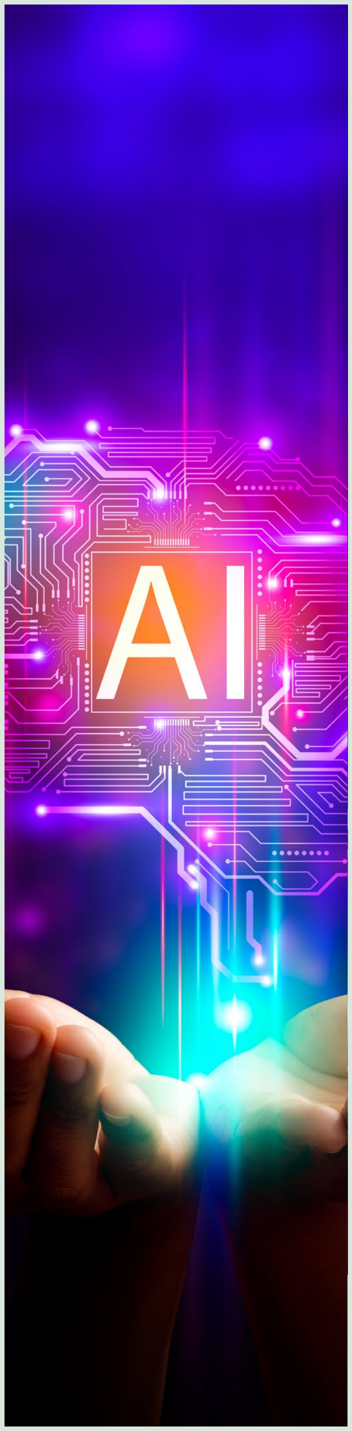
WHITE PAPER



UNLEASHING THE POWER OF RAG: SUPERCHARGE YOUR APPLICATIONS

Abstract

Generative AI models are at the cutting edge of artificial intelligence, transforming its capabilities and unlocking new frontiers. However, these models are not without their limitations, raising concerns about trust and reliability. AI-generated content can sometimes be inaccurate, biased, or even fabricated, leading to skepticism and hesitancy in its adoption. Moreover, issues like hallucinations, where the AI generates plausible but incorrect information, pose significant challenges for real-world and business applications. Additionally, access to specific and current data can be limited, hindering the model's ability to provide up-to-date information.

Generative models are powerful tools that can generate new and creative content. However, they are limited by the data they are trained on and may not be familiar with your organization's specific data, keywords, abbreviations, or access control policies.

Infosys®
Navigate your next

Table of Contents

Introduction to RAG

Retrieval-Augmented Generation (RAG) emerges as a promising solution to address these limitations and foster trust in generative AI models. RAG systems enhance the relevance and accuracy of generative AI responses by incorporating information from external data sources that were not part of the model's initial training. This integration of external knowledge empowers generative AI models to provide more reliable, unbiased, and up-to-date information, effectively addressing the concerns that have hindered their widespread adoption. It's like giving them a big library of information to help them make better decisions. This makes their creations more accurate and trustworthy.

RAG can also help the model understand the organization or domain-specific keywords, abbreviations, and vocabulary and their relation by enabling domain fine-tuning.

- RAG makes AI models more relevant by connecting them to real-time information.
- RAG makes AI models timelier by eliminating the need for frequent retraining.
- RAG makes AI models more accurate by providing them with access to relevant data sources.

Potential Applications of RAG

RAG, a powerful combination of retrieval and generation models, unlocks incredible potential across diverse tasks. It excels in knowledge-intensive areas like question answering and information retrieval, providing accurate and contextually relevant results. For creative endeavors, RAG acts as a digital muse, inspiring writing and data-driven storytelling. But its impact extends beyond static content, powering real-time applications like chatbots, live captioning, and personalized news feeds. This versatile technology is sure to revolutionize how we interact with information in the future.

Constructing an Effective RAG Production Pipeline

We will cover these six aspects for constructing an effective RAG pipeline:

- Hybrid search
- Selection of embedding, fine-tuning, and re-ranking
- Fine-tuning LLM for RAG
- Chunking and retrieval strategies
- Query expansion, routing, and query construction
- Prompt engineering

Hybrid Search

Hybrid search in RAG pipelines bridges the gap between semantic and keyword accuracy. Its alpha parameter balances the influence of each search method.

Embedding-based search isn't ideal for:

- Finding names (e.g., Leonardo Da Vinci, Taj Mahal)
- Understanding abbreviations (e.g., NATO, ASAP)
- Identifying codes (e.g., SKU#12345, flight KL543)

Keyword Search Limitations:

- **Synonyms:** Searching for "happy" might miss articles using synonyms like "joyful" or "elated" because a keyword search just looks for exact matches.
- **Broad Topics:** Looking up "bird" with keyword search could return everything from bird-watching tips to different bird species.

The best approach is to combine embedding-based search with keyword search. Think of it as a navigation system blending a detailed map (semantic) with GPS accuracy (keyword). You reach your destination faster and avoid wandering the wrong streets. By combining semantic retrieval with precise keyword matching, hybrid search offers:

- **Improved recall:** Find relevant documents even if keywords are missing or rephrased.
- **Increased precision:** Filters out irrelevant results that match keywords but lack semantic meaning.

Selection of Embedding, Fine-tuning and Re-ranking

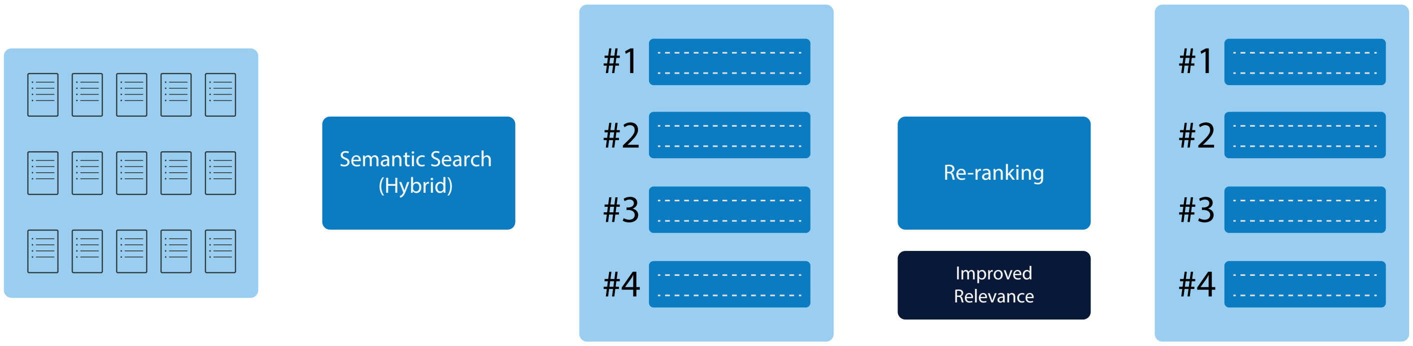
The success of retrieval systems hinges on the effective selection and combination of embedding and re-ranking models. While various architectures exist for both retrieval (dense encoders like **Bi-encoders, Cross-encoders, and COLBERT**) and re-ranking, a crucial challenge lies in striking a balance between retrieval accuracy and computational efficiency.

We recommend exploring the benefits of utilizing compressed embedding representations such as **Binary Embeddings and Int8 Embeddings.** These methods offer significant computational advantages without sacrificing retrieval performance. Additionally, architectures like MRL (Matryoshka) enable flexible embedding dimensionality, further optimizing resource utilization.

“We propose a novel approach for enhancing Information Retrieval (IR) systems by integrating the classical BM25 algorithm with a state-of-the-art retrieval and re-ranking model trained on Colbert (late interaction) using Matryoshka embedding (variable dimensions) - its late interaction mechanism combined with token-level dimension flexibility.

Furthermore, we explore the efficiency of binary or quantized (int8) representations on top of this combined framework to achieve scalability while maintaining performance.”

The secret sauce for top-notch RAG performance? A sprinkle of embedding magic and a dash of re-ranking. For maximal accuracy, integrate hybrid search with fine-tuned embeddings and re-rankers. Optimizing RAG precision demands meticulous attention to **embedding selection and fine-tuning, coupled with powerful re-ranking models.** Choose domain-aligned embeddings, evaluating them on use case-specific datasets. Analyze retrieval errors for targeted customization. Further, fine-tune both embeddings and pre-built re-ranking models using supervised/unsupervised techniques, followed by reiterative evaluation.



Fine-tune LLM for Better RAG Generation

The perfect LLM for your RAG depends on your unique domain and goals. Choose wisely, evaluate rigorously, and fine-tune strategically for accurate and effective generation. Fine-tune your LLM for specific formats and styles of generation. Give your RAG system a technological makeover! **Refine the LLM to nail specific formats and styles for a polished and professional touch.**

Chunking and Retrieval - Best Practices

Large Language Models (LLMs) used in Retrieval-Augmented Generation (RAG) pipelines struggle to process massive documents in one go. That's where chunking comes in, like folding the map into manageable sections.

Six Chunking Strategies for Optimal Retrieval:

- **Fixed size:** Fixed size word or by token size with some sliding window.
- **Content-based:** Leverages document structure - Split by HTML tags, splits by headers, line breaks, or characters.
- **Variable size:** Adapts to content; can append metadata, header, footer, and watermark to maintain context.
- **Multimodal:** Handles different content types like text, tables, and images.
- **Structure aware:** Analyzes document structure (e.g., property graphs) for deeper understanding.
- **Reference chunking:** Summarizes variable-length chunks and creates links to the original text.

Intelligent retrieval in RAG systems refers to the use of advanced retrieval techniques to identify and retrieve relevant documents for a given query. These techniques go beyond traditional keyword matching and statistical measures to incorporate semantic similarity, contextual understanding, and knowledge graphs. The goal of intelligent retrieval in RAG systems is to provide LLMs with a more comprehensive and informative set of context documents, leading to improved text generation performance.

Five Retrieval Strategies for Intelligent Retrieval:

- **Sentence-window retrieval:** Do not just retrieve the relevant sentence, but the window of appropriate sentences before and after the retrieved one.
- **Knowledge Graph-based retrieval:** Knowledge graphs are structured representations of real-world entities and their relationships, providing a rich source of contextual information. By incorporating knowledge graphs into the RAG framework, systems can better understand the context of queries and retrieve documents that are not only semantically similar but also connected through relevant entities and relationships.
- **Recursive retrieval:** with a summary node at the top and smaller chunked segments as child nodes. This structure facilitates efficient retrieval by enabling an initial search on the summary node followed by a refined selection from child nodes based on relevance scores.
- **Structure-aware retrieval:** uses documents as property graphs and retrieves them accordingly.
- **Auto-merging retrieval:** Segment the document into a hierarchical structure of chunks: 2048 words (top-level), 512 words (mid-level), and 128 words (leaf-level) retrieval, start with leaf-level chunks and merge them into their parent chunks if most leaf-level chunks within a parent are selected.

Few other points to consider for chunking and retrieval:

- Recent Nvidia research says chunk size for embedding with 300 words with k=5-10 when combined with 32k LLM performs better than providing full context.

Our analysis demonstrates a positive correlation between chunk size and retrieval accuracy. Notably, chunking along page boundaries emerges as a simple yet highly effective approach. Sentence-window retrieval gives more accurate results.

RAG with more data significantly improves the results of Generative AI applications. Our recommendation says using a chunk size of 1000-2000 with a top k value reranked to 10 with an overlapping window of 150-200 gives the best results.

- **Complete search stacks do better**

Hybrid search with domain fine-tuned re-ranking gives the best search results. It is a promising approach for improving the performance of semantic searches and retrieval augmented generations. This approach has the potential to provide more relevant and accurate results, which can lead to a better user experience.

Identify the good and the bad candidates

Normalize the score from hybrid re-ranking and discard documents below certain thresholds.

Query Expansion, Routing and Query Construction

Effective RAGs rely on query transformations like rewriting, expanding, and compressing user queries to improve information retrieval. Imagine asking, "Product launch guidelines?". An expanded query like "instructions, procedures, regulations for launching a new product?" would yield better results.

Routing and query construction further guide the search. Routing acts like a smart librarian, directing the query to the relevant data source (e.g., database, knowledge graph). Then, query construction translates the user's natural language into a format the chosen data source understands, ensuring accurate retrieval. Both steps are crucial for efficient and accurate information retrieval in RAG systems.

Prompt Engineering

Prompt engineering is considered a critical factor in achieving optimal performance from RAG. Here are a few more features:

- Clear prompts guide retrieval, finding relevant facts and passages.
- Tailored prompts direct generation, shaping style, format, and accuracy.
- Overall, strong prompts boost performance, satisfy users, and unlock new applications.

Example:

Bad Prompt: "What are the best leave policies?"

This prompt is vague and subjective, potentially leading to biased or irrelevant results.

Good Prompt: "Compare the effectiveness of different leave policies (e.g., unlimited leave, fixed-day allowance) in improving employee morale and reducing burnout."

This prompt is specific, neutral, and contextual, leading RAG to retrieve valuable information for informed decision-making.

Remember, crafting the perfect prompt requires skill and adaptation. But with thoughtful design, you can unleash the full power of RAG for accurate and impactful results.

RAG Performance: Navigating the Landscape of Vector Databases for Optimal RAG Performance

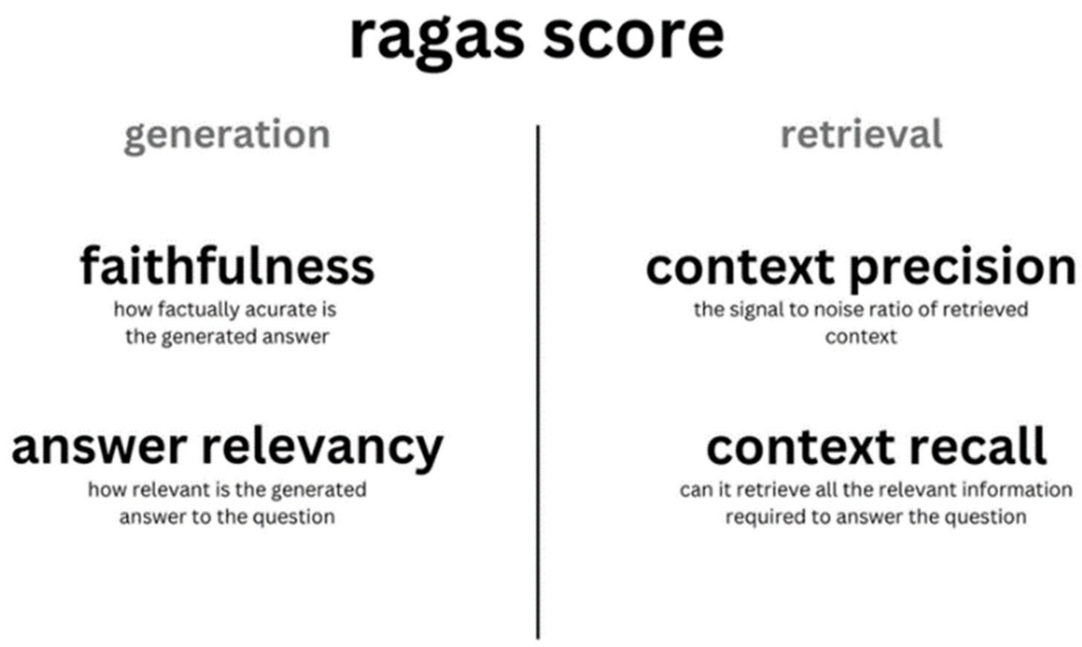
Building production-ready RAG applications requires careful consideration of the underlying Vector Database (VDB). The sheer abundance of options, including PineCone, Quadrant, Weaviate, Redis, pgVector, Vespa, and others, necessitates a nuanced evaluation process. Selecting the optimal VDB hinges on several key factors that directly impact the performance and robustness of your RAG system.

Critical Factors for RAG-oriented VDB Selection:

- **Query Performance:** Consider the VDB's query-per-second throughput to ensure it can handle the expected demand of your RAG application without latency bottlenecks.
- **Metadata Filtering:** The ability to filter based on rich metadata associated with vectors is crucial for efficiently refining retrieved information and tailoring results to specific contexts.
- **Multimodal Embedding and Hybrid Search Support:** If your RAG application utilizes diverse data modalities (text, audio, images), choose a VDB capable of storing and indexing multimodal embeddings effectively.
- **LLM Integration:** Seamless integration with LLM application development frameworks like Langchain and LLamaIndex simplifies deployment and streamlines workflows.
- **Retrieval Optimization:** Advanced features like nearest neighbor search optimizations and k-nearest neighbors' retrieval further enhance the accuracy and efficiency of information retrieval.
- **Licensing Considerations:** Carefully evaluate licensing models and costs to ensure the chosen VDB aligns with your budget and deployment requirements.

RAG Evaluation: Context Relevance, Answer Faithfulness, and Overall Response Quality

RAG evaluation needs to assess both generated text (faithfulness, answer relevance) and retrieved information (NDCG, recall, QPS). Balancing retrieval speed and accuracy is crucial. While zero-shot or few-shot LLMs can evaluate basic metrics, fine-tuning an LLM specifically for RAG evaluation provides the most accurate and nuanced assessment of its performance. This involves tailoring the LLM to generate metrics like context relevance, answer faithfulness, and overall response quality.


ragas score
generation
retrieval
faithfulness
how factually acurate is the generated answer
answer relevancy
how relevant is the generated answer to the question
context precision
the signal to noise ratio of retrieved context
context recall
can it retrieve all the relevant information required to answer the question

"Seven Failure Points When Engineering a Retrieval Augmented Generation System" makes a comprehensive list of places where failure can occur when designing RAG.

- **Missing Content:** For questions related to the content but don't have answers the system could be fooled into giving a response.
- **Missed the Top Ranked Documents:** The answer to the question is in the document but did not rank highly enough to be returned to the user. In practice, the top K documents are returned where K is a value selected based on performance.
- **Not in Context:** Consolidation strategy limitations: Documents with the answer were retrieved from the database but did not make it into the context for generating an answer. It occurs when many documents are returned from the database.
- **Not Extracted:** Here, the answer is present in the context, but the LLM failed to extract the correct answer. Typically, occurs when there is too much noise or contradicting information in the context.
- **Wrong Format:** The question involved extracting information in a certain format, such as a table or list, and the LLM ignored the instruction.
- **Incorrect Specificity:** The answer is returned in the response but is not specific enough or is too specific to address the user's need.
- **Incomplete Answers:** which are not incorrect but miss some of the information even though that information was the context and available for extraction. An example question is, "What are the key points covered in documents A, B, and C?". A better approach is to ask these questions separately.

Given the susceptibility of Retrieval-Augmented Generation (RAG) systems to these seven failure points, thorough robustness testing becomes essential. This testing should evaluate the system's performance across diverse scenarios and question formats.

RAG Guardrails: Responsible by Design

Enabling best practices in RAG guardrails hinges on meticulous data governance, continuous monitoring, user-controlled generation, and embracing the latest research advancements. By nurturing a vigilant approach to RAG's development and deployment, we can unlock its full potential while ensuring responsible and trustworthy information access and generation.

RAG's powerful information retrieval requires robust guardrails. Ensure unbiased data, monitor performance, empower users with guided prompts, and leverage explainable generation techniques like **Chain of Thought (COT), Chain of Verification (COV), React and ACT (REACT)** for a reliable and trustworthy RAG experience.

Conclusion

To build a powerful and ethical RAG model, it's crucial to:

- Optimize: Fine-tune settings, combine search methods, refine data, and craft clear prompts.
- Evaluate: Test your model thoroughly and address any biases.
- Deploy responsibly: Consider ethics, transparency, and user privacy.

By harnessing the potential of RAG with meticulous attention to detail and a commitment to ethical principles, you can unlock a new era of information access, creative expression, and intelligent interaction with the world around you. Remember, the greater the potential, the bigger the responsibility of use. Use RAG thoughtfully and responsibly to pave the way for a future where technology empowers and benefits all.

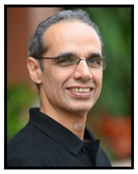
References

- LlamaIndex - Data Framework for LLM Applications
- Seven Failure Points When Engineering a Retrieval Augmented Generation System (arxiv.org)
- LangChain
- A Guide on 12 Tuning Strategies for Production-Ready RAG Applications | by Leonie Monigatti | Towards Data Science
- Introduction | Ragas

About the Authors

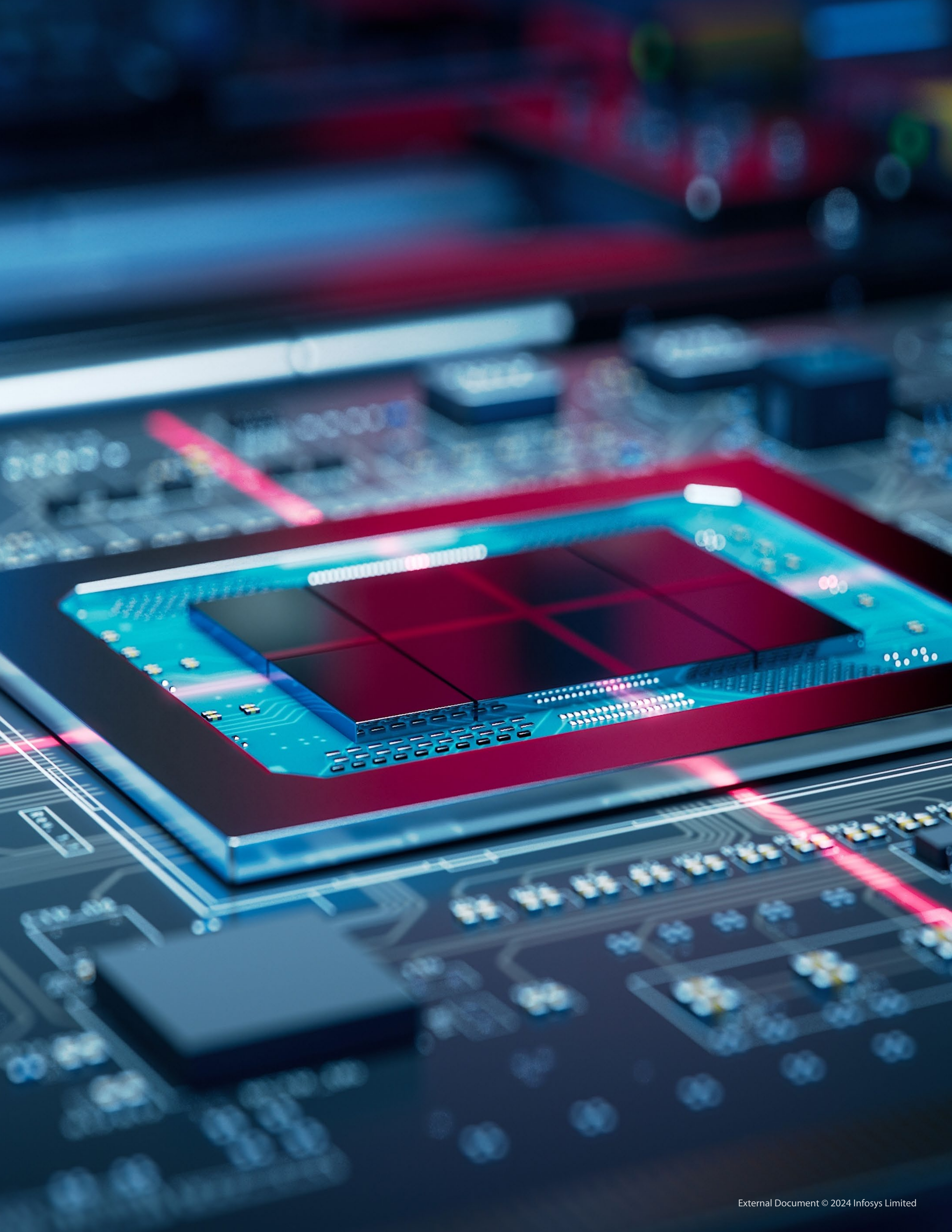
Amit Kumar

Amit is a senior architect with extensive experience in conversational, Generative AI and Classical AI. He has filed 3 patents and is passionate about translating cutting-edge AI research into real-world business solutions. Amit is also an MLOps-certified professional and expert in creating production-ready AI services. His expertise spans a wide range of AI technologies, from Discriminative AI and Classical AI to large language models. Amit has won multiple hackathons and ranked among the top in many Kaggle and Hugging Face challenges.



Vishal Manchanda

Vishal is a Senior Principal Architect with Infosys Center for Emerging Technology Solutions and has over 24 years of experience in the IT Industry. With expertise in Conversational AI, Vishal has been developing, architecting, and incubating various IT solutions/IPs around contact centers, personalized intelligent interfaces involving hyper-contextual personalized videos, voice interfaces, and comprehensive Conversational AI platforms.



For more information, contact askus@infosys.com

Navigate your next

Infosys.com | NYSE: INFY

Stay Connected