KNOWLEDGE GATE: A PERSONALIZED GATEWAY FOR A SECURE, AND CUSTOM LLM

Sabari Vishnu Jayanthan, Jaikrishnan
Jerald, Constantino
Jhon Vincent, Gupo

Operations Excellence, Data Analytics Engineering
Western Digital Corporation, 109 Technology Ave., SEPZ, Laguna Technopark, Binan, Laguna, Philippines
sabari.vishnu.jayanthan.jaikrishnan@wdc.com, jerald.constantino@wdc.com, jhon.vincent.gupo@wdc.com

ABSTRACT

This paper presents Knowledge Gate, a state-of-the-art system that seamlessly integrates user-provided documents into a large language model (LLM)-powered chatbot's knowledge base. Traditional chatbot systems require manual tuning, data preprocessing, or complex pipeline development to ingest and utilize new information. Knowledge Gate eliminates this bottleneck by automating the ingestion, parsing, and embedding of structured and unstructured documents, such as PDFs, Word, and PowerPoint files, directly into the chatbot's context-aware memory architecture. Leveraging advanced techniques in natural language understanding, retrieval-augmented generation (RAG), and vector-based semantic search, Knowledge Gate enables dynamic, contextually accurate responses grounded in user-specific content without retraining the underlying model. This approach significantly reduces integration effort, enhances adaptability for domain-specific applications, and democratizes access to enterprise-level knowledge automation. Experimental evaluations demonstrate high relevance scores and accuracy across multiple use cases, including customer support, technical documentation access, and knowledge management systems.

1.0 INTRODUCTION

The emergence of Large Language Models (LLMs) has transformed the landscape of natural language processing (NLP), enabling human-like dialogue, text summarization, code generation, and more [1]. Despite their remarkable capabilities, current LLM-based systems are inherently limited by static training data and lack mechanisms for incorporating user-specific or context-specific knowledge on the fly. This bottleneck has sparked a growing demand for systems that can dynamically augment LLMs with external knowledge, especially in enterprise and research settings where proprietary information is often stored in document formats inaccessible to traditional chatbot architectures.

To address this challenge, we present Knowledge Gate—an intelligent document ingestion and knowledge augmentation framework that enables LLM-powered chatbots to dynamically integrate user-uploaded files into their operational context. Knowledge Gate supports three commonly used document formats: Microsoft PowerPoint (.pptx), Microsoft Word (.docx), and Portable Document Format (.pdf). These file types constitute most of the business and academic documentation, making Knowledge Gate a practical solution for real-world use cases [2].

The system is designed around a pipeline that combines document parsing, text extraction, semantic chunking, vector embedding, and retrieval-augmented generation (RAG). Upon receiving a document, Knowledge Gate preprocesses its contents, breaks them into semantically meaningful segments, and embeds them into a vector database using a high-dimensional embedding model. During user interaction, relevant document fragments are retrieved and used as contextual input to the LLM, significantly improving the chatbot's ability to generate accurate and context-aware responses [3], [4].

Several frameworks have explored retrieval-based enhancements to LLMs, including systems like LangChain [5] and Haystack [6]. However, these frameworks often require significant manual configuration or code-level integration. Knowledge Gate distinguishes itself by offering a no-code, automated interface that abstracts away the underlying complexity, allowing end-users to enrich their chatbot's knowledge base simply by uploading supported files. This design prioritizes usability, real-time performance, and scalability, making it suitable for both technical and non-technical stakeholders.

Furthermore, Knowledge Gate ensures that all processing is carried out with respect to data privacy and organizational compliance requirements. No document is stored persistently unless explicitly permitted, and embeddings are confined to secure, session-specific environments.

2.0 REVIEW OF RELATED WORK

The integration of user-specific documents into conversational AI systems has gained momentum in corporate and industrial environments, particularly within knowledge-intensive sectors such as manufacturing, automotive, and high-tech assembly. The motivation lies in improving operational efficiency, knowledge dissemination, and decision-making by enabling intelligent agents to access domain-specific documentation in real time.

Several major corporations have pioneered the use of retrieval-augmented generation (RAG) and document-based augmentation techniques in manufacturing settings. For instance, Siemens has implemented AI systems that ingest technical manuals, maintenance protocols, and compliance documentation to assist engineers on the production floor through intelligent assistants [7]. These systems leverage semantic search and contextual response generation to help technicians diagnose issues faster and reduce machine downtime.

Similarly, General Electric (GE) has integrated natural language interfaces with digital twins and process documentation. Their systems parse extensive equipment documentation and operational guidelines to support real-time troubleshooting and predictive maintenance in industrial environments [8]. GE's approach exemplifies the potential for LLM-augmented document systems to become part of mission-critical infrastructure.

In the automotive industry, Bosch and Volkswagen have deployed AI solutions to improve training and operations by embedding engineering documents, manuals, and SOPs into virtual assistants used in assembly lines [9]. These assistants can respond to technician queries by retrieving and presenting information from documents that would otherwise require manual lookup, thereby reducing human error and improving productivity.

Another noteworthy example is IBM Watsonx Assistant, which allows enterprises to upload documents in formats such as Word and PDF to create a searchable and interactive knowledge base for internal operations [10]. Used across sectors—including finance, telecom, and manufacturing, this system shares conceptual similarity with Knowledge Gate in its use of semantic chunking, vector embeddings, and retrieval mechanisms.

In terms of open-source tooling, the Haystack framework by deepset.ai has been adopted in various manufacturing use cases where process manuals, ISO certifications, and compliance documents must be queried through conversational agents [6]. This framework allows integration of PDF and DOCX documents and uses dense vector search to deliver contextual responses to production staff.

These industry applications validate the core design of Knowledge Gate, which aims to democratize this capability by offering a lightweight, real-time solution tailored for SMEs and R&D environments. While enterprise solutions often require extensive customization and infrastructure, Knowledge Gate emphasizes usability and immediate value by enabling LLMs to ingest PowerPoint, Word, and PDF files without specialized configuration.

3.0 METHODOLOGY

This study proposes an automated data ingestion pipeline designed to convert user-uploaded documents into a structured format suitable for integration with a large language model (LLM)-powered knowledge base.

3.1 Downloader / Interpreter / Ingester:

The process begins when users upload document files—specifically Microsoft Word (.docx), PowerPoint (.pptx), and Portable Document Format (.pdf) files—via a web-based front-end interface. These uploaded documents are stored in a temporary front-end database for preprocessing. Each document is then programmatically parsed into individual pages, and each page is rendered as a high-resolution image to facilitate accurate text extraction.

Subsequently, these images are passed through an ingestion engine to convert the visual content into machine-readable text. The extracted text for each page is compiled and tagged with relevant metadata, including document ID, page number, and upload timestamp. This structured text data, along with its associated metadata, is then uploaded and securely stored in a designated Amazon S3 bucket for persistent storage and future retrieval.

Following storage, a batch processing mechanism triggers the ingestion of this data into the LLM's vectorized knowledge base. The ingestion process involves tokenization, embedding generation, and semantic indexing to allow context-aware retrieval during user queries. This integration ensures that the knowledge base is enriched with the most recent and relevant user-uploaded content.

Finally, upon successful ingestion of the processed content into the LLM's knowledge base, an automated acknowledgment email is dispatched to the user. This email serves as confirmation that their documents have been successfully processed and assimilated into the intelligent system.

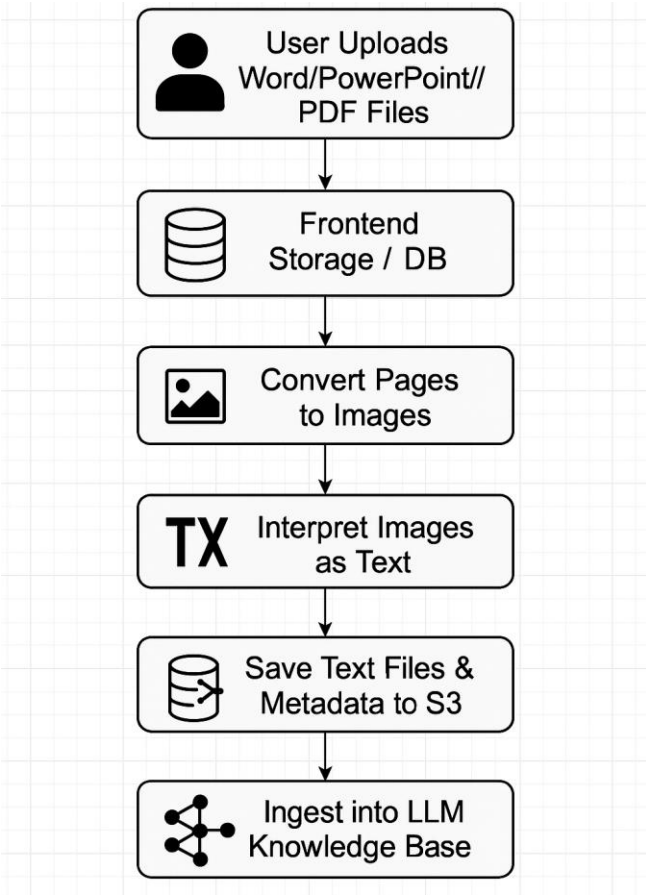


Fig. 1. Workflow diagram of knowledge gate in action.

This methodology offers a scalable, automated, and user-transparent framework for real-time knowledge augmentation in enterprise LLM applications.

3.2 Auto Question and Answering

Auto QnA is a core submodule of Knowledge Gate that, given a target knowledge-graph identifier (kg_id), produces both explicit ("Level 1") and implicit ("Level 2") question–answer pairs for downstream truth-prompting. Figure X shows the end-to-end flow.

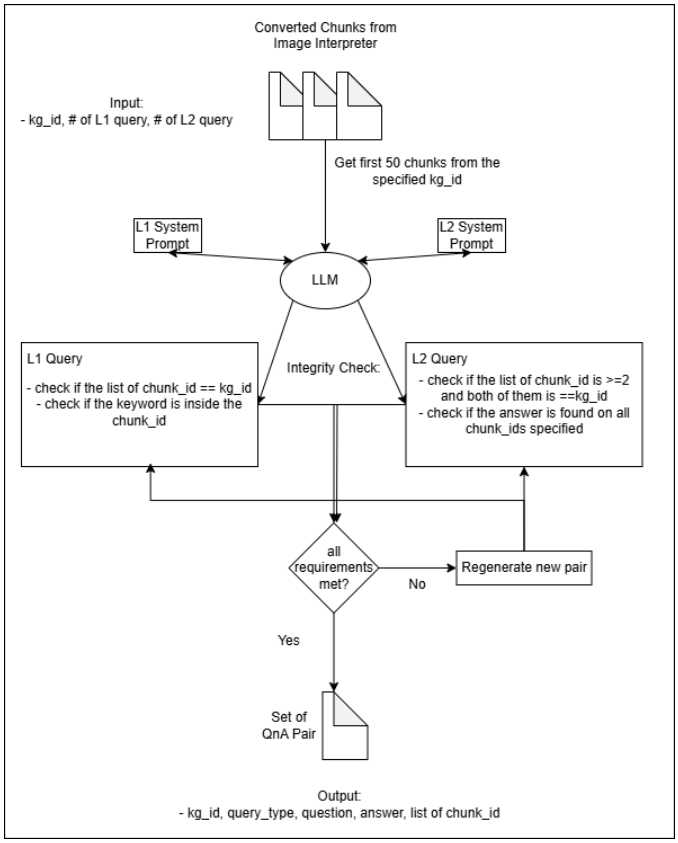


Fig. 2. Workflow diagram of auto question answering

3.2.1 Inputs

- kg_id: Unique identifier of the knowledge graph or document segment
- n_1: Number of Level 1 (explicit fact) queries to generate
- n_2: Number of Level 2 (implicit fact) queries to generate

3.2.2 Inputs Chunk Extraction

1. Source material (e.g., text, OCR'd image) is tokenized into discrete "chunks."

2. Select up to the first 50 chunks tagged with the target kg_id.

3.2.3 LLM Prompting

Two separate system prompts are issued to an LLM, each operating over the same set of retrieved chunks:

- Level 1 System Prompt

"Using exactly one chunk, generate an explicit-fact question and its answer. The question should retrieve a single fact verbatim from that chunk."

- Level 2 System Prompt

"Using at least two distinct chunks, generate an implicit-fact question and its answer. The answer must be supported by synthesizing information across all selected chunks."

3.2.4 Integrity Checks

Each candidate QnA pair is validated before acceptance:

- Level 1

1. Exactly one chunk_id is referenced, and it matches the input kg_id.
2. The answer appears verbatim (or as a direct fact) within that chunk.

- Level 2

1. At least two distinct chunk_ids are referenced; all must match the input kg_id.
2. The answer is substantiated by information present in each referenced chunk.

If any check fails, the pair is discarded, and the corresponding prompt is reissued. This loop continues until the module has produced n_1 valid Level 1 pairs and n_2 valid Level 2 pairs.

3.2.5 Outputs

A collection of structured QnA records, each comprising: *"kg_id, query_type (L1 or L2), question_text, answer_text, [list of chunk_id]"*

Example
- L1:

- **Question:** "What is the capital of France?"
- **Answer**: "Paris"
- **Chunk** IDs: [42]

- L2:

- **Question:** "Why is Paris considered France's primary cultural hub?"
- **Answer:** "Because it contains the highest density of national museums, architectural landmarks, and historical archives across multiple regions."
- **Chunk IDs**: [42, 47]

By enforcing these prompts and integrity checks, Auto QnA guarantees that every generated pair is both syntactically valid and semantically grounded in the source data.

3.3 Truth Prompt Pairs Evaluation

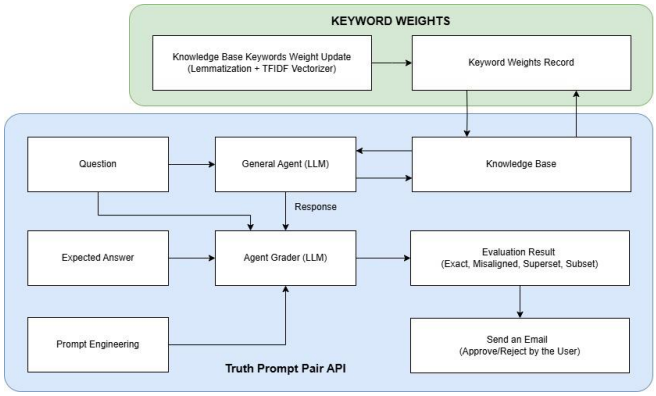


Fig. 3. Truth Prompt Pair High Level Architecture. The Agent Grader evaluates the General Agent's response against the expected answer.

The Truth Prompt Pair (TPP) Evaluation pipeline is designed to support automated response quality monitoring for LLM-based systems. It enables structured evaluation of model-generated outputs by comparing them against predefined expected answers. As depicted in Fig. X, the system architecture consists of two core components: the Keyword Weights Module and the Truth Prompt Pair API. The Keyword Weights Module is responsible for dynamically computing and updating keyword importance across the knowledge base, while the TPP API enables automated evaluation of LLM responses through a structured interaction between the General Agent, the Agent Grader, and the Knowledge Base. The following sections provide a detailed exposition of these core components.

3.3.1 Keyword Weights

An asynchronous text processing pipeline was developed to compute weighted keyword representations using lemmatization and Term Frequency–Inverse Document Frequency (TF-IDF) vectorization. Raw textual data was first normalized by removing punctuation and special characters using regular expressions. Each cleaned chunk was then lemmatized in two sequential passes using the WordNetLemmatizer, initially with the part-of-speech parameter set to noun and subsequently to adjective. This dual-stage lemmatization ensured consistent reduction of words to their canonical forms, improving feature quality. Following lemmatization, the TfidfVectorizer was employed to convert the processed text into numerical vectors. The TF-IDF value for a term t in a document d, within a corpus D, was calculated using the following formulation:

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

Where:

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$$IDF(t, D) = \log\left(\frac{N}{1 + |\{d \in D: t \in d\}|}\right)$$

As described in Eq. 1, $f_{(t,d)}$ represents the frequency of term t in document d, and N denotes the total number of documents in the corpus. The TF component captures term relevance within a document, while the IDF component down-weights terms that appear frequently across multiple documents, thus emphasizing more discriminative keywords. The final output is a keyword-weight dictionary, mapping each term to its computed TF-IDF score. The entire process was designed to run asynchronously to ensure scalability and responsiveness, and operational metrics were logged for performance monitoring.

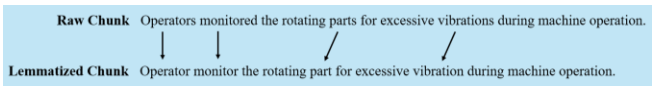


Fig. 4. Lemmatization Example in Manufacturing Context. This figure illustrates the transformation of a raw text chunk into its lemmatized form.

Fig. 4 illustrated the application of this pipeline in a manufacturing domain. Consider the following example text chunk extracted from a production log:

Raw chunk: "Operators monitored the rotating parts for excessive vibrations during machine operation."

The preprocessing begins with lemmatization, which standardizes words to their base or dictionary form. After lemmatizing the text first as nouns and then as adjectives, the resulting chunk is:

Lemmatized chunk: "Operator monitor the rotating part for excessive vibration during machine operation."

This transformation reduces morphological variance (for example, "Operators" to "Operator", "vibrations" to "vibration") and improves consistency in token representation across similar documents.

Subsequently, the TF-IDF vectorizer assigns importance scores to each term within a corpus of similar maintenance logs or quality reports. For instance, if the term "vibration" appears frequently in this document but rarely across the entire corpus, its TF-IDF score will be relatively high, indicating its discriminative relevance. Conversely, generic terms like "machine" or "operation", which are common across documents, will receive lower scores.

Mathematically, the TF-IDF weight for the term "vibration" in this document is computed as:

$$TF - IDF(\text{"vibration"}) = TF(\text{vibration}) \times \log\left(\frac{N}{1 + DF(\text{"vibration"})}\right)$$

Where N is the total number of documents in the corpus and DF("vibration") is the number of documents containing the term "vibration."

This method ensures that domain-specific keywords such as "vibration," "rotating," or "overheating" are emphasized in downstream applications like automated diagnostics, keyword-based retrieval, or LLM prompt evaluation in manufacturing support systems.

3.3.2 Truth Prompt Pair API

The Truth Prompt Pair (TPP) API is a critical component of the system pipeline, designed to evaluate the performance of the General Agent following the ingestion of new knowledge into the Knowledge Base. This pipeline ensures that only high-quality, semantically accurate content is retained for downstream interaction. Prior to permanently storing newly ingested knowledge, the TPP API performs validation to verify that the General Agent can produce accurate and contextually relevant responses using the updated knowledge. The outcome of this evaluation determines whether the knowledge is approved for active use or flagged for rejection.

As illustrated in Fig. X, the system processes a combination of user-provided and auto-generated question–answer (Q&A) pairs. These Q&A pairs originate from the Knowledge Gate UI and the Auto-Q&A Pipeline, respectively. Each question is forwarded to the General Agent, which generates a corresponding response based on the current state of the knowledge base. The generated response is then assessed by the Agent Grader, a dedicated evaluation module that compares the response against the expected answer.

The output of the evaluation process comprises three key elements. First, each response generated by the General Agent is assigned a classification label based on its alignment with the expected answer. The classification falls into one of four categories: (i) Exact, indicating the response fully matches or captures the intended meaning of the expected answer; (ii) Superset, where the response is correct but includes additional information; (iii) Subset, denoting a partially correct response that lacks essential details; and (iv) Misaligned, representing responses that deviate significantly from the expected answer or contain semantic inconsistencies. Second, the evaluation includes a confidence score, which quantitatively reflects the overall quality and reliability of the generated response. Finally, a justification is provided to explain the reasoning behind the assigned classification, offering transparency and interpretability in the decision-making process.

Upon completion of the evaluation, a summary email is sent to the user, presenting the original Q&A pair, the General Agent's response, its classification, and the accompanying justification. If the user approves the evaluation result, the corresponding data is permanently committed to the knowledge base, thereby enabling interactive querying through the General Agent. Conversely, if the user rejects the evaluation, the ingested knowledge is marked as rejected in the backend, and access to this data is restricted from agent-level interactions.

This human-in-the-loop validation framework ensures robust quality control and accountability, safeguarding the integrity of the knowledge base and the reliability of the conversational agent's outputs.

4.0 RESULTS AND DISCUSSION

Interpretation Phase

The interpretation phase involves semantic parsing and contextual understanding of each page. Using 5 parallel interpreter instances, we were able to process all 65 pages in 280 seconds, which translates to an average of ~5.23 seconds per page. This performance metric serves as a baseline for projecting system throughput under increased loads.

Ingestion Phase

Post-interpretation, the pipeline stores both the content and associated metadata (1:1 ratio) into an S3-compatible object store. This amounts to 130 individual uploads for 65 pages. Using a single ingestion instance, the total ingestion time was measured at 400 seconds, averaging ~3.08 seconds per object (or ~6.15 seconds per page, accounting for both content and metadata).

Given the I/O-bound nature of this phase, parallelization yields significant potential benefits. As with interpretation, increasing the number of ingestion workers from 1 to 10 is expected to reduce ingestion time by an order of magnitude, assuming minimal network or disk I/O contention.

Scalability Plan

Based on these observations, we propose scaling the system to 10 interpreter instances and 10 ingestion instances for production-grade throughput. This configuration is projected to handle tons of files, achieving our target for high-volume document processing within a practical operational window..

5.0 CONCLUSION

Knowledge Gate represents a significant advancement in the integration of dynamic, user-provided knowledge into LLM-powered conversational systems. By enabling the ingestion of commonly used document formats—PowerPoint, Word, and PDF—into a chatbot's knowledge base, the system bridges a critical gap between static pretrained models and the ever-evolving informational needs of users. Through automated document parsing, semantic segmentation, and retrieval-augmented generation, Knowledge Gate empowers conversational agents to deliver contextually rich, accurate, and personalized responses in real time.

While the current synchronous processing model presents performance limitations under high concurrency, plans to migrate to an asynchronous architecture aim to ensure scalability and responsiveness as the system scales. This enhancement will position Knowledge Gate as a robust and production-ready solution for industries and organizations that require adaptive, document-aware AI interfaces.

In an era where timely access to domain-specific knowledge is a competitive differentiator, Knowledge Gate sets a forward-looking precedent for how LLMs can be transformed into intelligent, user-augmented assistants capable of real-time learning and contextual reasoning. Future work will explore expanded file type support, multimodal ingestion capabilities, and tighter integration with enterprise knowledge management systems.

6.0 RECOMMENDATIONS

One of the key observations during the deployment and testing of Knowledge Gate was the system's limited scalability under high user load, primarily due to its current synchronous execution model. In the existing architecture, each document ingestion request—whether a PowerPoint (.pptx), Word (.docx), or PDF (.pdf) file—is processed sequentially. This synchronous workflow includes parsing, semantic chunking, embedding generation, and database insertion, all performed in a blocking manner. As a result, when multiple users simultaneously attempt to upload and integrate documents, the system experiences bottlenecks, leading to increased latency and diminished responsiveness.

To address this limitation and improve throughput, we strongly recommend transitioning the ingestion pipeline to an asynchronous architecture. By decoupling the document uploading interface from the processing backend and leveraging asynchronous task queues (e.g., Celery, RabbitMQ, or AWS SQS with background workers), ingestion tasks can be offloaded and executed in parallel. This shift would allow the user interface to remain responsive while long-running operations such as file parsing and embedding computation are handled asynchronously in the background.

Furthermore, asynchronous processing enables the use of concurrent resource management, such as batched vector insertions and parallel I/O operations, which can significantly enhance system performance and resource utilization.

Ultimately, adopting an asynchronous ingestion pipeline will improve scalability, reduce processing delays, and future-proof Knowledge Gate for broader deployment in multi-user environments. This architectural evolution aligns with the best practices in modern AI system design, where non-blocking operations are essential for real-time responsiveness and operational efficiency.

7.0 ACKNOWLEDGMENT

We would like to thank all those who contributed, directly or indirectly, to the development of this work. Their support, encouragement, and insightful input have been greatly appreciated throughout the research process. A special thanks to Mr Alberto Zaldivar for leading this project.

8.0 REFERENCES

[1] T. Brown and e. al, "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems,* vol. 33, pp. 1877-1901, 2020.

[2] M. O'Neill and A. Ellis, "Enterprise Document Usage Trends," *International Journal of Information Management,* vol. 57, pp. 1021-26, 2021.

[3] J. Karpukhin and e. al, "Dense Passage Retrieval for Open-Domain Question Answering," *Proc. of EMNLP,* p. 6769–6781, 2020.

[4] S. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," *arXiv preprint arXiv:2007.01282,* 2020.

[5] H. Chase and L. Johnson, "LangChain: Modular Framework for LLM-Powered Applications," GitHub Repository, [Online]. Available: https://github.com/hwchase17/langchain. [Accessed 2025].

[6] M. de Vries and e. al, "Haystack: A Framework for LLM-Augmented Search," GitHub Repository, [Online]. Available: https://github.com/deepset-ai/haystack. [Accessed 2025].

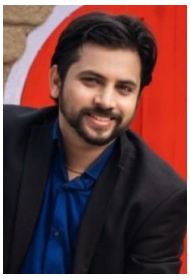
[7] S. AG, "AI-Powered Assistants on the Factory Floor," Siemens Industrial AI Blog, 2022. [Online]. Available: https://new.siemens.com/global/en/company/stories/industry/ai-factory-assistants.html.

[8] G. Digital, "Digital Twin and AI in Predictive Maintenance," GE Reports, 2023. [Online]. Available: https://www.ge.com/digital/press-releases/digital-twin-and-ai. [Accessed 2025].

[9] B. Global, "Artificial Intelligence in Manufacturing," Bosch Press Center, 2021. [Online]. Available: https://www.bosch.com/stories/artificial-intelligence-in-manufacturing/. [Accessed 2025].

[10] I. Corporation, "IBM Watsonx Assistant Documentation," IBM Knowledge Center, 2024. [Online]. Available: https://www.ibm.com/docs/en/watsonx. [Accessed 2025].

9.0 ABOUT THE AUTHORS



Sabari Vishnu Jayanthan Jaikrishnan is a Data Scientist at Western Digital Storage Technologies Limited specializing in Machine Learning, Deep Learning, and Video Analytics projects. He holds a master's degree in Information & Data Science where he researched Automated Machine Learning and a bachelor's degree in Electronics & Communication Engineering.

Jerald Constantino is an associate data scientist at Western Digital Corporation Philippines, specializing in machine vision, machine learning, and generative AI projects. He holds a Bachelor of Science in Computer Engineering from Batangas State University - TNEU and is currently studying for a Master of Science in Computer Engineering at Mapúa University.

Jhon Vincent A. Gupo is an Associate Data Scientist at Western Digital Storage Technologies for Operation Excellence. He holds a bachelor's degree in computer science at Laguna State Polytechnic University – Los Baños.