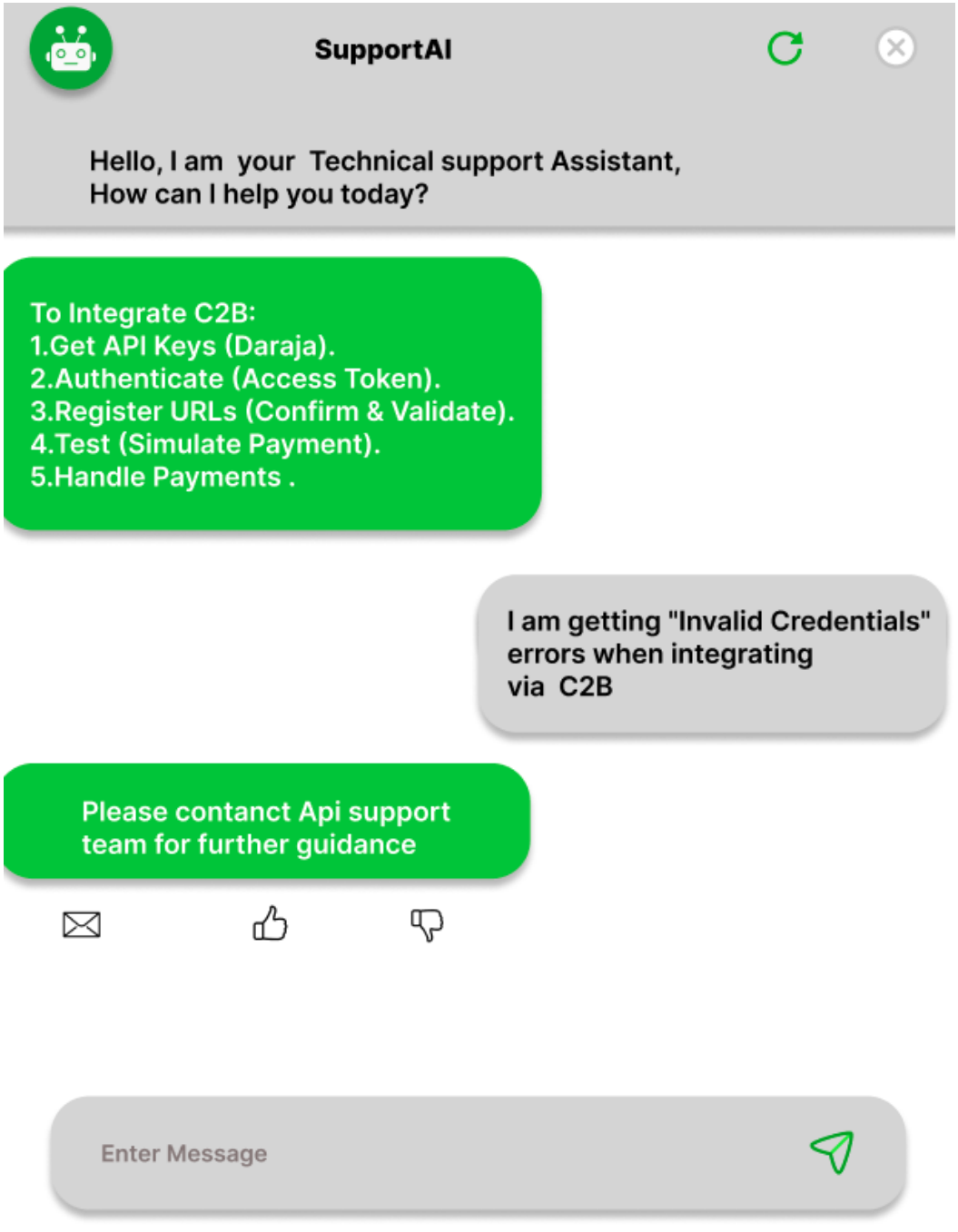


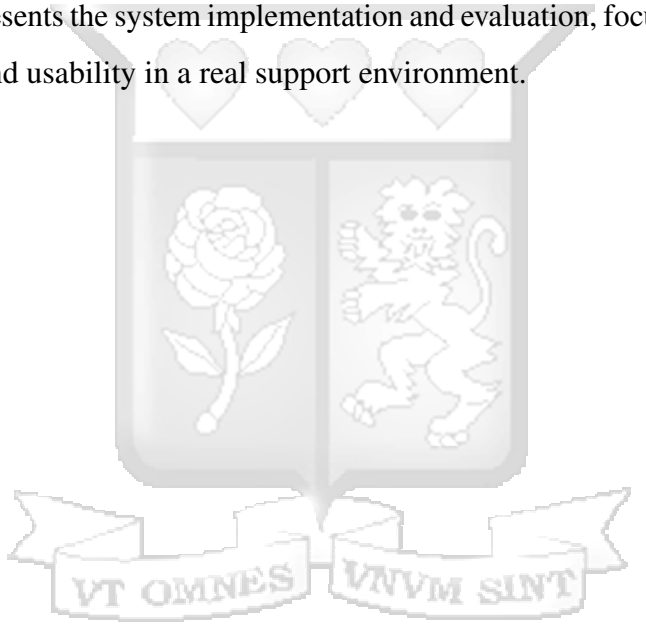
Figure 4.4: Chatbot User Interface

4.3 Conclusion

This chapter outlined the design and architecture of the Retrieval Augmented Generation system, highlighting key components such as semantic retrieval, response generation, and user interface design. The system integrates Azure AI Search for document indexing and similarity matching, the *text-embedding-ada-002* model for embedding, and GPT 4 for generating context aware responses.

By combining these components into a structured pipeline with a user friendly interface, the system supports accurate and efficient resolution of technical support queries.

The next chapter presents the system implementation and evaluation, focusing on performance, response quality, and usability in a real support environment.



Chapter 5

System Implementation and Model Evaluation

5.1 Introduction

This chapter outlines the implementation and evaluation of the Retrieval Augmented Generation (RAG) system developed to improve the accuracy and contextual relevance of responses within an enterprise IT technical support setting. The system combines Azure AI Search for semantic document retrieval with GPT-4, accessed via Azure OpenAI, to generate grounded and informative responses.

Section 5.2 describes the system setup, including document embedding and integration of the retrieval and generation pipeline. Section 5.3 explains the configuration of generation parameters and prompt engineering strategies used to optimize response quality. Section 5.4 presents the evaluation procedures, detailing both automated and manual methods used to assess system performance and effectiveness.

5.2 System Implementation

The Retrieval Augmented Generation (RAG) system was developed to address common challenges in enterprise technical support, including delayed responses, repetitive inquiries, and a lack of contextual understanding. The implementation focused on four core components: **document embedding and indexing, query processing, vector-based retrieval,** and **response generation**.

The process began with preparing internal enterprise documentation for semantic retrieval. These documents were segmented into smaller, meaningful chunks and embedded using the *text-embedding-ada-002* model from Azure OpenAI. The resulting vector representations were indexed using Azure AI Search to support fast and semantically accurate retrieval.

At runtime, user queries were processed using the same embedding model and submitted to Azure AI Search. The service returned the most contextually relevant document chunks based on vector similarity scores.

The retrieved content, along with the user query, was then passed to GPT-4 via Azure OpenAI. The model generated a contextually grounded response tailored to the specific support scenario.

5.2.1 Prompt Flow and Context-Aware Query Handling

To enable dynamic and context-aware generation, the system leveraged Azure AI Studio's Prompt Flow to orchestrate the entire retrieval and generation pipeline. This flow accepted a user query and corresponding chat history, enriched the input through a sequence of processing nodes, and produced a grounded response using GPT-4o.

As shown in Figure 5.1, the pipeline began with the *modify_query_with_history* node, which used GPT-4o to reformulate the query based on prior conversation context. The updated query was sent to the *lookup* node, where Azure AI Search performed semantic retrieval against a pre-indexed corpus embedded using the *text-embedding-ada-002* model.

The *generate_prompt_context* node then extracted the most relevant document snippets. These were formatted into prompt templates by the *Prompt_variants* node, which supported evaluation of different prompt structures. Finally, the *chat_with_context* node passed the constructed prompt to GPT-4o for response generation.

Hyperparameter Configuration:

- `temperature = 0` was used for both query rewriting and response generation to ensure deterministic outputs. This minimized randomness and increased factual consistency, which is critical in enterprise support contexts.
- `top_k = 2` limited retrieval to the two most semantically relevant chunks. This maintained response precision while keeping within token limits of the generation model.

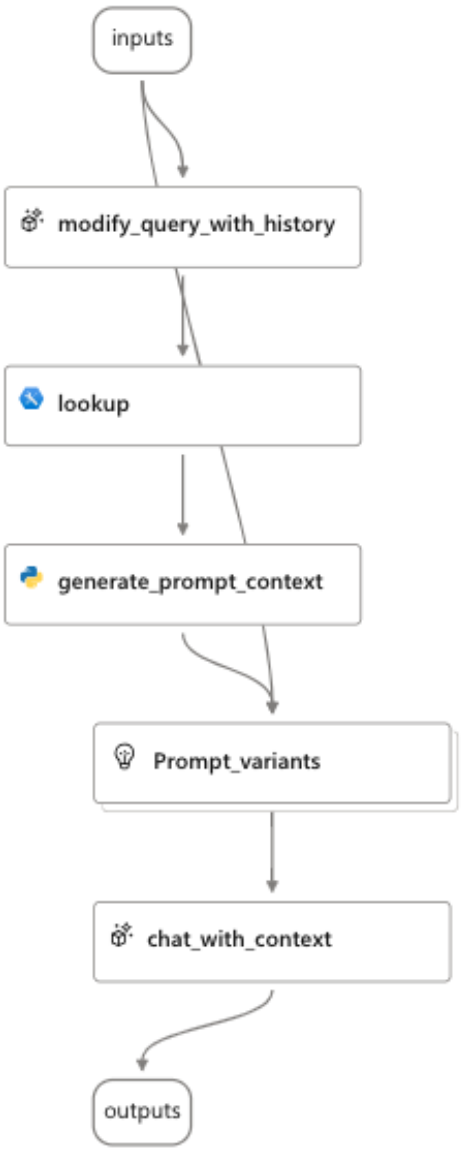


Figure 5.1: Prompt Flow structure for context-aware query processing and response generation

5.3 Model Evaluation

The performance of the Retrieval Augmented Generation (RAG) system was evaluated using a combination of automated metrics and manual human review. This dual approach

provided both statistical and practical insight into the system's effectiveness in answering developer-focused technical support queries.

5.3.1 Automated Evaluation

Quantitative evaluation was performed using four widely adopted natural language generation metrics: F1 Score, BLEU Score, ROUGE-F1, and METEOR. These metrics assessed the degree of similarity between the model's generated responses and reference answers, focusing on token overlap, phrase-level accuracy, and semantic alignment.

A sample of five representative user queries was selected, covering topics such as STK integration, callback configuration, API authentication, and transaction troubleshooting. Table 5.1 summarizes the system's scores for each query.

Query	**F1 Score**	**BLEU**	**ROUGE-F1**	**METEOR**
How do I initiate an STK push?	0.51	0.32	0.55	0.63
What is the callback URL for?	0.28	0.03	0.30	0.31
Which credentials are required?	0.49	0.20	0.48	0.51
How can I check C2B status?	0.47	0.08	0.44	0.41
Purpose of password field in STK?	0.45	0.29	0.50	0.42

Table 5.1: Automated Evaluation Scores for Variant 1

The average results across all queries were:

- **F1 Score:** 0.44 – indicating balanced precision and recall.
- **BLEU Score:** 0.19 – showing partial alignment at the phrase level.
- **ROUGE-F1 Score:** 0.45 – reflecting moderate content overlap with references.

- **METEOR Score:** 0.46 – suggesting good semantic and synonym-level similarity.

These results demonstrate the model's ability to generate accurate and semantically relevant responses, suitable for deployment in technical support use cases.

5.3.2 Manual Evaluation

To complement the automated metrics, a manual evaluation was conducted using Azure AI Studio's built-in review interface. This process assessed the factual accuracy, fluency, and coherence of the generated responses against predefined enterprise support queries derived from internal documentation.

The evaluation framework consisted of two human reviewers independently scoring each response based on the following criteria:

- **Coherence:** Logical flow and structural clarity of the response.
- **Fluency:** Grammatical correctness and natural language usage.

Each query was considered "Passed" if both reviewers marked the response as meeting the criteria (2/2). As illustrated in Figure 6.6, all five sampled queries received a full pass rating, with coherence and fluency consistently marked as satisfactory.

This outcome confirms that the system produced contextually accurate and linguistically polished responses that aligned with human expectations in a real-world technical support scenario.

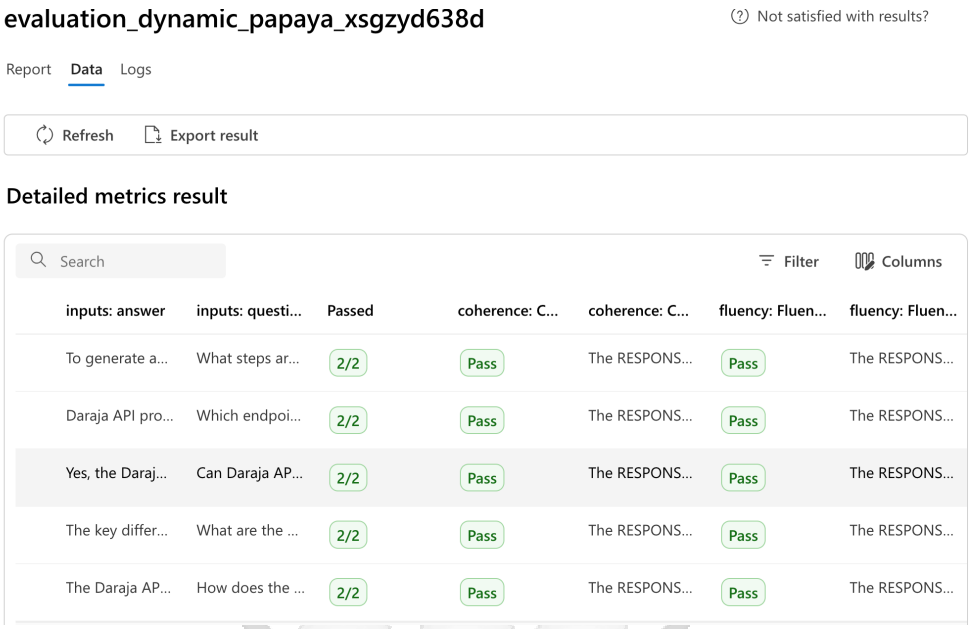


Figure 5.2: Manual evaluation scores showing pass rates for coherence and fluency

Together, the automated and manual evaluations validate the system's ability to deliver high-quality, grounded responses. While quantitative metrics provided measurable performance indicators, human assessment confirmed the practical utility of the chatbot in a support environment.

5.4 Model Deployment and Integration

The Retrieval Augmented Generation system used the GPT 4o model deployed on Azure OpenAI for response generation. GPT 4o was selected for its strong benchmark performance in natural language understanding, multi-turn dialogue, and factual accuracy, making it suitable for enterprise-level technical support.

The model was deployed via Azure AI Studio under the Deployments and Endpoints section. A secure API endpoint and access key were generated for authenticated access. The OpenAI Python SDK was used to integrate the deployed model into the system. Credentials were stored securely using environment variables.

The model was initialized as follows:

```
from openai import AzureOpenAI
from azure.core.credentials import AzureKeyCredential

client = AzureOpenAI(
    api_version="2024-12-01-preview",
    endpoint="https:***********",
    credential=AzureKeyCredential("*********")
)
```

A standard call to the GPT 4o chat completion API was structured as follows:

```
response = client.chat.completions.create(
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "I am going to Paris, what should I see?"}
    ],
    max_tokens=4096,
    temperature=1.0,
    top_p=1.0,
    model="gpt-4o"
)
```

To improve the user experience, streaming output was also supported:

```
response = client.chat.completions.create(
    stream=True,
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "I am going to Paris, what should I see?"}
    ],
    max_tokens=4096,
```

```
    temperature=1.0,
    top_p=1.0,
    model="gpt-4o"
)

for update in response:
    if update.choices:
        print(update.choices[0].delta.content or "", end="")
```

Parameter Configuration:

- `temperature = 1.0` allows balanced creativity and factual accuracy by enabling some variation in token selection, useful for elaborative yet grounded responses.
- `top_p = 1.0` applies nucleus sampling with a broad range of possible outputs, ensuring completeness of responses without restricting the model's expressiveness.
- `max_tokens = 4096` sets the maximum length for the generated response, allowing detailed multi-turn outputs while avoiding token overflows.

This setup enabled secure, real time, and accurate generation of responses within the RAG pipeline.

5.4.1 User Interface

The system provided a simple web-based interface through which users could submit natural language queries and receive generated responses. The interface supported interaction with the RAG model and allowed for seamless testing and evaluation.

5.5 Conclusion

This chapter presented the implementation, cloud deployment, and evaluation of the RAG-based technical support system. The system successfully combined retrieval with language generation to produce accurate and relevant responses. The evaluation results confirmed that it met the research objectives. The next chapter discusses the key findings and outlines areas for improvement.

Chapter 6

Discussions, Conclusions and Recommendations

6.1 Introduction

This chapter presents a discussion of the results from the implementation and evaluation of the Retrieval-Augmented Generation (RAG) system as described in Chapter 5. It examines how the system met the research objectives by analyzing its performance and the quality of responses generated. Section 6.2 discusses the system's performance using automated metrics and expert feedback. Section 6.3 outlines the limitations encountered during development and testing. Section 6.4 provides suggestions for future work aimed at improving the system and extending its capabilities.

6.2 System Performance and Expert Review

The automated evaluation of the Retrieval Augmented Generation (RAG) system was conducted using four standard natural language generation metrics: F1 Score, BLEU, ROUGE F1, and METEOR. These metrics were used to assess the alignment between system-generated responses and expert-written reference answers from internal technical support documentation.

As illustrated in Figure 6.1, the system demonstrated moderate performance across all metrics. F1 and ROUGE F1 scores averaged above 0.44, while METEOR scores indicated strong semantic alignment. BLEU scores were comparatively lower, reflecting the model's tendency

to paraphrase rather than replicate exact n-gram structures. The following subsections provide a detailed breakdown of each metric.

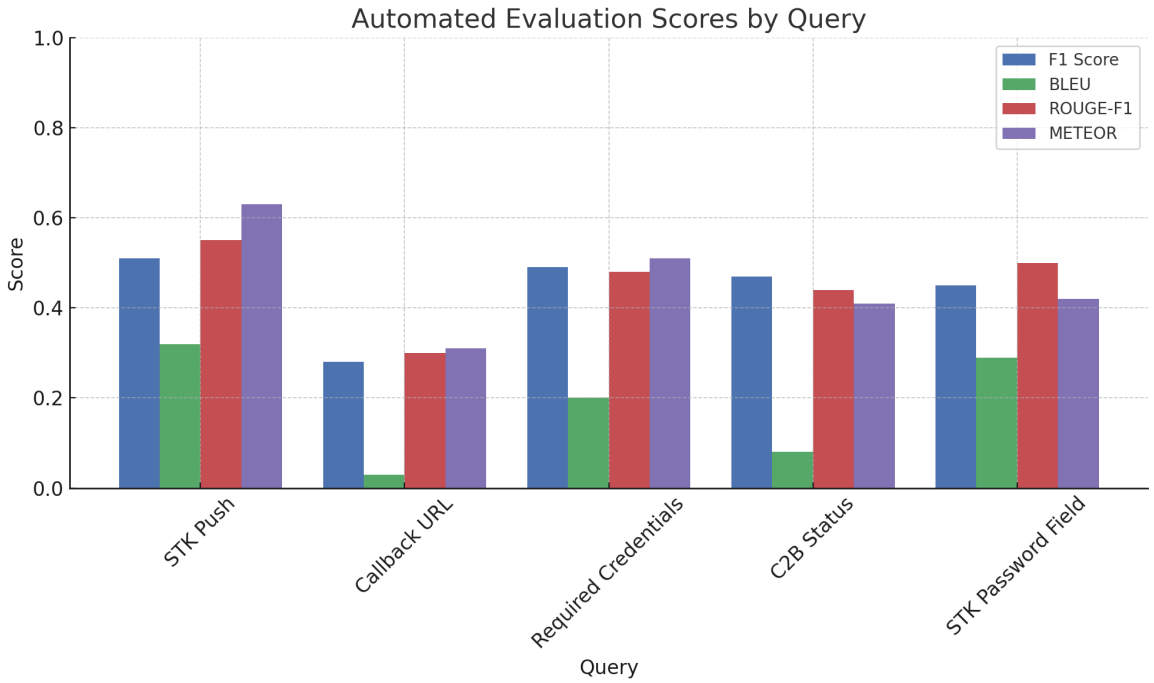


Figure 6.1: Automated evaluation scores by query (F1, BLEU, ROUGE F1, METEOR)

6.2.1 Discussion of Evaluation Results

The F1 Score, which balances precision and recall, was used to assess how effectively the Retrieval Augmented Generation (RAG) system retrieved and applied relevant content. The average F1 Score across all queries was 0.44, indicating moderate accuracy in generating responses that align with reference answers.

As illustrated in Figure 6.2, the highest F1 Scores were observed in the STK Push and Required Credentials queries, both scoring close to 0.50. This suggests that the system performed well in retrieving relevant information for structured and frequently asked queries. In contrast, the Callback URL query recorded the lowest F1 Score at approximately 0.28, likely due to its vague phrasing or open-ended nature.

Overall, F1 Scores across queries ranged between 0.28 and 0.51, which is acceptable for open-domain generation tasks where query variability is high. These results highlight the model's consistency in retrieving semantically relevant content across diverse support queries.

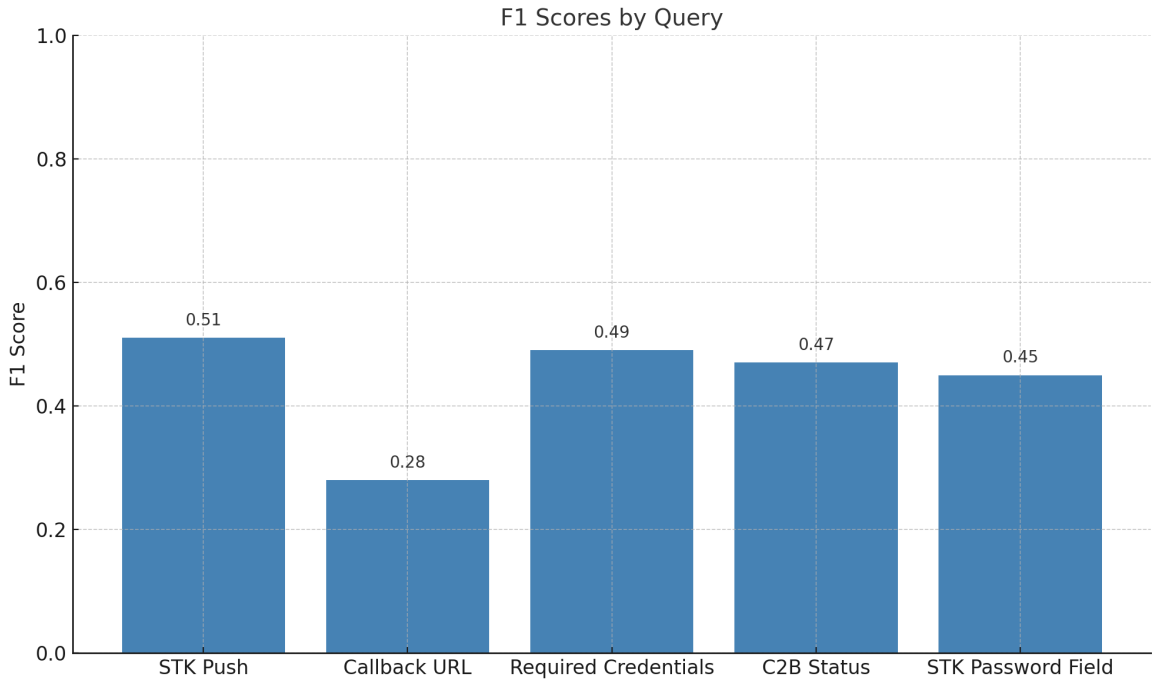


Figure 6.2: F1 scores by query, showing balance between precision and recall across response types

Evaluation Results – BLEU Score

The BLEU Score evaluates how closely system-generated responses match reference answers at the phrase level, using n-gram overlap. As illustrated in Figure 6.3, BLEU scores across the five sampled queries were comparatively lower than the other metrics, with values ranging from 0.03 to 0.32.

The highest BLEU score was recorded for the STK Push query at 0.32, suggesting partial alignment in phrasing between the generated and reference responses. The lowest score, 0.03, was observed for the Callback URL query, indicating significant differences in word choice or phrasing.

These lower BLEU scores reflect the system's use of paraphrasing rather than exact repetition, which is common in open-ended generation tasks. While BLEU is useful for detecting direct overlap, it tends to undervalue semantically accurate but lexically varied responses. This limitation highlights the importance of complementary metrics such as METEOR and ROUGE for a more holistic evaluation.

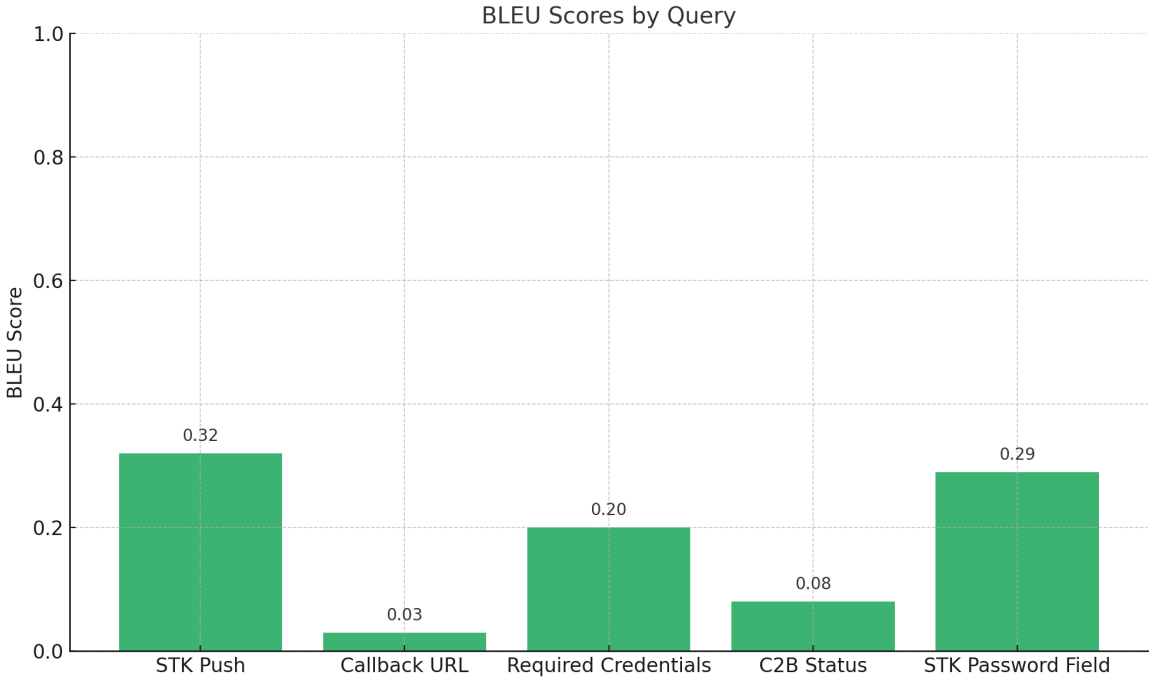


Figure 6.3: BLEU scores by query, showing variation in phrase-level alignment

Discussion of Evaluation Results – ROUGE F1 Score

The ROUGE F1 Score measures the overlap of key phrases and sequences between generated responses and reference answers, focusing on recall and partial matches. This makes it particularly valuable in technical support contexts where expressing the same concept in different ways is common.

As illustrated in Figure 6.4, the ROUGE F1 Score averaged 0.45 across all queries, indicating a strong alignment between generated content and expert-written references. The highest score, 0.55, was recorded for the STK Push query, demonstrating the model's ability to

reproduce critical information using similar phrasing. STK Password Field and Required Credentials also scored highly, at 0.50 and 0.48 respectively.

The lowest score, 0.30, was observed for the Callback URL query. This may be attributed to the abstract or underspecified nature of the question, which could lead to variations in phrasing and reduced lexical overlap.

These results suggest that the model performs well in capturing essential content, even when full phrase-level matches are not present. The consistent ROUGE F1 scores reinforce the model's strength in reflecting key information units found in support documentation.

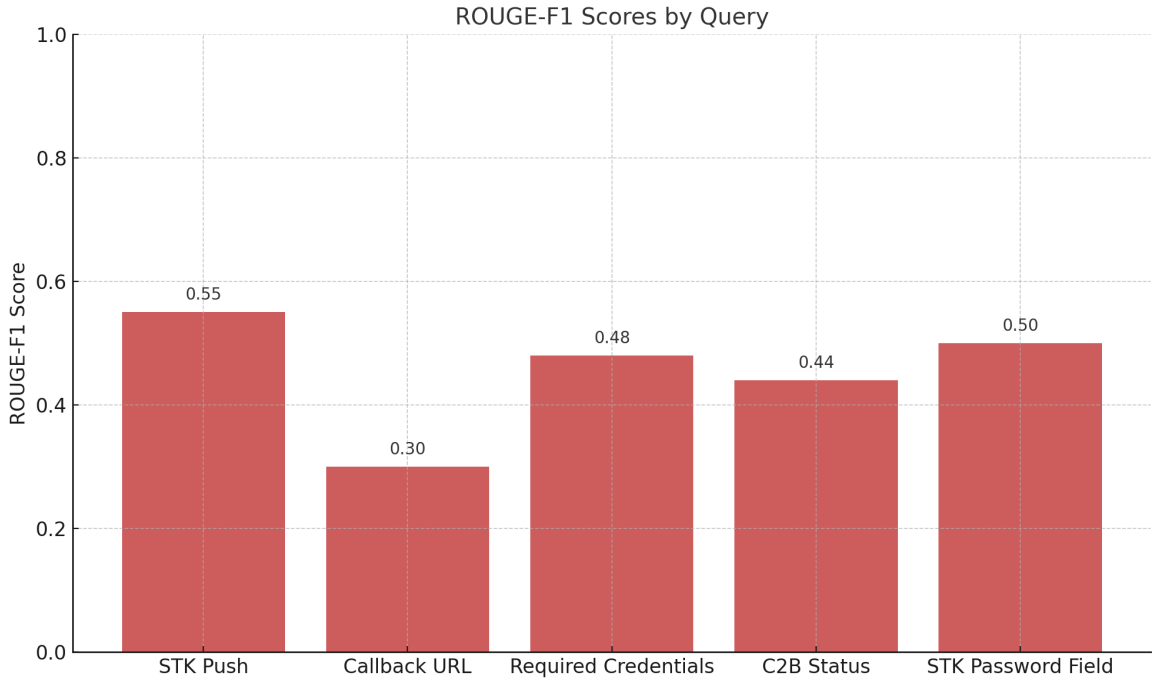


Figure 6.4: ROUGE F1 scores by query, showing degree of key phrase overlap with reference responses

Discussion of Evaluation Results – METEOR Score

The METEOR Score evaluates semantic similarity by accounting for synonymy, paraphrasing, and flexible word order, making it more closely aligned with human judgment than surface-level metrics. It is particularly valuable for assessing open-ended responses where exact lexical overlap is less important than capturing the intended meaning.

As illustrated in Figure 6.5, the system achieved an average METEOR score of 0.46, which indicates strong semantic alignment between generated and reference responses. The highest score, 0.63, was observed for the STK Push query, demonstrating the model's ability to accurately reproduce intended meanings even with varied wording. The Required Credentials query followed closely with a score of 0.51.

The lowest score, 0.31, was recorded for the Callback URL query, reflecting the system's relative difficulty in generating responses with high semantic fidelity for loosely structured or underspecified queries.

Overall, scores above 0.4 suggest that the model consistently preserved the semantic intent of the source content. These results highlight the system's capacity to produce human-like, contextually relevant responses in enterprise support scenarios.

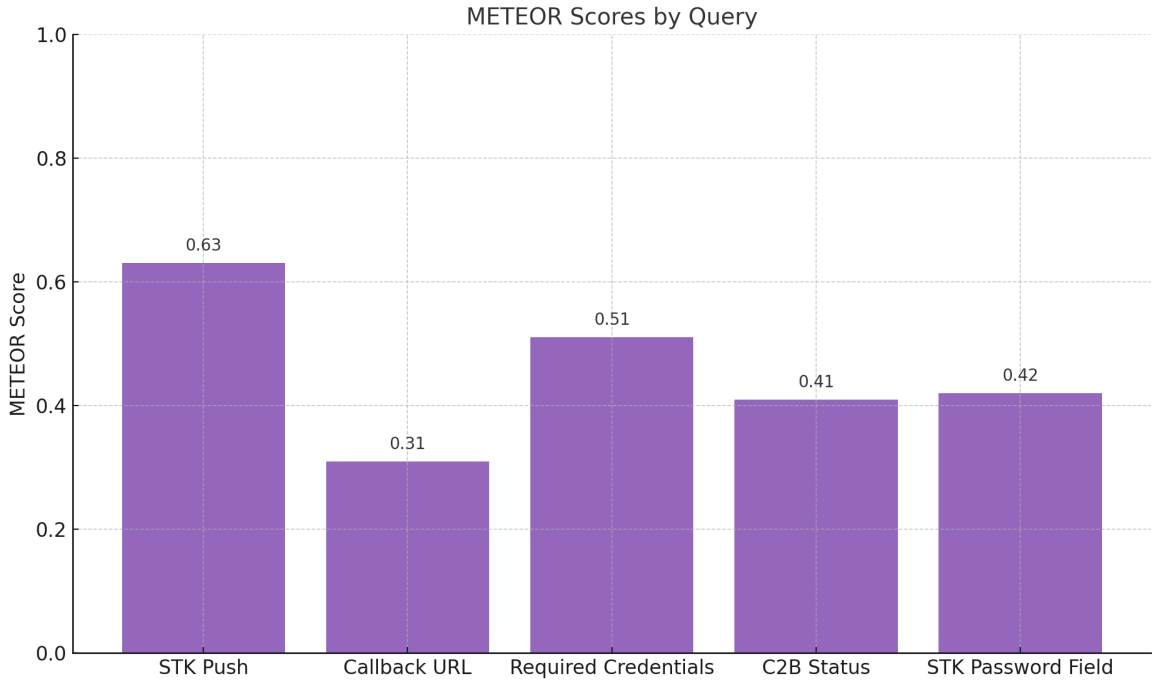


Figure 6.5: METEOR scores by query, reflecting semantic similarity and paraphrasing accuracy

6.2.2 Manual Evaluation

To complement the automated metrics, a manual evaluation was conducted using Azure AI Studio's built-in review interface. This process aimed to assess the quality of generated responses in terms of factual accuracy, fluency, and coherence, particularly in the context of real-world enterprise support.

Evaluation Methodology A set of five queries was sampled from common developer issues, with responses generated by the RAG system. Two human evaluators independently reviewed each response using Azure's AI quality dashboard, applying a standardized rubric across two key dimensions:

Coherence: The logical flow and structural clarity of the response.

Fluency: Grammatical correctness and clarity in natural language usage.

Each response was marked as "Pass" if both reviewers agreed that it met expectations for both criteria. A total of 5 responses (100

Results and Inter-Rater Reliability As illustrated in the dashboard results, all five responses achieved full marks in both categories. The high pass rate reflects the system's ability to generate responses that are not only accurate but also linguistically polished and easy to interpret.

While inter-rater agreement was not computed using a formal statistical measure like Cohen's Kappa, consensus was reached through independent scoring followed by discussion on any discrepancies. The agreement rate was effectively 100% across all test items.

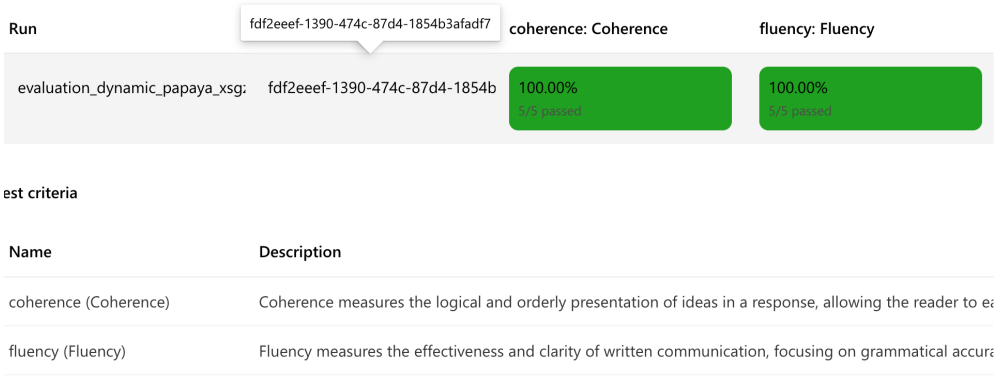


Figure 6.6: Manual evaluation scores showing pass rates for coherence and fluency

6.2.3 Limitations

The RAG-based technical support system demonstrated strong performance and reliability in generating relevant responses; however, several limitations were observed during development and testing that warrant consideration for future improvement.

One key limitation is the system's reliance on a static knowledge base, which restricts its ability to incorporate real-time updates or newly published documentation. This affects its adaptability in fast-changing technical environments where content evolves frequently.

The use of GPT-4, while effective in producing high-quality, context-aware responses, demands considerable computational resources and GPU support. This requirement limits the system's suitability for low-resource or lightweight deployment contexts, especially in environments where scalability and infrastructure cost are critical concerns.

Quantitative evaluation using automated metrics such as BLEU, ROUGE, METEOR, and F1 Score provided a measurable baseline for performance. However, these metrics do not fully capture subjective aspects such as user satisfaction, clarity, or perceived usefulness. To address this, the system incorporated a user feedback mechanism within the interface to collect real-world insights and support iterative improvement beyond static evaluation metrics.

From a system-level perspective, the integration of multiple components;embedding generation, semantic retrieval, and generative response synthesis introduces technical debt and operational complexity. Coordinating these components over time will require rigorous maintenance, continuous monitoring, and infrastructure upgrades to avoid degradation in performance.

Scalability remains a concern in high-demand scenarios. Without robust load balancing or distributed processing, the system may experience latency or throughput bottlenecks under concurrent usage, which could affect its reliability in enterprise-wide deployments.

Another critical challenge lies in error propagation across pipeline stages. Inaccurate or suboptimal retrieval results may lead to irrelevant or misleading generated outputs, especially since the generative model can confidently articulate incorrect responses a phenomenon known as hallucination. Additionally, the system is vulnerable to data drift, where evolving language, terminology, or query structure over time may weaken semantic matching. If embedding models or document indexes are not updated regularly, retrieval relevance and response accuracy may decline.

These limitations highlight the importance of incorporating dynamic knowledge management, robust system architecture, and continuous model monitoring to ensure the long-term scalability, reliability, and accuracy of the RAG system in real-world enterprise environments.

6.3 Conclusion

This study designed, implemented, and evaluated a Retrieval Augmented Generation system for technical support automation in enterprise environments. The system combined Azure AI Studio for the user interface, Azure AI Search for document indexing and semantic retrieval, the text embedding ada 002 model for vectorization, and GPT 4 from Azure OpenAI for response generation. This architecture supported the entire process from user input to the delivery of accurate and context aware responses based on internal documentation.

Evaluation using both automated metrics such as F1, BLEU, ROUGE, and METEOR, along with expert review, confirmed the system's effectiveness in producing clear, relevant, and

well-structured answers. The user interface was designed to enhance usability by supporting query history and real time feedback. Although the system still relies on a static document base and requires substantial computing resources, it provides a strong foundation for future improvement.

Overall, this research demonstrates the value of Retrieval Augmented Generation in automating enterprise support. It contributes a functional prototype, a well-integrated Azure based solution, and practical insights for building intelligent and scalable support systems.

6.4 Recommendations

To enhance the effectiveness and scalability of the Retrieval Augmented Generation system, several improvements are proposed. The system would benefit from integrating a dynamic knowledge pipeline that allows real time updates to enterprise documentation, ensuring that responses remain accurate and current. Introducing a clarification mechanism could help resolve vague or ambiguous queries by prompting users for more specific input. Expanding the system's capabilities to include multimodal input such as diagrams, tables, or screenshots would support a broader range of technical support scenarios. In addition, adopting an agentic approach where the system can autonomously ask follow up questions, validate responses, or recommend next steps would significantly improve interactivity and task completion. Optimizing deployment through strategies such as model compression or containerized services could also reduce resource consumption and make the system more accessible for diverse enterprise environments.

6.5 Future Work

Future research can extend this work by integrating multimodal capabilities, enabling the system to process visual elements such as screenshots, diagrams, and log files alongside text. This would expand its utility in handling more complex or context-rich support queries.

Incorporating multilingual support would allow the system to serve diverse user bases across different regions, making it suitable for global enterprises. There is also potential to explore the use of reinforcement learning to adapt responses based on user feedback over time, improving performance through continued interaction. Additionally, the Retrieval Augmented Generation approach developed in this study can be adapted for use in other domains such as healthcare, legal advisory, telecommunications, and public service, where timely and context aware assistance is critical.

References

Al-Hawari, F., Barham, H., et al. (2021). A machine learning based help desk system for it service management. *Journal of King Saud University-Computer and Information Sciences*, 33(6):702–718.

Alexey, D. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929*.

Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

Bölücü, N., Can, B., and Artuner, H. (2023). A siamese neural network for learning semantically-informed sentence embeddings. *Expert Systems with Applications*, 214:119103.

CHEN, P., LI, Q., ZHANG, D.-z., YANG, Y.-h., CAI, Z., and LU, Z.-y. (2020). A survey of multimodal machine learning. *Chinese Journal of Engineering*, 42(5):557–569.

Chen, Y.-C., Li, L., Yu, L., El Kholy, A., Ahmed, F., Gan, Z., Cheng, Y., and Liu, J. (2020). Uniter: Universal image-text representation learning. In *European conference on computer vision*, pages 104–120. Springer.

Crosley, N. and Wasito, I. (2023). Improving it support efficiency using ai-driven ticket random forest classification technique. *Sinkron : jurnal dan penelitian teknik informatika*, 8(4):2283–2293.

Deshai, N., Rao, K. V., Chittineni, S., and Ramana, S. (2024). Automated dialogue-based response and resolution of conversational it tickets using deep neural networks. pages 351–366. Springer Science+Business Media.

Devlin, J. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dunnmon, J. A., Ratner, A. J., Saab, K., Khandwala, N., Markert, M., Sagreiya, H., Goldman, R., Lee-Messer, C., Lungren, M. P., Rubin, D. L., et al. (2020). Cross-modal data programming enables rapid medical machine learning. *Patterns*, 1(2).

Ebrahimi, S., Arik, S. O., Nama, T., and Pfister, T. (2024). Crome: Cross-modal adapters for efficient multimodal llm.

Fuchs, S., Drieschner, C., and Wittges, H. (2022). Improving support ticket systems using machine learning: A literature review.

Gandhi, A., Adhvaryu, K., Poria, S., Cambria, E., and Hussain, A. (2023). Multimodal sentiment analysis: A systematic review of history, datasets, multimodal fusion methods, applications, challenges and future directions. *Information Fusion*, 91:424–444.

Gao, J., Li, P., Chen, Z., and Zhang, J. (2020). A survey on deep learning for multimodal data fusion. *Neural Computation*, 32(5):829–864.

Gentsch, P. (2018). *AI in marketing, sales and service: How marketers without a data science degree can use AI, big data and bots*. springer.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. (2021). Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., and Cheng, X. (2019). A deep look into neural ranking models for information retrieval.

Guo, K., Li, X., Fan, T., and Hu, X. (2022). Vanet: a medical image fusion model based on attention mechanism to assist disease diagnosis. *BMC bioinformatics*, 23(1):548.

Gupta, H. S. and Sengupta, B. (2012). Scheduling service tickets in shared delivery. In *Service-Oriented Computing: 10th International Conference, ICSOC 2012, Shanghai, China, November 12-15, 2012. Proceedings 10*, pages 79–95. Springer.

Gupta, P., Ding, B., Guan, C., and Ding, D. (2024). Generative ai: A systematic review using topic modelling techniques. *Data and Information Management*, 8(2):100066. Systematic Review and Meta-analysis in Information Management Research - Part II.

Hou, A. Y., Weller, O., Qin, G., Yang, E., Lawrie, D., Holzenberger, N., Blair-Stanek, A., and Durme, B. V. (2024). Clerc: A dataset for legal case retrieval and retrieval-augmented analysis generation.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. (2021). Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Khan, M. S., Khan, K. I., Mahmood, S., and Sheeraz, M. (2019). Symmetric and asymmetric volatility clustering via garch family models: An evidence from religion dominant countries. *Khan, MS, Khan, KI, Mahmood, S., & Sheeraz, M.(2019). Symmetric and asymmetric volatility clustering via GARCH family models: An evidence from religion dominant countries. Paradigms*, 13(1):20–25.

Kim, W., Son, B., and Kim, I. (2021). Vilt: Vision-and-language transformer without convolution or region supervision. In *International conference on machine learning*, pages 5583–5594. PMLR.

Lewis, M. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kulkarni, S., Ju, D., Lewis, M., Yih, W.-t., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32.

Luan, Y., Eisenstein, J., Toutanova, K., and Collins, M. (2021). Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.

Luschi, A., Petraccone, C., Fico, G., Pecchia, L., and Iadanza, E. (2023). Semantic ontologies for complex healthcare structures: a scoping review. *IEEE Access*, 11:19228–19246.

Marin, R. (2021). Employee engagement: An actual theme, in a permanent evolution. *Journal of Human Resources Management Research*, 1(2):1–15.

Martinez-Rodriguez, J. L., Hogan, A., and Lopez-Arevalo, I. (2020). Information extraction meets the semantic web: a survey. *Semantic Web*, 11(2):255–335.

Meng, X., Du, Y., Zhang, Y., and Han, X. (2023). A survey of context-aware recommender systems: From an evaluation perspective. *IEEE Trans. on Knowl. and Data Eng.*, 35(7):6575–6594.

Mitra, B. and Craswell, N. (2019). An updated duet model for passage re-ranking.

Nguyen, T. (2022). Freshdesk-ticketing system software in digital customer service.

Ni, J., Schimanski, T., Mao, L., Sachan, M., Ash, E., and Leippold, M. (2024). Diras: Efficient llm-assisted annotation of document relevance in retrieval augmented generation.

OpenAI (2023). Gpt-4 technical report. Accessed: 2024-03-25.

Pan, J. J., Wang, J., and Li, G. (2024). Survey of vector database management systems. *The VLDB Journal*, 33(5):1591–1615.

Paulin, G., Scheffler, P., Benz, T., Cavalcante, M., Fischer, T., Eggimann, M., Zhang, Y., Wistoff, N., Bertaccini, L., Colagrande, L., Ottavi, G., Gürkaynak, F. K., Rossi, D., and Benini, L. (2024). Occamy: A 432-core 28.1 dp-gflop/s/w 83

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., and Almazrouei, E. (2023). The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*. Accessed: 2024-03-25.

Poole, D. L. and Mackworth, A. K. (2023). *Knowledge Graphs and Ontologies*, page 701–730. Cambridge University Press.

Prabhu, V. V. D. and Anand, A. (2024). Dexter: A benchmark for open-domain complex question answering using llms.

Qi, J., Sarti, G., Fernández, R., and Bisazza, A. (2024). Model internals-based answer attribution for trustworthy retrieval-augmented generation.

Qu, C., Dai, S., Wei, X., Cai, H., Wang, S., Yin, D., Xu, J., and Wen, J. (2024). Colt: Towards completeness-oriented tool retrieval for large language models.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

rin Lee, H. and Kim, S. (2024). Bring retrieval augmented generation to google gemini via external api: An evaluation with big-bench dataset.

Sansone, C. and Sperlí, G. (2022). Legal information retrieval systems: State-of-the-art and open issues. *Information Systems*, 106:101967.

Sterbentz, M., Barrie, C., Shahi, S., Dutta, A., Hooshmand, D., Pack, H., and Hammond, K. J. (2024). Satyrn: A platform for analytics augmented generation.

Taglino, F., Cumbo, F., Antognoli, G., Arisi, I., D'Onofrio, M., Perazzoni, F., Voyat, R., Fiscon, G., Conte, F., Canevelli, M., et al. (2023). An ontology-based approach for modelling and querying alzheimer's disease data. *BMC Medical Informatics and Decision Making*, 23(1):153.

Tan, H. and Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. (2021). Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., , et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*. Accessed: 2024-03-25.

Tsai, Y.-H. H., Liang, P. P., Zadeh, A., Morency, L.-P., and Salakhutdinov, R. (2018). Learning factorized multimodal representations. *arXiv preprint arXiv:1806.06176*.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. (2022a). Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. (2022b). Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Wu, L.-T., Lin, J.-R., Leng, S., Li, J.-L., and Hu, Z.-Z. (2022). Rule-based information extraction for mechanical-electrical-plumbing-specific semantic web. *Automation in Construction*, 135:104108.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P., Ahmed, J., and Overwijk, A. (2020). Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Yao, Y., Zeng, W., Chen, Y., Zhang, M., and Hu, S. (2020). Efficient and robust multimodal fusion for video action recognition. *IEEE Transactions on Multimedia*, 23:3032–3043.

Zhang, Y., Sidibé, D., Morel, O., and Mériaudeau, F. (2021). Deep multimodal fusion for semantic image segmentation: A survey. *Image and Vision Computing*, 105:104042.

Zhou, J., Liu, D., Wei, F., and Chen, L. (2024). Semi-parametric retrieval via binary token index.

Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., and Chua, T.-S. (2021). Retrieving and reading: A comprehensive survey on open-domain question answering. *arXiv preprint arXiv:2101.00774*.

Appendix A

Similarity Report

Sharon Kemunto

Information Retrieval Techniques for Technical Support Ticket Resolution.pdf

Strathmore University (Main Account)

Document Details

Submission ID
trn:oid:::2945:275138918

Submission Date
Mar 28, 2025, 2:40 PM GMT+3

Download Date
Mar 28, 2025, 2:50 PM GMT+3

File Name
Information Retrieval Techniques for Technical Support Ticket Resolution.pdf

File Size
2.8 MB

113 Pages

21,223 Words

132,000 Characters

22% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

318 Not Cited or Quoted 18%
Matches with neither in-text citation nor quotation marks

74 Missing Quotations 3%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

13% Internet sources
10% Publications
17% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **318** Not Cited or Quoted 18%
  Matches with neither in-text citation nor quotation marks
- **74** Missing Quotations 3%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

13% Internet sources
10% Publications
17% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
arxiv.org		3%
2	Internet	
su-plus.strathmore.edu		1%
3	Submitted works	
University of Arizona on 2024-04-07		<1%
4	Publication	
Pingli Jiang, Ruixuan Fan, Yating Yong. "Retrieval Augmented Generation via Con...		<1%
5	Submitted works	
University of Greenwich on 2024-08-16		<1%
6	Internet	
pure.rug.nl		<1%
7	Internet	
www.coursehero.com		<1%
8	Submitted works	
City University on 2024-03-06		<1%
9	Submitted works	
Liverpool John Moores University on 2024-08-09		<1%
10	Internet	
core.ac.uk		<1%

Appendix B

Ethical Approval

Strathmore UNIVERSITY

10th February 2025

Nyabuti, Sharon Kemunto

168403

sharon.kemunto@strathmore.edu

Dear Sharon,

RE: Information Retrieval Techniques for Technical Support Ticket Resolution

This is to inform you that the Office of Graduate Studies on 7th December 2024 received your request for intervention/assistance following your referral by the Strathmore University Institutional Scientific and Ethics Review Committee (SU-ISERC) to our Office due to the fact that you stated that you had already collected and/or analysed its data prior to seeking Ethical clearance. The ethics approval process is ONLY done before any collection of primary or secondary data.

We have taken note of your response that the information that you provided was misinterpreted. This is a letter for you to proceed with the next steps of your academic requirements.

Please be advised, that in future, all research proposals should be submitted to the SU-ISERC through the RHInnO Ethics platform: https://strathmoreuniversity.rhinno.net/login

Disclaimer: 1) This is not in any way an ethical approval letter. 2) Should there be any legal implications/actions emanating from the research in terms of any ethical violations, you will be personally liable.

Yours sincerely,

Prof. Bernard Shibwabo

Director of Graduate Studies

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu

Appendix C

Approval

30 October 2024

Sharon Kemunto Nyabuti

Email: [redacted]

Dear Sharon,

RE: LETTER OF AUTHORIZATION TO CONDUCT RESEARCH AT [redacted]

We refer to the above matter.

We have reviewed your application dated **8 February 2024** to conduct research within [redacted] PLC for your academic project titled **"Leveraging Natural Language Processing(NLP) for enhanced Operational Resilience and Customer Support".**

We are pleased to inform you that your application has been approved and you may proceed to carry out the research by sharing a digital questionnaire to the targeted staff members. Please note that the results must be shared with the Resources Division before they are published.

We will accord the necessary assistance for the completion of the above research. For any clarifications kindly get in touch with HR team through [redacted] or [redacted]

Yours Faithfully,

[redacted]

Figure C.1: Approval letter

Appendix D

Questionnare

Information Retrieval Techniques for Technical Support Ticket Resolution

This questionnaire aims to gather feedback from developers to identify how a chatbot can improve their experience on the Technical Support Platforms.

The focus is on addressing time sensitive needs, managing high volumes of queries, and leveraging Machine Learning (ML) and Natural Language Processing (NLP) to enhance support.

Developer Experience with Current Support Channels

1. Which support channels do you primarily use when seeking help on the Technical Support Platforms? (Select all that apply)

- Email support
- Remedy
- Skype
- Developer Forum
- Virtual meetings
- Other (please specify): _____

2. How satisfied are you with the current response time for your queries on these channels?

- Very dissatisfied
- Dissatisfied
- Neutral
- Satisfied
- Very satisfied

3. What are the most common issues you face with the existing support channels? (Select all that apply)

- Delayed responses to time sensitive issues
- Repetitive and recurring queries not being addressed efficiently •
  Lack of personalized support for complex issues
- Difficulty finding relevant information quickly

- Overwhelming volume of queries in the support email folder
- Other (please specify): _____

Potential Chatbot Integration

4. How likely are you to use a chatbot if it could provide instant responses to your queries on the Technical Support Platforms?

- Very unlikely
- Unlikely
- Neutral
- Likely
- Very likely

5. Which features would you find most valuable in a chatbot integrated into the Technical Support Platforms? (Select all that apply)

- Realtime responses to API and technical queries
- Automated solutions to repetitive inquiries
- Guidance on navigating API documentation and troubleshooting •
  Proactive suggestions based on frequently asked questions
- Integration with other support channels (e.g., Remedy, Skype, Developer Forum)
- Other (please specify): _____

Improving Developer Engagement

6. How often do you engage with the FAQs or static resources on the Technical Support Platforms?

- Never
- Rarely
- Occasionally
- Frequently
- Always

7. In what ways do you think a chatbot could improve your engagement with the Technical Support Platforms? (Select all that apply)

• Providing instant support without waiting for manual responses • Offering personalized and relevant answers based on past interactions
• Reducing the need to search through static FAQs
• Streamlining the support process for common and repetitive tasks • Other (please specify): _____

Repetitive Tasks and Automation

8. What repetitive tasks do you often encounter that could be automated through a chatbot? (Select all that apply)

• Answering common API usage questions
• Troubleshooting error codes
• Providing API key and authentication guidance
• Redirecting to appropriate documentation links
• Other (please specify): _____

9. How do you feel about the use of Machine Learning (ML) and Natural Language Processing (NLP) to automate support responses on the Technical Support Platforms?

• Strongly opposed
• Opposed
• Neutral
• Supportive
• Strongly supportive

Feedback and Suggestions

10. What improvements would you like to see in the support system on the Technical Support Platforms?

• Faster response times for urgent issues
• Better handling of high-volume queries
• Improved accuracy of responses through AI technologies •

Appendix E

Informed Consent Form

Informed Consent Form

Title of Study:Information Retrieval Techniques for Technical Support Ticket Resolution

Section 1: Information for Participants

Principal Investigator: Sharon Kemunto Nyabuti
Institutional Affiliation: Strathmore University
Contact Information: sharon.kemunto@strathmore.edu

Purpose of the Study
This study explores how developers interact with support systems and evaluates how Information Retrieval (IR) techniques can enhance technical support ticket resolution. The findings will contribute to the design of an automated system that improves response accuracy, reduces resolution time, and enhances the overall support experience for developers.

What Participation Involves
If you agree to participate:

1. You will be asked to complete a short digital questionnaire about your experience with existing support channels, challenges faced, and suggestions for improvement.
2. The questionnaire will take approximately 10–15 minutes to complete.

Voluntary Participation
Participation is entirely voluntary. You may decline to participate or withdraw at any time without any consequences or penalties. You can also skip any questions you do not wish to answer.

Confidentiality and Data Security
All responses will be anonymized, and data will be securely stored. Only the research team will have access to the information you provide. No personally identifiable information will be shared or published. Data will be used strictly for academic purposes.

Risks and Benefits
There are no foreseeable risks associated with participation in this study. Your insights will help improve support experiences on the Technial Support Platform and contribute to developing a more efficient support system.

Contact Information for Questions
If you have any questions or require clarification about this research, you can contact:

- **Researcher**: Sharon Kemunto Nyabuti, sharon.kemunto@strathmore.edu
- **Supervisor**: Allan Omondi, aomondi@strathmore.edu
- **Strathmore University Ethics Review Board**: ethicsreview@strathmore.edu

Section 2: Consent

I have read and understood the information provided about this study. I voluntarily agree to participate and know that I can withdraw without explanation.

Please indicate your agreement by ticking the boxes:

- □ I agree to participate in this research.
- □ I agree to have my responses included in the data analysis for this study.

Appendix F

Work Plan

Research Work Plan (April – August 2025)

Information Retrieval Techniques for Technical Support Ticket Resolution

The following is a detailed outline of the tasks and activities involved in this study:

Literature Review & Theoretical Foundation – 2 weeks
Conduct an in-depth review of literature on RAG, chatbots, IR theory, and existing support systems.

System Design and Dataset Preparation – 2 weeks
Define the chatbot architecture, collect relevant technical documentation, and structure the dataset.

Model Development and Implementation – 4 weeks
Build and fine-tune a RAG-based chatbot integrating dense retrieval and generative modules.

Model Evaluation and Testing – 2 weeks
Assess performance based on relevance, accuracy, and response quality using qualitative and quantitative methods.

Discussion, Conclusion, and Recommendations – 1 week
Interpret results, discuss implications, challenges, and suggest improvements or future directions.

Documentation and Dissertation Writing – 2 weeks
Compile all findings, analysis, and insights into a final cohesive thesis document.