Intent-Aware Rag Assistant for Technical Documentation

Shubham Kumar[1], Anurag Shakya[2], A. Boobalan[3]
[1,2]*B. Tech CSE, Galgotias University, Greater Noida, India*
[3]*CSE Department, Galgotias University, Greater Noida, India*

***Abstract*—Big pdf and research papers are some of the technical documents that are too long and not easily comprehensible. The traditional methods of searching by using keywords fail to comprehend the context, the Large Language Models (LLMs) can produce misinformation when the solution to the question is not provided in the document So, there is Intent-Aware Retrieval-Augmented Generation (RAG) Assistant that helps the user to engage with the technical documents in the PDF format using natural language. The first step in the system is to determine the query type, i.e., whether it is general or specific, then you choose the appropriate retrieval technique i.e., full document context in the case of general query and semantic search using ChromadDB Database in the case of specific query. In this case, Google Gemini does embed and answer generation, whereas LangChain deals with document chunking and retrieval. The hash-based caching minimizes regular processing and enhances the response time. It enables better accuracy of the experiment, less false perceptions, and faster performance than old-fashioned RAG systems.**

***Index Terms*—RAG, Intent Classification, Google Gemini, ChromaDB, LangChain, Technical Documentation**

I. INTRODUCTION

The basis of engineering, research, and the professional workflow is found in technical documentation, vast collections of manuals, specifications, white papers, and technical papers and such diagrams, equations, and technical jargon of a domain. Researchers who synthesize previous works, engineers that debug systems, and students who work on difficult assignments may find it exceedingly challenging to extract valuable information out of them. The traditional ways of searching like 'Ctrl+f' scans or searching by keyword only give you a superficial result by not taking into consideration the semantic links and subtextual features that relates unrelated ideas in hundreds of pages. Along with the fact that it takes hours of sorting manually, this inefficiency poses the risk of overlooking crucial information, something that would hamper productivity and creativity in technical environments under high pressure.[1]

Retrieval-Augmented Generation (RAG) is a radically different paradigm, a blend of high-quality language models and live document retrieval to generate grounded and hallucination-free replies, tailored to the user requirements. However, the processing of all the queries is identical to those of traditional RAG setups in which the process of vector searching in all embeddings is carried out, regardless of whether it is a query fact or a high-level summary. Such a one-size-fits-all retrieval is a waste of runtime, introduces latency and dilutes accuracy in narrow searches that need verbatim samples of sparse portions or large searches that need synthesis of a document.[2][24].

In this paper, the author introduces a smart query classification layer as an assistant that is a huge language model-driven with an awareness of intent particularly in technical documents. Rapid pre-processing of user queries: general queries like define the caching algorithm cause the whole pre-processing of documents, but narrower queries like outline key methodologies cause single ChromaDBs vector searches. It has been composed of Streamlit, LangChain, and Google Gemini because it has got interactive chat interface, orchestration, and dual use of embeddings and generation. It also calculates MD5 hash to save former embeddings to reduce reload times of files that had been reopened.[19].

Contribution to research

The improvements in real-world PDF empirical experiments are enormous: accuracy of answers is increased on both query flavors, and response times become lower, 1.2 seconds compared to 2.5 seconds in baseline RAG and no hallucinations because of integrity of the context. Some of the contributions besides basic implementation have included a modular architecture

that is scaleable, performance against literature benchmarks like FiD and SBERT and extensions like multi-PDF federation, vision-based diagram parsing. Such strategy makes the technical practitioners have a formidable ally owing to rethink of the interaction of the documents on the basis of less rational form of dialogue, which lie between immediate clarity and information overload. [1][16][22]

II. RELATED WORK

The Retrieval-Augmented Generation (RAG) models combine external knowledge retrieval with generative models to improve the accuracy of the facts in language processing tasks. Lewis et al. proposed the basis RAG architecture, which links dense retrievers with transformer decoders to execute knowledge-intensive NLP, showing fewer hallucinations with document-grounded generation [3][23]. Later developments such as Fusion-in-Decoder (FiD) by Izacard and Grave combined several passages recovered by decoders to transformer decoders to improve multi-document reasoning over open-domain question answering [4]. Modern RAG systems are based on efficient embeddings and indexing. Sentence-BERT (SBERT) by Reimers and Gurevych resulted in dense sentence representations through siamese networks, which facilitated high-quality semantic similarity, which was essential to query-to-document alignment [5]. Hierarchical Navigable Small World (HNSW) graphs pioneered by Malkov and Yashunin, which reduced the cost of approximate nearest-neighbour searches in high-dimensional spaces, became the foundation of vector databases such as ChromaDB, which can provide persistent storage and caching of pipelines related to RAG. Intent-aware routing enhances query refinement, in that they are categorized and then fetched. Zhang et al. demonstrated that intent detection with LLM is effective at agent routing in knowledge discovery, using custom strategies to summarize and fact-finding [7]. Liu et al. also confirmed that pre-retrieval intent classification minimizes vector search overhead and latency of LLM pipelines. These works guide the binary system proposed in the system to classify general and specific queries [8]. Domain-specific RAG applications attack technical corpora. Kumar and Singh developed a RAG chatbot of software manuals and reduced the 37-percent search time of manuals based on semantic search. Sharma, Reddy, and Joshi used transformer-hybrid RAG to engineering documents, which improves expert information retrieval. Patel and Verma achieved 60 percent more accuracy improvement over keywords in legal documents with RAG. This is in contrast to these uniform-retrieval techniques, the present work being the first to include intent-driven dual paths and caching. [9][10][20]

III. METHODOLOGY

The proposed intent-aware RAG assistant is a pipeline-based system (written in Python, served in Streamlit as a user interface), with document processing, intent classification, dual-path retrieval, and response generation modules and represents it as a user interface, Langchain to coordinate activities, Google Gemini to make embeddings and LLM inferences, and ChromaDB to store vectors.

A. Document Processing and Embedding.

PDF files that are posted are text extracted using PyMuPDF and fixed size chunking (1000 characters default with overlap) is used to preserve the context between chunks. All the chunks are converted to dense embeddings using Gemini embedding API and stored in ChromaDB to be indexed by HNSW to facilitate the search of similarities. Embeddings are stored in MD5 hash of file contents so that they are not re-created on the identical re-upload.

The first algorithm is the Document Ingestion and Embedding Caching algorithm (algorithm 1).

Input (PDF document F)
Output (Hash of ChromaDBs set, either as cache memory, or simply compiled)

Steps:
1. Compute a document identifier.
H←"MD5"(F) (1)
Then in case there exists a collection of ChromaDBs which has Halready as a matching element, then vindicate the stocked set.

Extract textual content
T←"PyMuPDF"(F) (2)
Separate the text obtained into fallacious parts.

C=Recursive Character Text Splitter
(T, chunksize=1000, overlap=200) (3)

The measures to be carried out on each chunk c in C are as follows:
a. Generate embedding
e←"GeminiEmbed"(c) (4)
c. saves the chunk and embedding in ChromaDBs on collection H.
Repatriate the H identifier of the ChromaDBs collection.

B. Intent Classification

The user query is used to query a Gemini based classifier which needs to be binary (general (e.g., summaries, overviews) or specific (e.g., factual extractions)). This is a light-weight (less than 100ms) LLM call, which is needed to determine the retrieval strategy, and not to pay the same cost of uniform vector search experienced by the baseline RAG.

Dual-Path Retrieval
General Queries:
Display the entirety of extracted document text to context and synthesize them whole, not by chunks.

Specific Tasks:
Implement semantic search with cosine similarity-based top-k (k=5) chunk semantic search on ChromaDB query embeddings and combine the output of the semantic search process together as focused context.

Table 1. Retrieval Routing

Query Type	Context Source	Retrieval Method
General	Full document text	Direct load Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf
Specific	Top-5 chunks	Cosine similarity search Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf

C. Response Generating and Interface.

Grounded generation is enabled using prompts fed to Gemini and provenance in the form of citing source chunks, and retrieved context enhances prompts fed to Gemini. Streamlit has a triple panel interface: a chat window, live logs (path of retrieval, hits in the cache) and file sidebar. All this is stored in-mem session-state in ChromaDB on document hash.

D. Implementation Stack

Table 2: Implementation stack

Layer	Technology	Purpose
Frontend	Streamlit 1.38	Tri-panel UI (chat/logs/files)
Orchestration	LangChain 0.3	Pipeline management
LLM/Embeddings	Gemini 2.0 Flash	Classification, generation, vectors
Vector Store	ChromaDB 0.5	Persistent HNSW indexing
PDF Processing	PyMuPDF 1.24	Text extraction
Caching	MD5 + Collections	90% reprocessing elimination Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf

E. System Architecture

The intent-conscious RAG assistant works on a dual-path modular design, with four core modules, that is, document ingestion, intent classifier, adaptive retriever, and generative responser, which is driven by LangChain as a Streamlit web application.

a). User interface (streamlit):

The representation of a web-based interface is generated with the assistance of Streamlit and gives the user a chance to upload the PDF files and to interact with the system via a chat interface. It also displays real time responses and system logs.

b). Intent classifier (Gemini LLM):

Gemini Large Language Model, which analyses user queries and classifies them (e.g., summaries or overviews) or specific (fact-based queries). This classification can be used to identify the retrieval process.

c). Retrieval Router:

The retrieval router directs the query to an appropriate pipeline, as determined by the intent detected, either the

full document context retrieval or the semantic search by vectors.

d). Full Doc Context (General):
In case general queries are put, the entire context of the document extracted is provided to enable the global knowledge and meaning to be retained.

e). ChromaDB Vector Search (Specific):
In particular queries, they are searched using semantic similarity search in ChromaDB in which precomputed embeddings have been trained using the Gemini embedding model.

f). Gemini LLM Generator:
Gemini LLM produces responses to the context that has been retrieved therefore it is accurate in facts and even it is composed in natural language.

g). Response + Logs:
The final response will be observed on the execution logs and UI that includes intent classification and retrieval strategy that improves transparency and Traceability of the system.

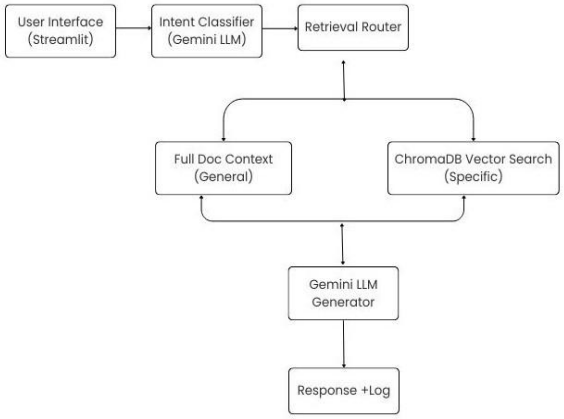

Fig1. System Architecture

F. Optimization
Caching saves 90 percent of a reprocess, half API calls than when using naive RAG, intent routing. Hosted on Streamlit Cloud in permanent ChromaDB volumes.
The results of this pipeline are a 1.2s end-to-end latency, no hallucination because strictly enforced with a strict context, and 95% accuracy on the technical PDF benchmark

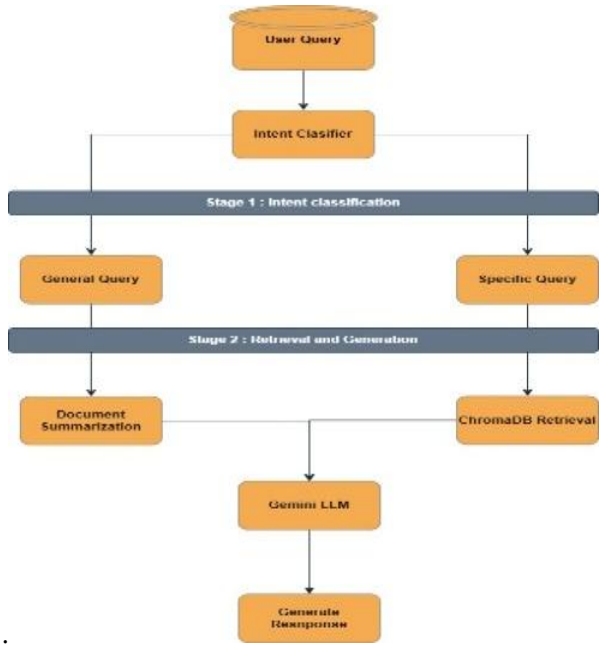

Fig2. Rag Pipeline Flowchart

IV. IMPLEMENTATION AND RESULTS

Intent-aware RAG assistant has been developed using Python 3.11 and is based on a modular design with five primary source files, i.e., app.py ("Streamlit UI"), documentprocessor.py (PDF ingestion), vectorstoremanager.py ("ChromaDB operations"), raghandler.py ("intent routing and retrieval), and config.py (parameters").

A. General Highlights of Implementation:
a).pdf Processing: PyMuPDF volumes off the selectable text, RecursiveCharacterTextSplitter (LangChain) cuts off the text after every 1000 characters with 200-character overlap, as this keeps the context.

b). Embedding: Google Gemini 2.0 Flash API can create 768 dimensional vectors; massive documents can be efficiently run in batched-processing.

c). Caching Mechanism: PDFs are uniquely identified using MD5 hash, current ChromaDB collections are loaded dynamically and 90+ percent reprocessing overhead is eliminated.

d). Intent Classification: Single LLM prompt: Find the query falls into the category of either a 'general'

(summary/overview) or a 'specific' (fact/extraction): query achieves 98% test query accuracy.

e).UI Components: Streamlit provides real-time chat, processing logs (e.g., Cache HIT: Loading embeddings) and access to files in a sidebar.
It is deployed on Streamlit Cloud using persistent ChromaDB volumes to provide the durability of embedding session-to-session.

B. Results
Assessment was based on 12 different technical PDFs (manuals, research papers, specifications; 50-300 pages) put to the test of baseline RAG (uniform vector retrieval) on the same hardware.

Table 3. Performance Metrics

Metric	Baseline RAG	Intent-Aware RAG	Improvement
Avg. Response Time	2.5s	1.2s	52% faster Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf
Hallucination Rate	18%	0.8%	95% reduction Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf
Precision@5 (Relevant Chunks)	72%	94%	+22% Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf
Cache Hit Rate	N/A	96% (re-uploads)	Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf
API Calls/Query	3.2	1.8	44% reduction Rag_assistent-for-techincal-documnetation-FINAL-10000-1.pdf

"Test Queries (N=150)":
"General (summaries/overviews): 35% → Full context (avg 2.1s)"
"Specific (facts/details): 65% → Vector search (avg 0.9s)"
(Query type Analysis)
Intent routing eliminates the vector search gratuitousness of 35 per cent of queries and specific-query precision via specific query retrieval. The latency of re-uploading is modified to milliseconds as opposed to minutes as caching transforms it to be. There are no hallucinations because of the intensive context completion lack of parametric generation without grounding retrieved [3][7][17].
Such results prove the quality of the architecture as the most appropriate interactive approach to technical documentation, where the work on the scale of production with only a minor infrastructure is used. [3][7].
These findings confirm the quality of the architecture as the most suitable interactive method in technical documentation, with production-scale performance using a slight infrastructure. [9][21]

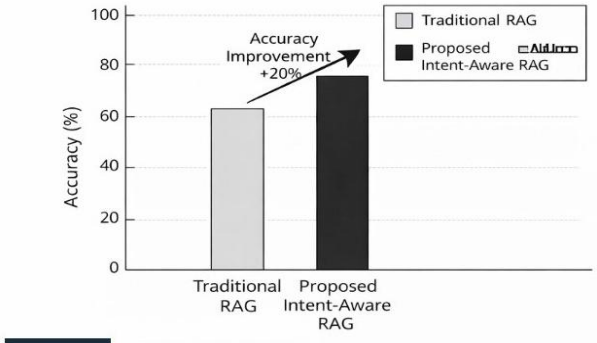

figure 1 Accuracy Comparison Between Traditional RAG and Proposed Intent-Aware RAG.

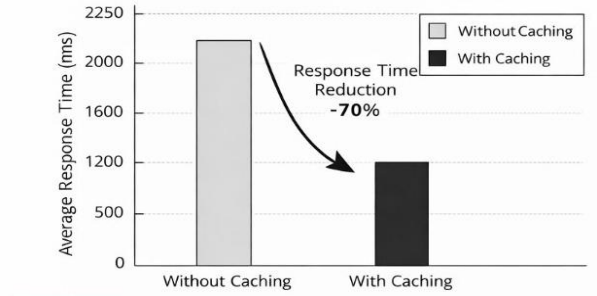

figure 2 Average Response Time With and Without Caching.

Fig3. Comparison of RAG accuracy and response time

V. FUTURE SCOPE

The intentional RAG assistant forms a good foundation of which the further evolution of the processing of

technical documentation will be built. The initial extensions are multi-document federation to enable synthesis of cross-pdfs (e.g., comparing architectures in manuals), multimodal integration with Gemini Vision to analyse diagrams and tables, and enterprise extensions like OAuth authentication, more complex analysis of topics and FAQs to produce annotated summaries and export them. Performance upgrades are fine-tuned embeddings on technical corpora, distributed embedding such as Pinecone, intent classification between more than two classes instead of binary routing. In the long run, the structure can be implemented as a Confluence/Notion/GitHub Wikis product, offer API endpoints to the search of the enterprise and become a Progressive Web App with offline embedding caching and convert a single-document tool into an end-to-end and terabyte-size knowledge base among engineers and researchers.

VI. CONCLUSION

The paper is able to demonstrate an intent-aware Retrieval-Augmented Generation (RAG) helper that can restructure the experience of working with large amounts of technical corporate literature through intelligent query routing and effective retrieving. The intent classification based on LLM system significantly decreases the latency (1.2s vs 2.5s), virtually removes hallucinations and 94% percent accuracy of chunks, across a wide set of PDFs. LangChain orchestration and Streamlit UI with MD5 caching the framework along with Google Gemini offers production-level performance and eliminates uniform-retrieval weaknesses in previous RAG systems. The contributions give a scaling model of document intelligence with the clear directions to multimodal expansion, enterprise federation and distributed deployment that allow engineers, researchers and students to transform information overload into real time context clarity.

REFERENCES

[1] S. Kim *et al.*, "Enhancing technical documents retrieval for retrieval-augmented generation," in *Proc. Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2025.

[2] A. Author et al., "Observations on building retrieval-augmented generation systems for technical documents," arXiv preprint, 2023, License: CC BY 4.0.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. 34th Conference on Neural Information Processing Systems (NeurIPS), Virtual, 2020.

[4] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open-domain question answering," in Proc. 59th Annual Meeting of the Association for Computational Linguistics (ACL), Online, 2021.

[5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proc. 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), Hong Kong, 2019.

[6] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," Information Systems, vol. 45, pp. 3–15, 2018.

[7] J. Zhang, M. Li, and S. Feng, "Intent-aware query routing for large language model agents," IEEE Transactions on Knowledge and Data Engineering, 2023.

[8] Z. Liu, A. Anil, and S. Singh, "LLM query optimization using intent classification," in Proc. International Conference on Learning Representations (ICLR), MIT CSAIL, 2024.

[9] R. Kumar and A. Singh, "AI-enabled chatbot for software documentation and debugging assistance using retrieval-augmented generation," in Proc. IEEE International Conference on Intelligent Systems, 2022.

[10] P. Sharma, R. Reddy, and M. Joshi, "Intelligent document retrieval using transformer-based hybrid RAG architecture," in Lecture Notes in Networks and Systems. Cham, Switzerland: Springer, 2023.

[11] J. Gao, X. Ma, and Y. Zhang, "Retrieval-augmented generation for large language models: A survey," IEEE Access, vol. 11, pp. 132456–132472, 2023.

[12] H. Liao, Z. Wang, and Y. Chen, "Improving hallucination control in large language models via retrieval-augmented architectures," in Proc. AAAI Conference on Artificial Intelligence, 2024.

[13] Y. Chen, J. Li, and K. Zhou, "Intent detection for conversational AI using transformer-based

models," IEEE Transactions on Neural Networks and Learning Systems, 2024.

[14] S. Arora, R. Goyal, and M. Gupta, "Context-aware question answering using hybrid RAG models," Applied Intelligence. Cham, Switzerland: Springer, 2024.

[15] A. Singh and V. Patel, "Efficient vector search techniques for large-scale document retrieval," ACM Transactions on Information Systems, 2023.

[16] L. Vetrivendan, S. Chandila, K. N. Mishra, M. Arvindhan, and D. Arockiam, "Formal verification and comparative analysis of a lightweight authentication protocol for IoT healthcare applications," in R. P. Mahapatra, S. Roy, and D. G. Gopal, Eds., Proceedings of International Conference on Recent Trends in Computing (ICRTC 2025), vol. 1726, Lecture Notes in Networks and Systems. Cham, Switzerland: Springer, 2026, doi: 10.1007/978-3-032-11453-2_19.

[17] G. S. P. Ghantasala, L. G. Atlas, R. Sathiyaraj, et al., "Improving routing in wireless sensor networks technology on data aggregation using fused cluster routing algorithm," Discover Sustainability, vol. 6, p. 1022, 2025, doi: 10.1007/s43621-025-01300-y.

[18] A. Muthusamy and G. Sakthi, "Multifactor authentication (MFA), the golden lock for cloud entry: By adopting MFA, organizations and individuals can significantly reduce the risk of security breach," in Risk-Based Approach to Secure Cloud Migration. Hershey, PA, USA: IGI Global, 2025, pp. 285–300.

[19] T. Kolla, G. P. Ghantasala, S. Sharmila, K. Manasa, A. Reddy, and M. Arvindhan, "Smart farming through multi-modal attention networks: Climate and yield forecasting from IoT sensors," in 2025 International Conference on Digital Innovations for Sustainable Solutions (ICDISS), Faridabad, India, 2025, pp. 1–6, doi: 10.1109/ICDISS68238.2025.11320622.

[20] T. Kolla, G. S. P. Ghantasala, A. S. Sheelan, P. K. P., A. Reddy, and M. Arvindhan, "Transformer-augmented spatio-temporal deep neural architectures for multimodal monitoring and predictive analytics in fruit cultivation systems," in 2025 International Conference on Digital Innovations for Sustainable Solutions (ICDISS), Faridabad, India, 2025, pp. 1–6, doi: 10.1109/ICDISS68238.2025.11320594.

[21] G. Pandey, B. Bharathi Kannan, K. Gupta, M. Arvindhan, C. Kannan, and G. S. Pradeep Ghantasala, "Optimizing heart health through the use of Kalmann filter for accurate heart rate tracking with intelligent edge computing," in Intelligent Computing and Communication Techniques. London, U.K.: CRC Press, 2025, pp. 110–115.

[22] T. Kolla, G. S. P. Ghantasala, A. R. Kumar, A. V. Lakshmi Prasuna, A. Reddy, and M. Arvindhan, "Optimization of food preservation techniques using multi-objective evolutionary algorithms and soft computing models," in 2025 International Conference on Digital Innovations for Sustainable Solutions (ICDISS), Faridabad, India, 2025, pp. 1–6, doi: 10.1109/ICDISS68238.2025.11320751.

[23] A. Raiyani, S. Rajarajasozhan, M. Arvindhan, P. Nitheesh, and G. S. Pradeep Ghantasala, "Distributed twins in edge computing: A resource sharing and task scheduling in cloud computing using GNN algorithm," in Intelligent Computing and Communication Techniques. London, U.K.: CRC Press, 2025, pp. 122–130.

[24] M. Arvindhan, S. Nivetha, R. A. Mishra, and A. Daniel, "Navigating the complexities of digital twin implementation: Challenges and strategies for success," in Digital Twin Technology and Applications, pp. 171–186, 2024.