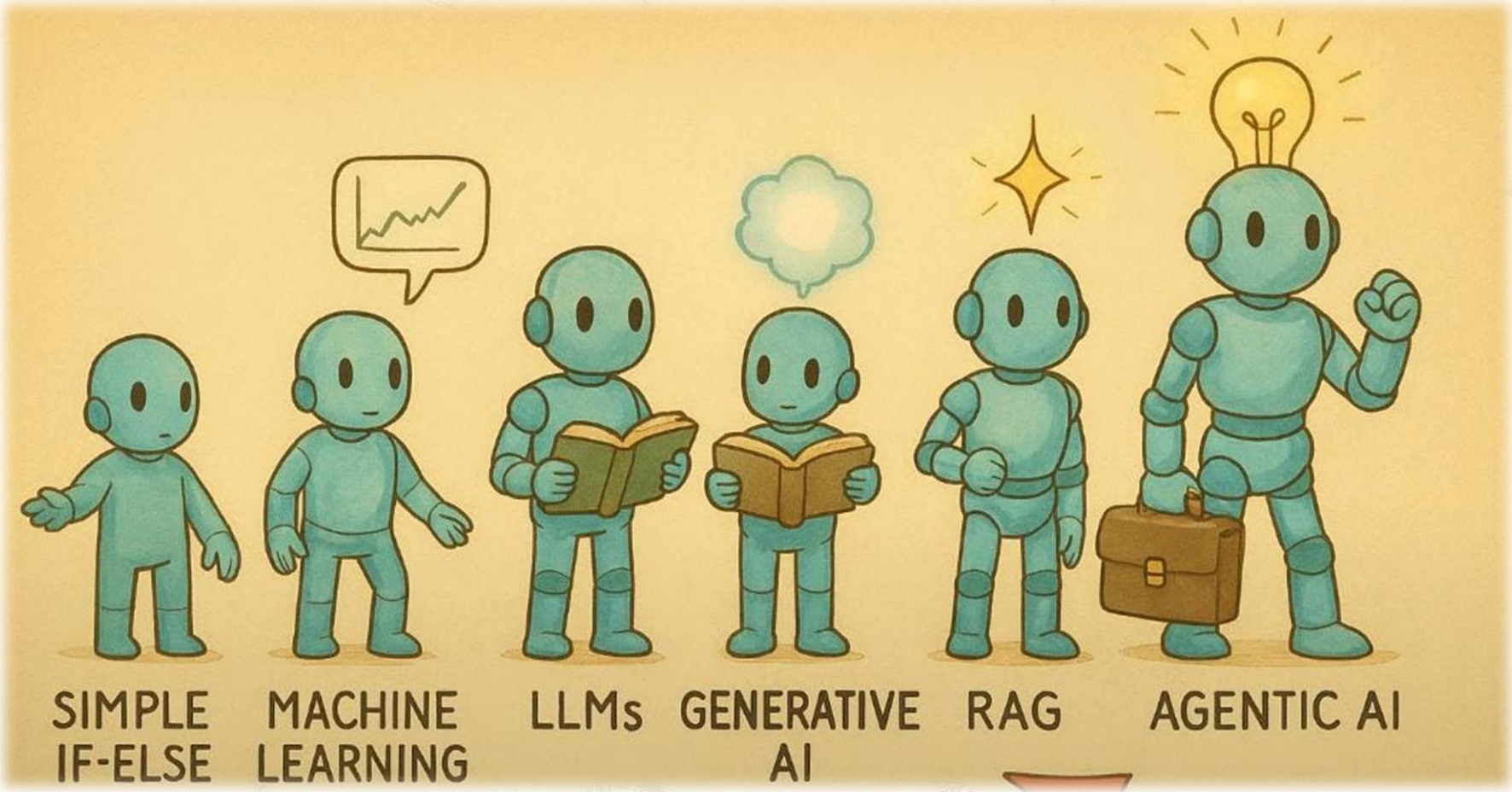
Scope and opportunities in the areas of

Artificial Intelligence - Machine Learning to Agentic AI

Dr. Sarwan Singh

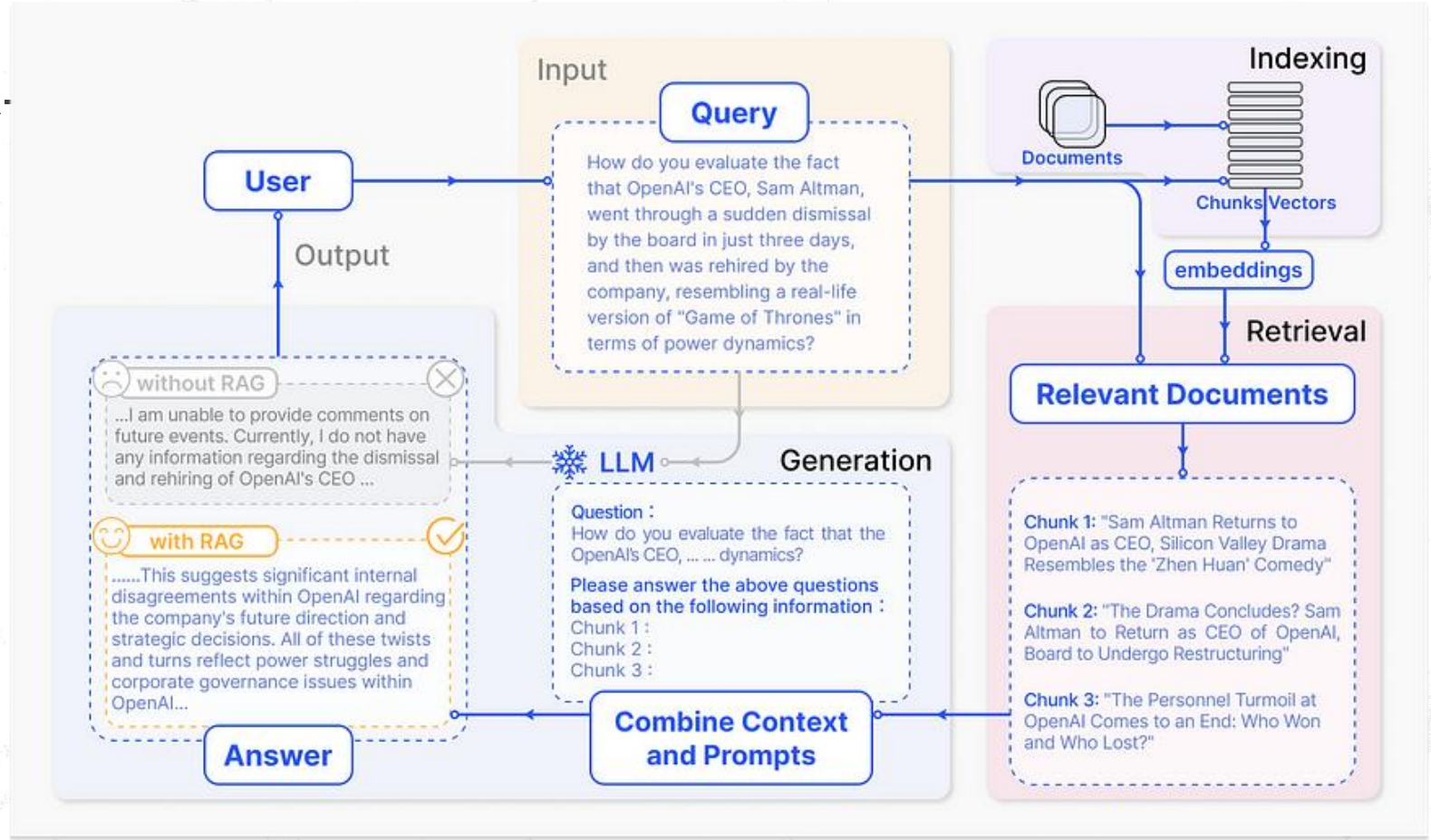
Scientist – D, NIELIT Chandigarh

Journey from Traditional Programming to Agentic AI



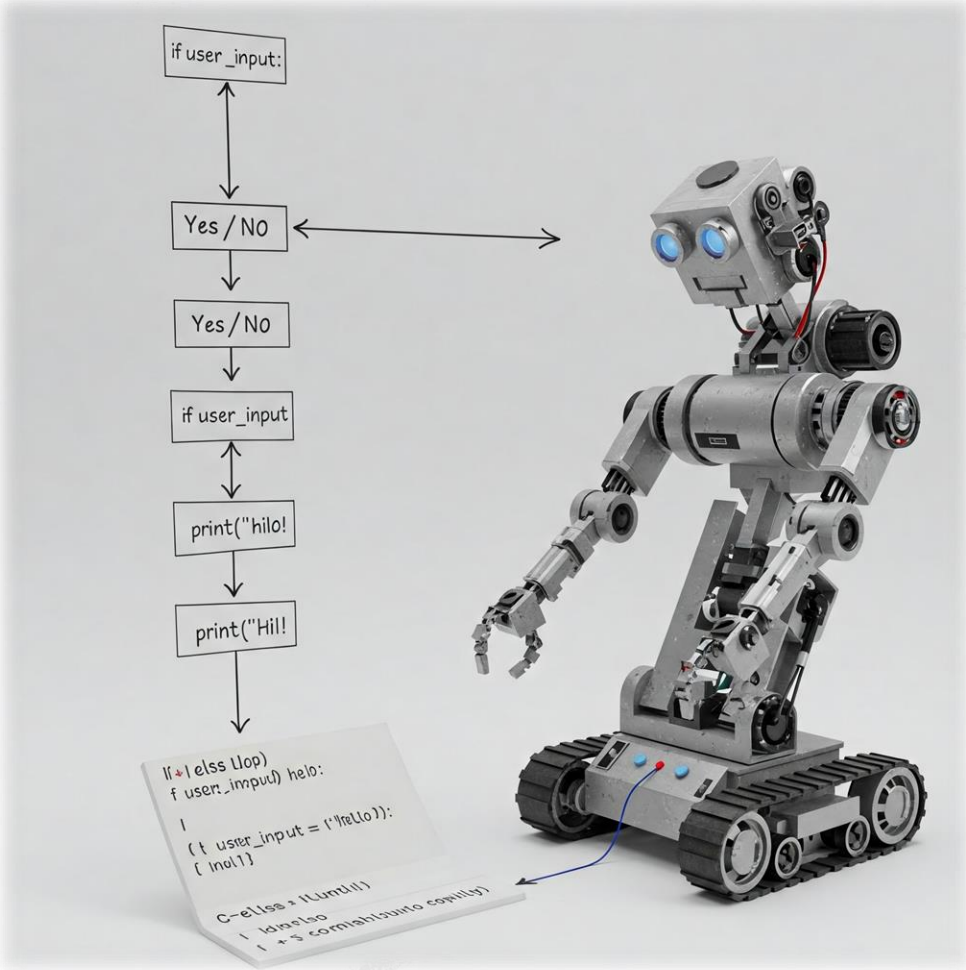
References

- Website : deepgram.com, datacamp , linkedin,
- medium.com/ an-introduction-to-rag-and-simple-complex-rag-9c3aa9bd017b



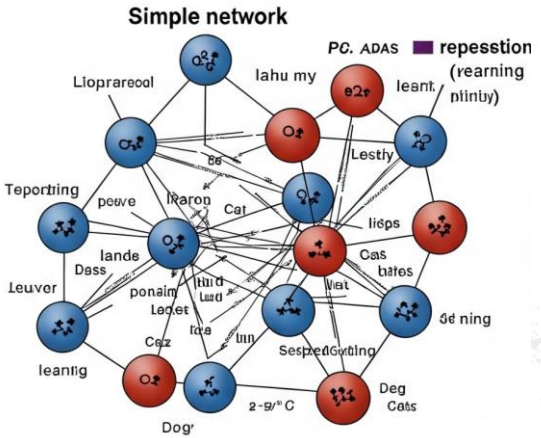
Traditional Programming (Imperative & Rule-Based) (1950s–Present)

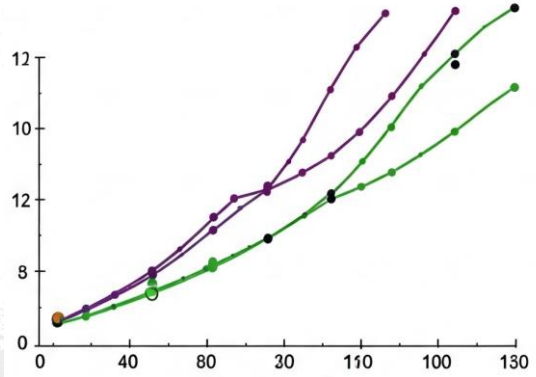
- **Description**: Explicit logic using if-else, loops, and functions. Developers write rules to process inputs and produce outputs.
- **Era**: Dominant since the dawn of computing (e.g., Fortran, C, Python).
- **Key Idea**: "Tell the computer exactly what to do.“



Machine Learning (ML) (1980s–Present)

- **Description**: Systems learn patterns from data instead of relying on hardcoded rules. Includes supervised/unsupervised learning (e.g., regression, decision trees).
- **Key Advance**: Let algorithms *learn* rules from data rather than requiring explicit programming.
- **Milestones**:
  - 1986: Backpropagation for neural networks.
  - 1990s: SVMs, Random Forests.
  - 2010s: Rise of deep learning (CNNs, RNNs).



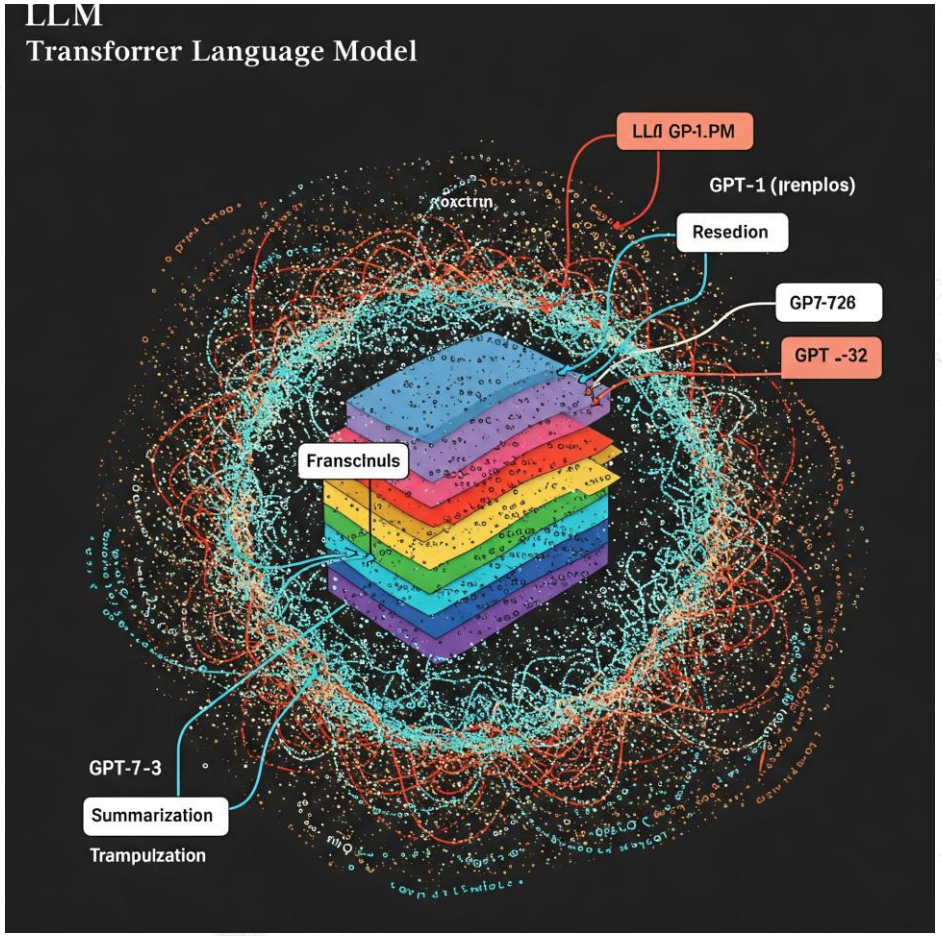


Generative AI (2014–Present)

- **Description**: Models that generate new content (text, images, etc.) by learning data distributions. Early examples: GANs (2014), VAEs.
- **Key Idea**: Create *new* data similar to training data.
- **Applications**: Image synthesis (e.g., StyleGAN), early text generation (e.g., GPT-1).

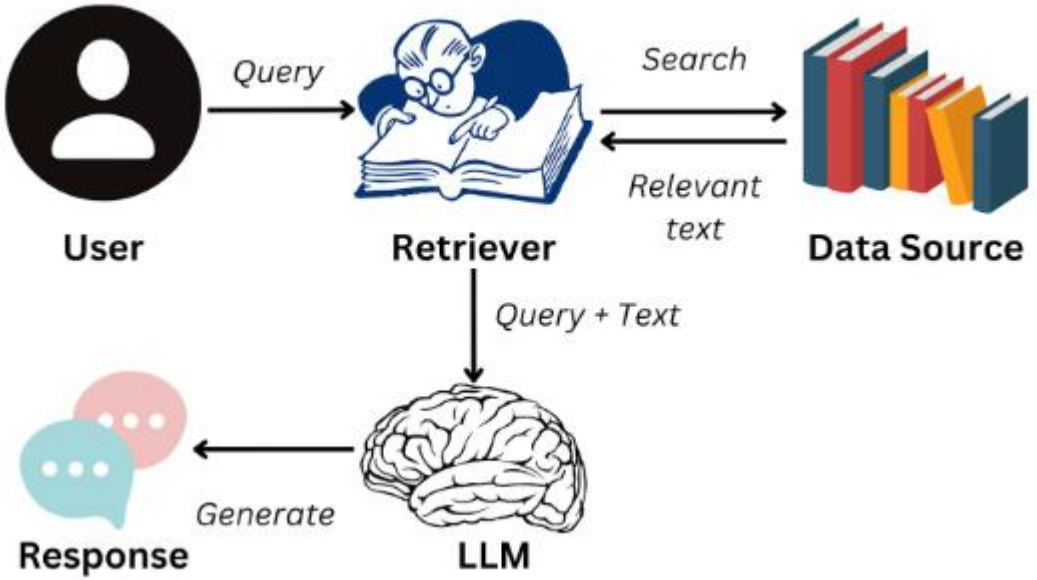
Large Language Models (LLMs) (2018–Present)

- **Description**: Transformer-based models (e.g., GPT, BERT) trained on massive text data for tasks like translation, summarization, and Q&A.
- **Milestones**:
  - 2017: Transformer architecture (Vaswani et al.).
  - 2018: GPT-1, BERT.
  - 2020: GPT-3 (175B parameters).
- **Shift**: Few-shot/zero-shot learning without task-specific training.



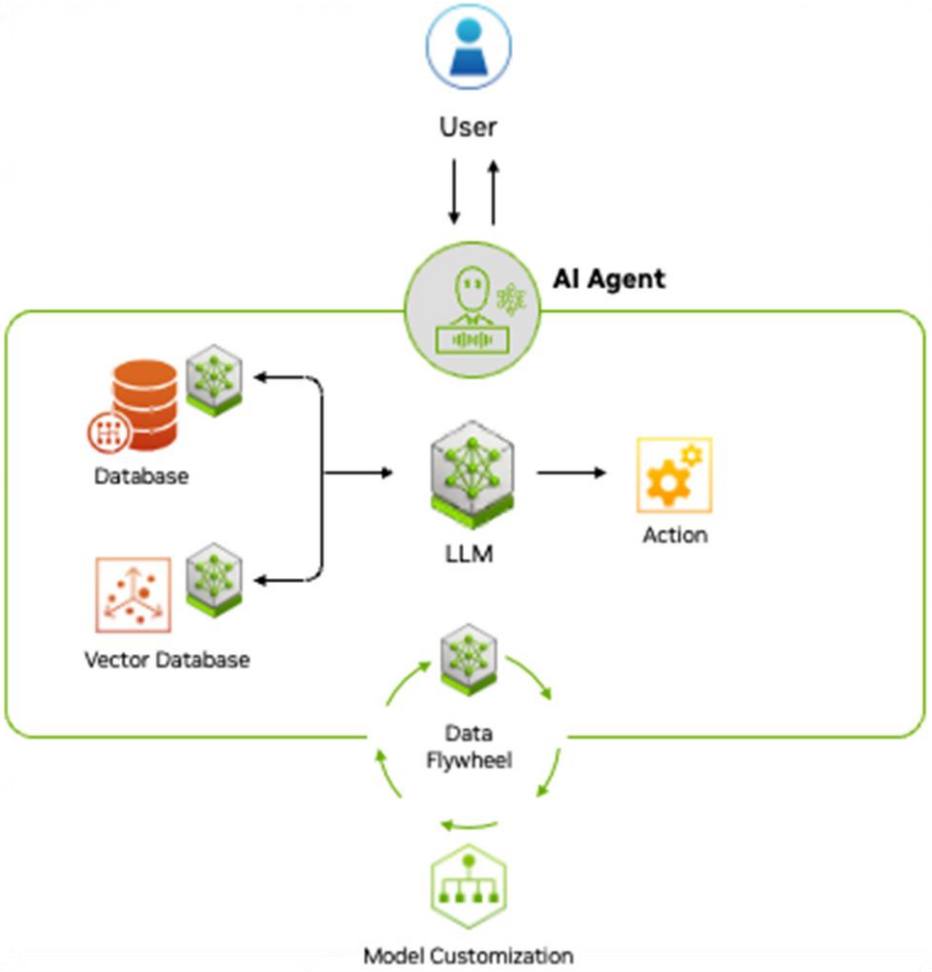
Retrieval-Augmented Generation (RAG) (2020–Present)

- **Description**: Combines LLMs with external knowledge retrieval (e.g., databases, search) to improve accuracy and reduce hallucinations.
- **Key Idea**: Ground LLM outputs in real-time, verifiable data.
- **Example**: ChatGPT + web search for up-to-date answers.



Agentic AI (2023–Present)

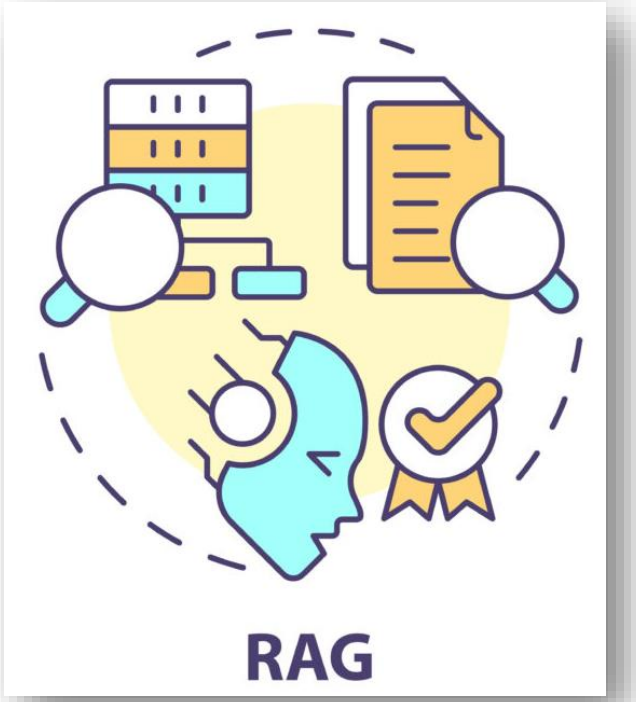
- **Description**: AI "agents" that autonomously plan, act, and iteratively improve (e.g., AutoGPT, BabyAGI). Uses LLMs + tools (APIs, code execution).
- **Key Idea**: *Autonomous* goal-directed behavior with memory/feedback loops.
- **Features**: Self-prompting, multi-step reasoning, tool use.



Timeline Summary:

- **1950s–1980s**: Rule-based programming.
- **1980s–2010s**: Machine learning (statistical models → deep learning).
- **2010s**: Generative AI (GANs, early transformers).
- **2018–2022**: LLMs (GPT, BERT, scaling laws).
- **2020–2023**: RAG (dynamic knowledge integration).
- **2023–Future**: Agentic AI (autonomous systems).
- Each phase builds on prior advances, with LLMs acting as the foundation for modern generative AI, RAG, and agents.
- The trend moves from *explicit programming → learning from data → autonomous reasoning*.

Retrieval-Augmented Generation(RAG)

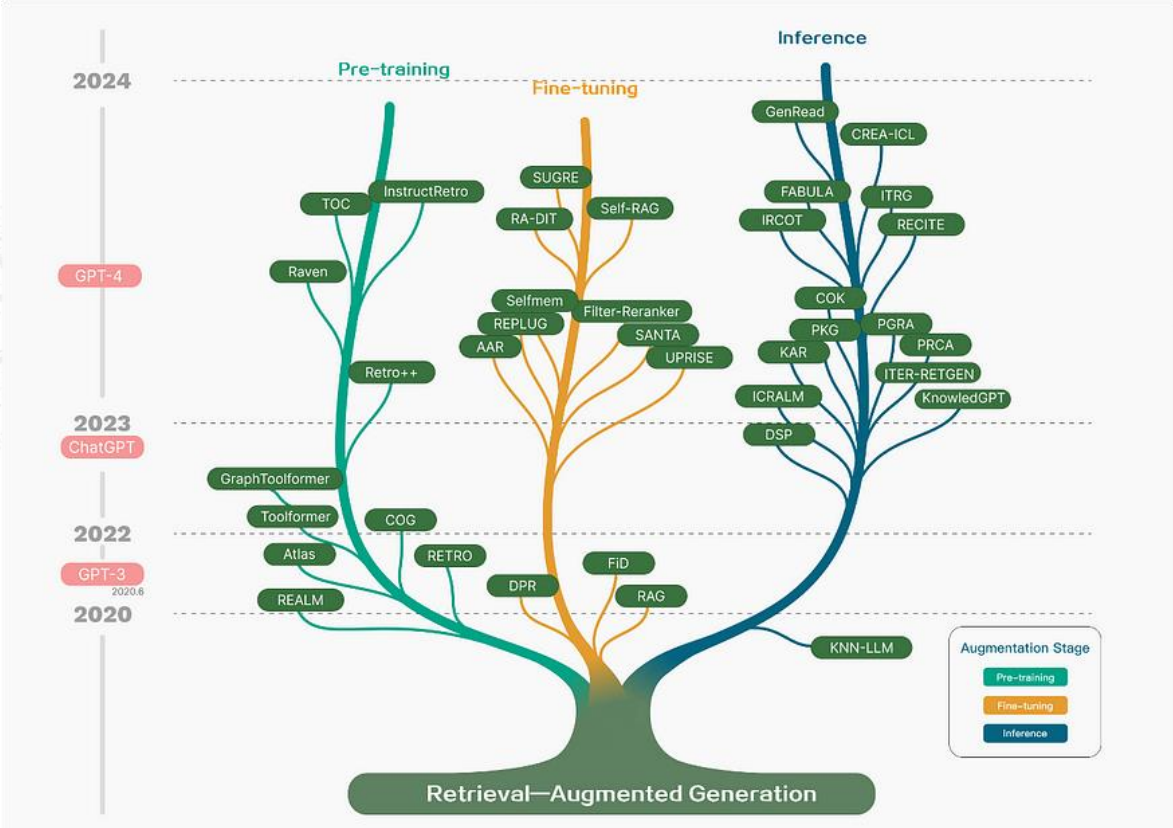


RAG is the biggest business use-case of LLMs

- LARGE LANGUAGE MODELS (LLMS) LIKE GPT-4 HAVE BROUGHT AMAZING PROGRESS, BUT THEY COME WITH **LIMITATIONS—OUTDATED KNOWLEDGE, HALLUCINATIONS, AND GENERIC RESPONSES**.
- THAT'S A PROBLEM WE CAN SOLVE USING **RETRIEVAL AUGMENTED GENERATION (RAG)**.

Agenda - RAG

- Introduction,
- Working
- Limitations, Challenges
- RAG vs Fine Tuning LLMs



Retrieval-Augmented Generation (RAG)

- Retrieval-augmented generation, or RAG, is a framework that allows LLMs to extract information from external knowledge databases.
- We can use it to give current AI models more up-to-date information, rather than having them rely solely on training data gathered before 2021, for example.
- Within the umbrella of Information Retrieval tasks, RAG is poised to be a game-changer.

Retrieval-Augmented Generation (RAG)

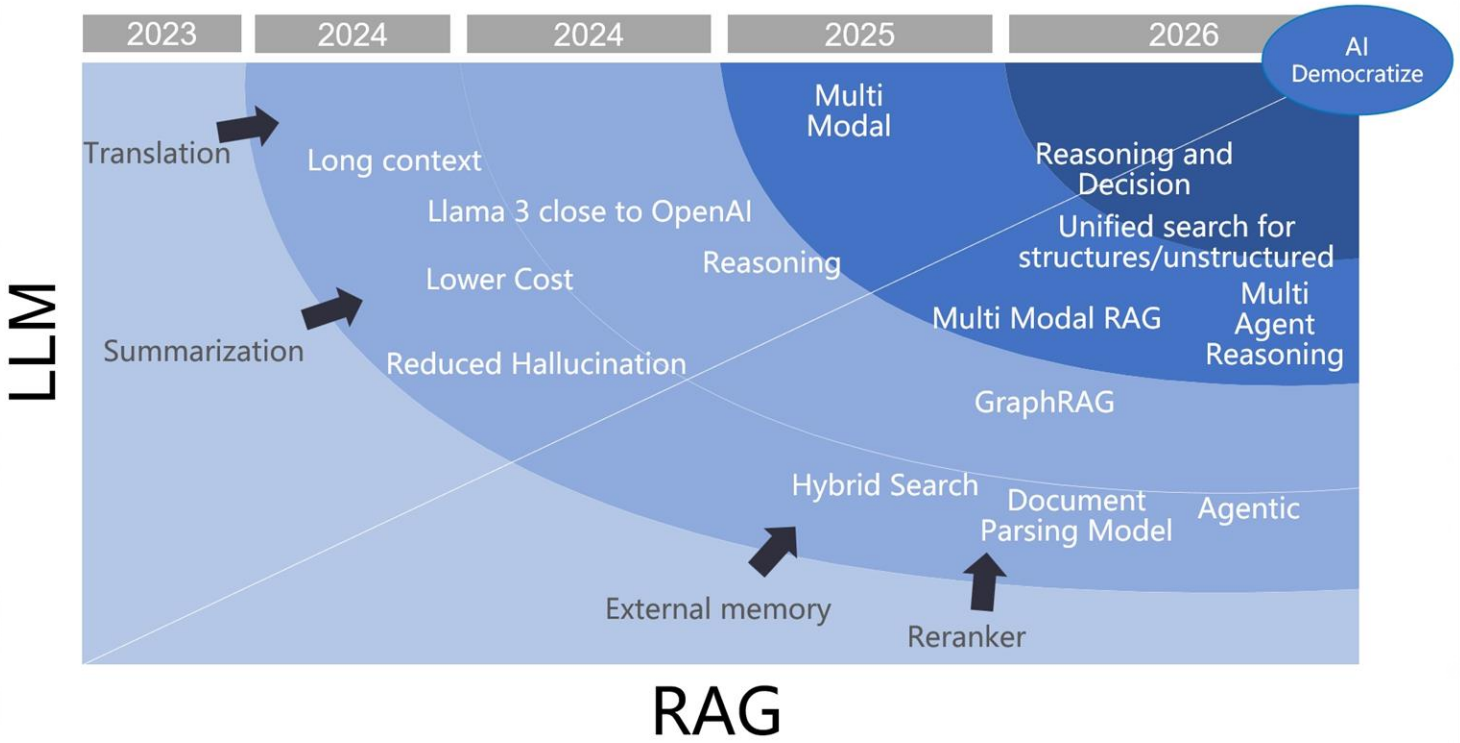
- RAG is a framework for improving model performance by augmenting prompts with relevant data outside the foundational model, grounding LLM responses on real, trustworthy information. Users can easily “drag and drop” their company documents into a vector database, enabling a LLM to answer questions about these documents efficiently.
- In 2023, the use of LLMs saw significant growth in enterprise applications, particularly in the domain of RAG & information retrieval.
- According to the 2023 Retool Report, an impressive 36.2% of enterprise LLM use cases now employ RAG technology. RAG brings the power of LLMs to structured and unstructured data, making enterprise information retrieval more effective and efficient than ever.

Retrieval-Augmented Generation (RAG)

- Retrieval Augmented Generation (RAG) is a technique that enhances LLMs by integrating them with external data sources. By combining the generative capabilities of models like GPT-4 with precise information retrieval mechanisms, RAG enables AI systems to produce more accurate and contextually relevant responses.
- LLMs are powerful but come with inherent limitations:
  - **Limited knowledge**: LLMs can only generate responses based on their training data, which may be outdated or lack domain-specific information.
  - **Hallucinations**: These models sometimes generate plausible-sounding but incorrect information.
  - **Generic responses**: Without access to external sources, LLMs may provide vague or imprecise answers.

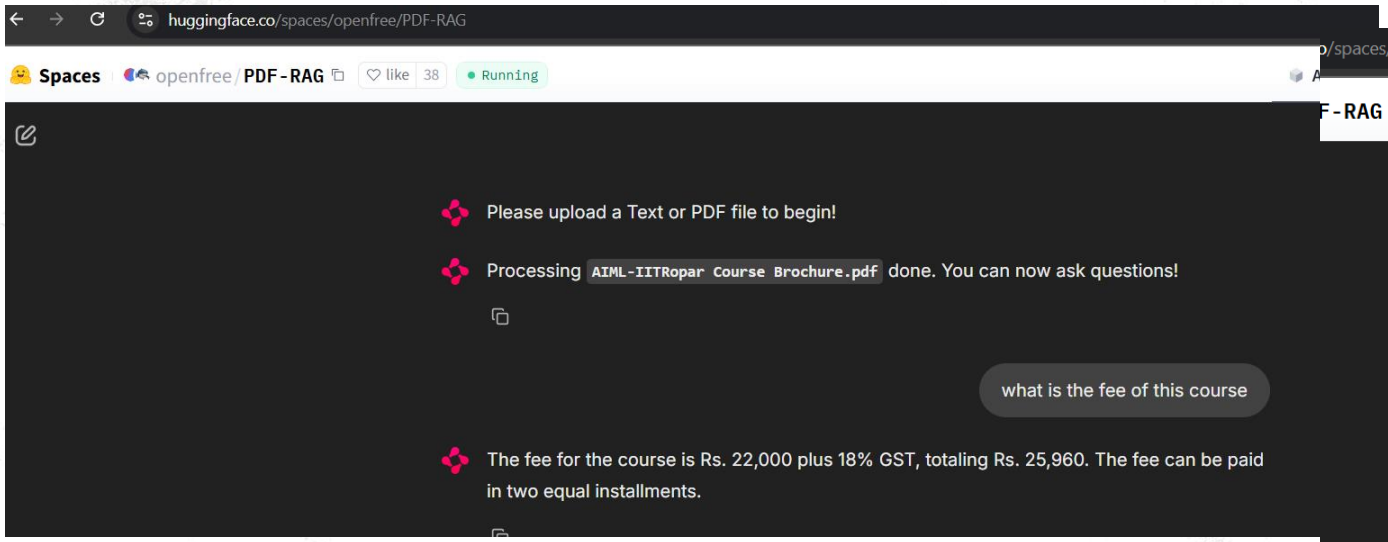
Retrieval-Augmented Generation (RAG)

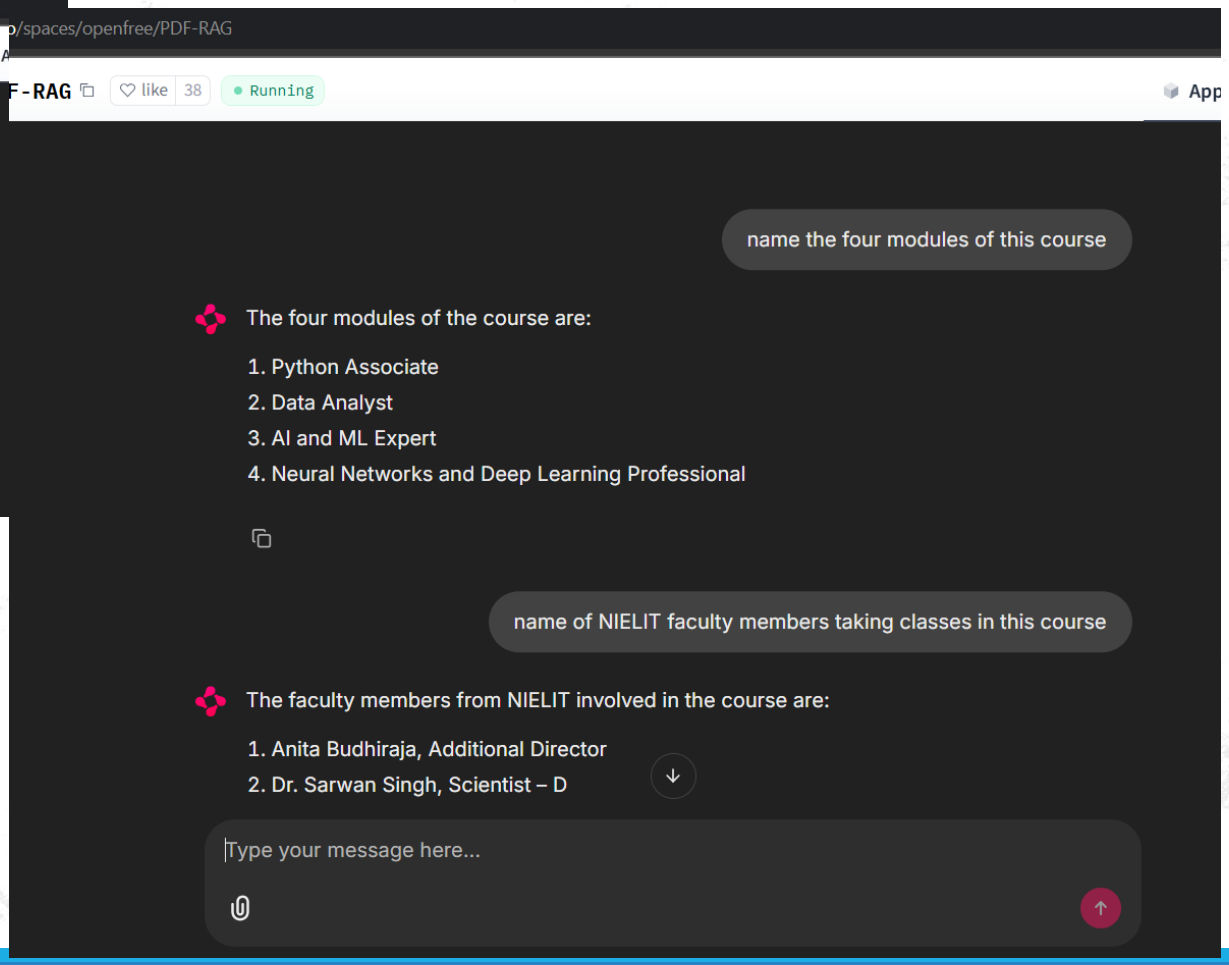
- RAG addresses these issues by allowing models to retrieve up-to-date and domain-specific information from structured and unstructured data sources, such as databases, documentation, and APIs.



Example

- https://huggingface.co/spaces/openfree/PDF-RAG





name of IIT Ropar faculty taking classes in this course

The faculty from IIT Ropar taking classes in this course include:

1. Dr. Puneet Goyal - Associate Professor, Dept. of Computer Science (Image Processing)
2. Dr. Shweta Jain - Assistant Professor, Dept. of Computer Science (Unsupervised Learning)
3. Dr. Shashi Shekhar - Associate Professor, Dept. of Computer Science (Recurrent Neural Networks)
4. Dr. Somdev Kar - Associate Professor, Dept. of Humanities & Social Sciences (Natural Language Processing)
5. Dr. Arun - Associate Professor, Mathematics Dept. (Mathematics and Statistics)

give me the admission link

The admission link is: https://nltchd.info/trgcandregn/studAdmission

Type your message here...

Why Use RAG to Improve LLMs?

- To better demonstrate what RAG is and how the technique works, let's consider a scenario that many businesses today face.
- Imagine you are an executive for an electronics company that sells devices like smartphones and laptops. You want to create a customer support chatbot for your company to answer user queries related to product specifications, troubleshooting, warranty information, and more.

Large Language Models -limitations

Lack of specific information

- Language models are limited to providing generic answers based on their training data. If users were to ask questions specific to the software you sell, or if they have queries on how to perform in-depth troubleshooting, a traditional LLM may not be able to provide accurate answers.
- This is because they haven't been trained on data specific to your organization. Furthermore, the training data of these models have a cutoff date, limiting their ability to provide up-to-date responses.

Hallucinations

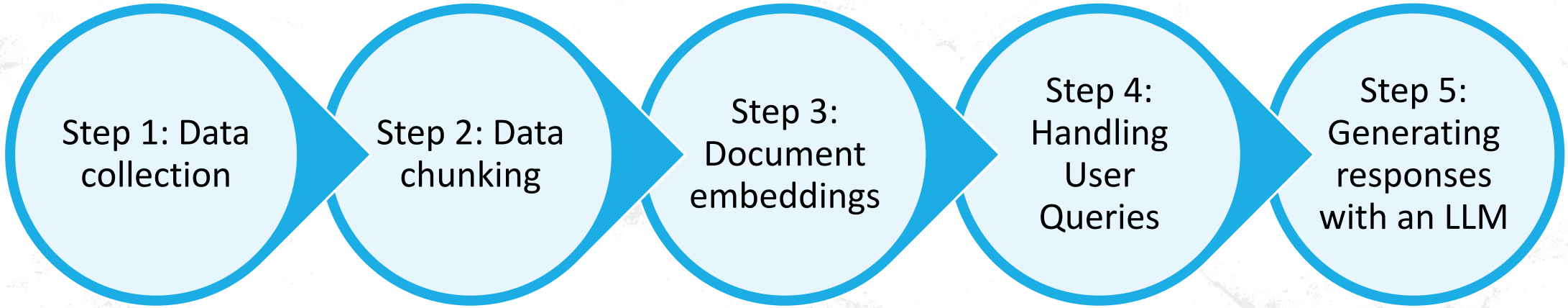
- LLMs can "hallucinate," which means that they tend to confidently generate false responses based on imagined facts. These algorithms can also provide responses that are off-topic if they don't have an accurate answer to the user's query, leading to a bad customer experience.

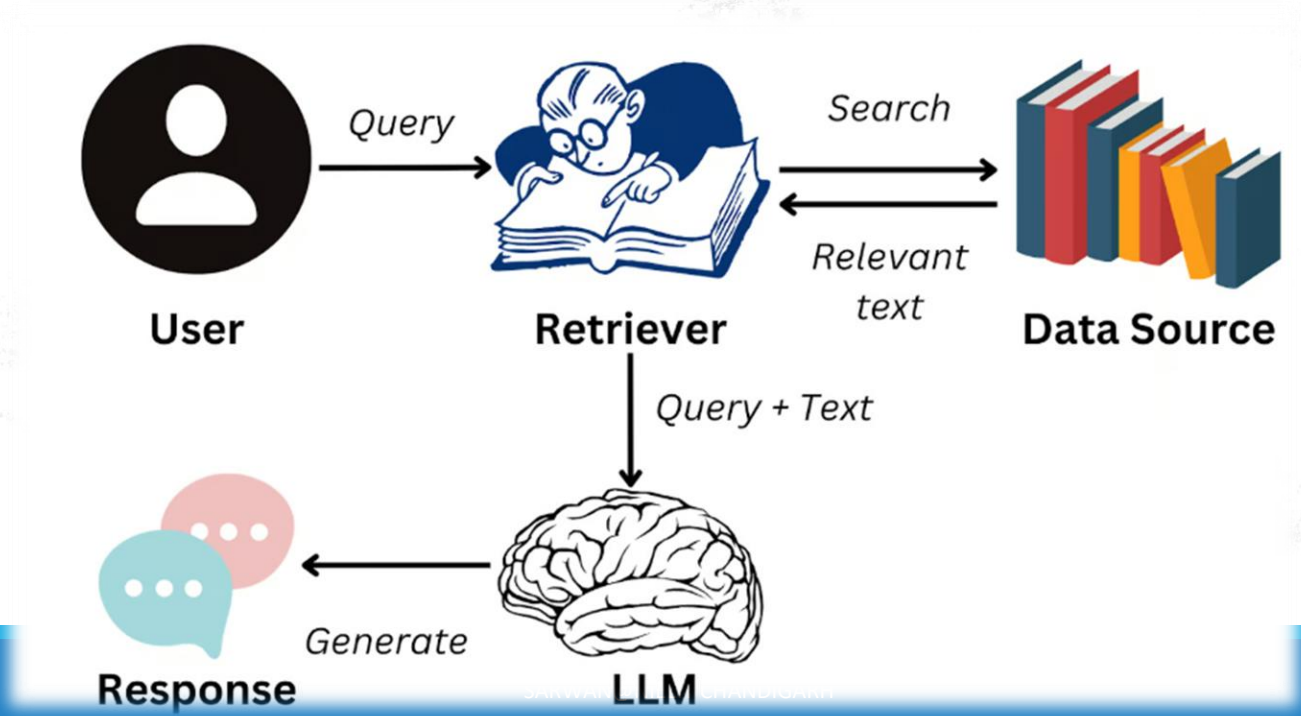
Large Language Models -limitations

Generic responses

- Language models often provide generic responses that aren't tailored to specific contexts. This can be a major drawback in a customer support scenario since individual user preferences are usually required to facilitate a personalized customer experience.
- RAG effectively bridges these gaps by providing you with a way to integrate the general knowledge base of LLMs with the ability to access specific information, such as the data present in your product database and user manuals. This methodology allows for highly accurate and reliable responses that are tailored to your organization's needs.

How does RAG works





How does RAG works

Step 1: Data collection

- You must first gather all the data that is needed for your application. In the case of a customer support chatbot for an electronics company, this can include user manuals, a product database, and a list of FAQs.

Step 2: Data chunking

- Data chunking is the process of breaking your data down into smaller, more manageable pieces. For instance, if you have a lengthy 100-page user manual, you might break it down into different sections, each potentially answering different customer questions.
- This way, each chunk of data is focused on a specific topic. When a piece of information is retrieved from the source dataset, it is more likely to be directly applicable to the user's query, since we avoid including irrelevant information from entire documents.
- This also improves efficiency, since the system can quickly obtain the most relevant pieces of information instead of processing entire documents.

How does RAG works

Step 3: Document embeddings

- Now that the source data has been broken down into smaller parts, it needs to be converted into a vector representation. This involves transforming text data into embeddings, which are numeric representations that capture the semantic meaning behind text.
- In simple words, document embeddings allow the system to understand user queries and match them with relevant information in the source dataset based on the meaning of the text, instead of a simple word-to-word comparison. This method ensures that the responses are relevant and aligned with the user's query.

How does RAG works

Step 4: Handling user queries

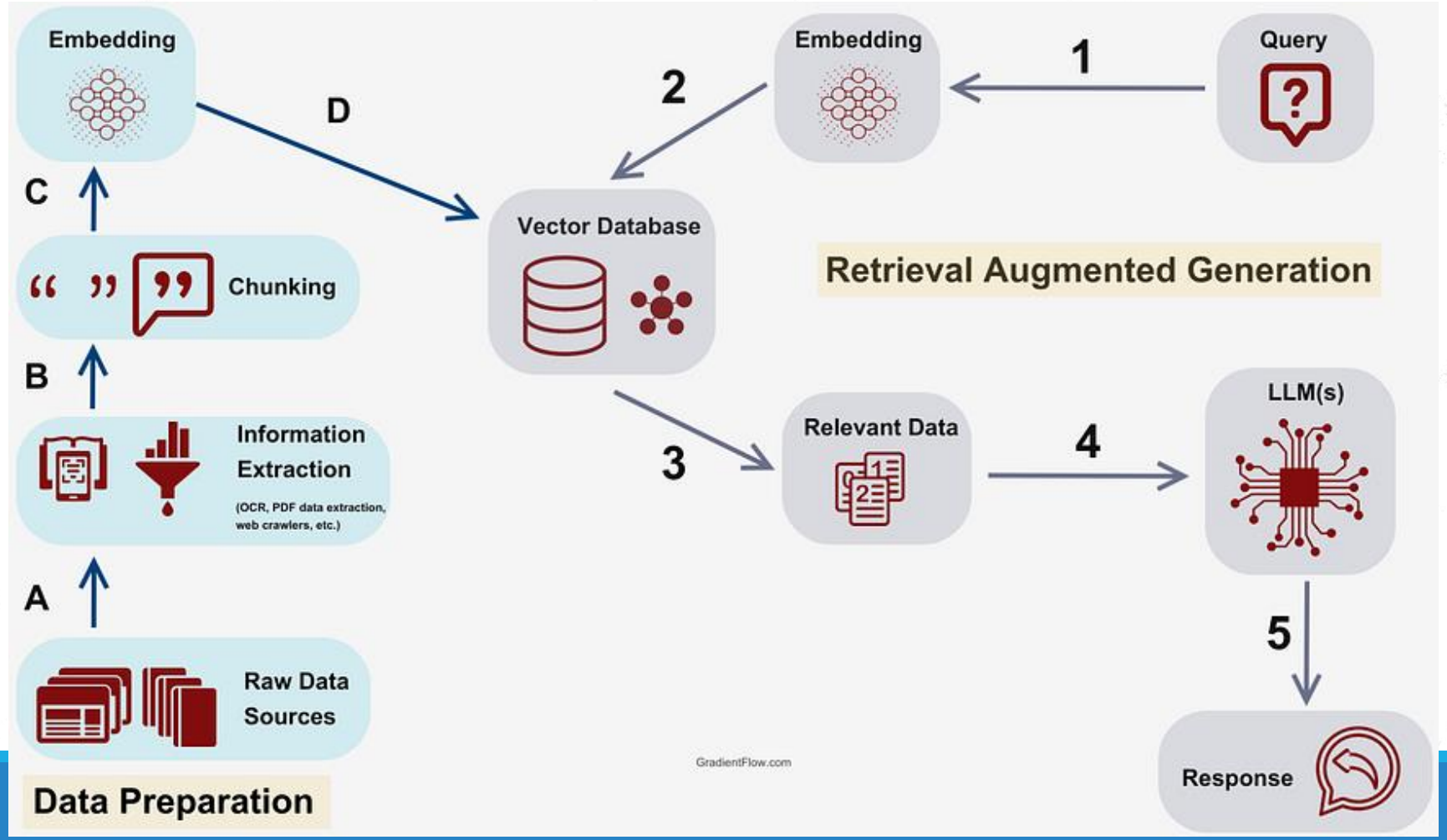
- When a user query enters the system, it must also be converted into an embedding or vector representation. The same model must be used for both the document and query embedding to ensure uniformity between the two.
- Once the query is converted into an embedding, the system compares the query embedding with the document embeddings. It identifies and retrieves chunks whose embeddings are most similar to the query embedding, using measures such as cosine similarity and Euclidean distance.
- These chunks are considered to be the most relevant to the user's query.

Step 5: Generating responses with an LLM

- The retrieved text chunks, along with the initial user query, are fed into a language model. The algorithm will use this information to generate a coherent response to the user's questions through a chat interface.

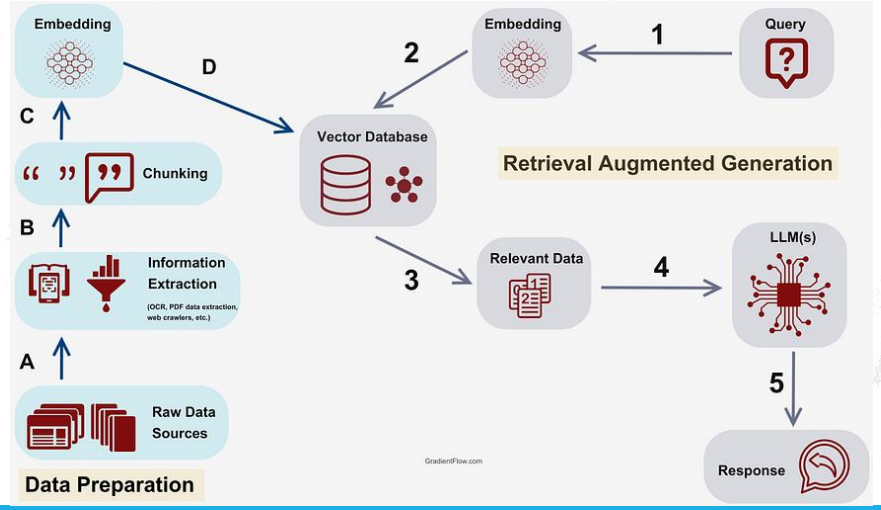
How does RAG works

- A typical RAG process, has an LLM, a collection of enterprise documents, and supporting infrastructure to improve information retrieval and answer construction.



How does RAG works

- The RAG pipeline looks at the database for concepts and data that seem similar to the question being asked, extracts the data from a vector database and reformulates the data into an answer that is tailored to the question asked.
- This makes RAG a powerful tool for companies looking to harness their existing data repositories for enhanced decision-making and information access.



How does RAG works

- RAG is a relevant solution across a wide variety of industries and use cases. In the legal and healthcare sectors, it aids in referencing precise information from vast databases of case law, research papers, and clinical guidelines, facilitating informed decision-making.
- In customer service, RAG is used to power sophisticated chatbots and virtual assistants, providing accurate and contextually relevant responses to user queries.
- RAG is also pivotal in content creation and recommendation systems, where it helps in generating personalized content and recommendations by understanding user preferences and historical data.

Practical applications of RAG

Text Summarization

- RAG can use content from external sources to produce accurate summaries, resulting in considerable time savings.
- For instance, managers and high-level executives are busy people who don’t have the time to sift through extensive reports.

Personalized recommendations

- RAG systems can be used to analyze customer data, such as past purchases and reviews, to generate product recommendations.
- This will increase the user’s overall experience and ultimately generate more revenue for the organization.
- For example, RAG applications can be used to recommend better movies

Practical applications of RAG

Business intelligence

- Organizations typically make business decisions by keeping an eye on competitor behavior and analyzing market trends. This is done by meticulously analyzing data that is present in business reports, financial statements, and market research documents.
- With an RAG application, organizations no longer have to manually analyze and identify trends in these documents. Instead, an LLM can be employed to efficiently derive meaningful insight and improve the market research process.

Challenges and Best Practices of Implementing RAG Systems

Integration complexity

- It can be difficult to integrate a retrieval system with an LLM. This complexity increases when there are multiple sources of external data in varying formats. Data that is fed into an RAG system must be consistent, and the embeddings generated need to be uniform across all data sources.

Scalability

- As the amount of data increases, it gets more challenging to maintain the efficiency of the RAG system. Many complex operations need to be performed - such as generating embeddings, comparing the meaning between different pieces of text, and retrieving data in real-time.

Challenges and Best Practices of Implementing RAG Systems

Data quality

- The effectiveness of an RAG system depends heavily on the quality of data being fed into it. If the source content accessed by the application is poor, the responses generated will be inaccurate.
- Organizations must invest in a diligent content curation and fine-tuning process. It is necessary to refine data sources to enhance their quality.

RAG vs Fine-Tuning

- The two competing solutions for “talking to your data” with LLMs are RAG and fine-tuning a LLM model. You may typically want to employ a mixture of both, though there may be a resource trade-off consideration.
- The main difference is in where and how company data is stored and used. When you fine-tune a model, you re-train a pre-existing black-box LLM using your company data and tweak model configuration to meet the needs of your use case.
- RAG, on the other hand, retrieves data from externally-stored company documents and supplies it to the black-box LLM to guide response generation. Fine-tuning is a lengthy, costly process, and it is not a good solution for working with company documents/facts that frequently change.
- However, fine-tuned models are very good at recognizing and responding to subtle nuances in tone and content generation (Think of the ‘Speak like Abraham Lincoln’ or ‘Write in my writing style’-type features).

RAG vs Fine-Tuning

- Fine-tuning is better suited for scenarios with a narrow, well-defined, static domain of knowledge and format. As Anyscale notes, “fine-tuning is for form, not facts.” For instance, in branding or creative writing applications, where the style and tone need to align with specific guidelines or a unique voice, fine-tuned models ensure the output matches the desired linguistic style or thematic consistency.
- Practically, RAG is likely preferable in environments like legal, customer service, and financial services where the ability to dynamically pull vast amounts of up-to-date data enables the most accurate and comprehensive responses.
- Additionally, RAG excels in customer service applications, where it can swiftly provide current product information or company policies by retrieving specific data from an extensive knowledge base.

RAG vs Fine-Tuning

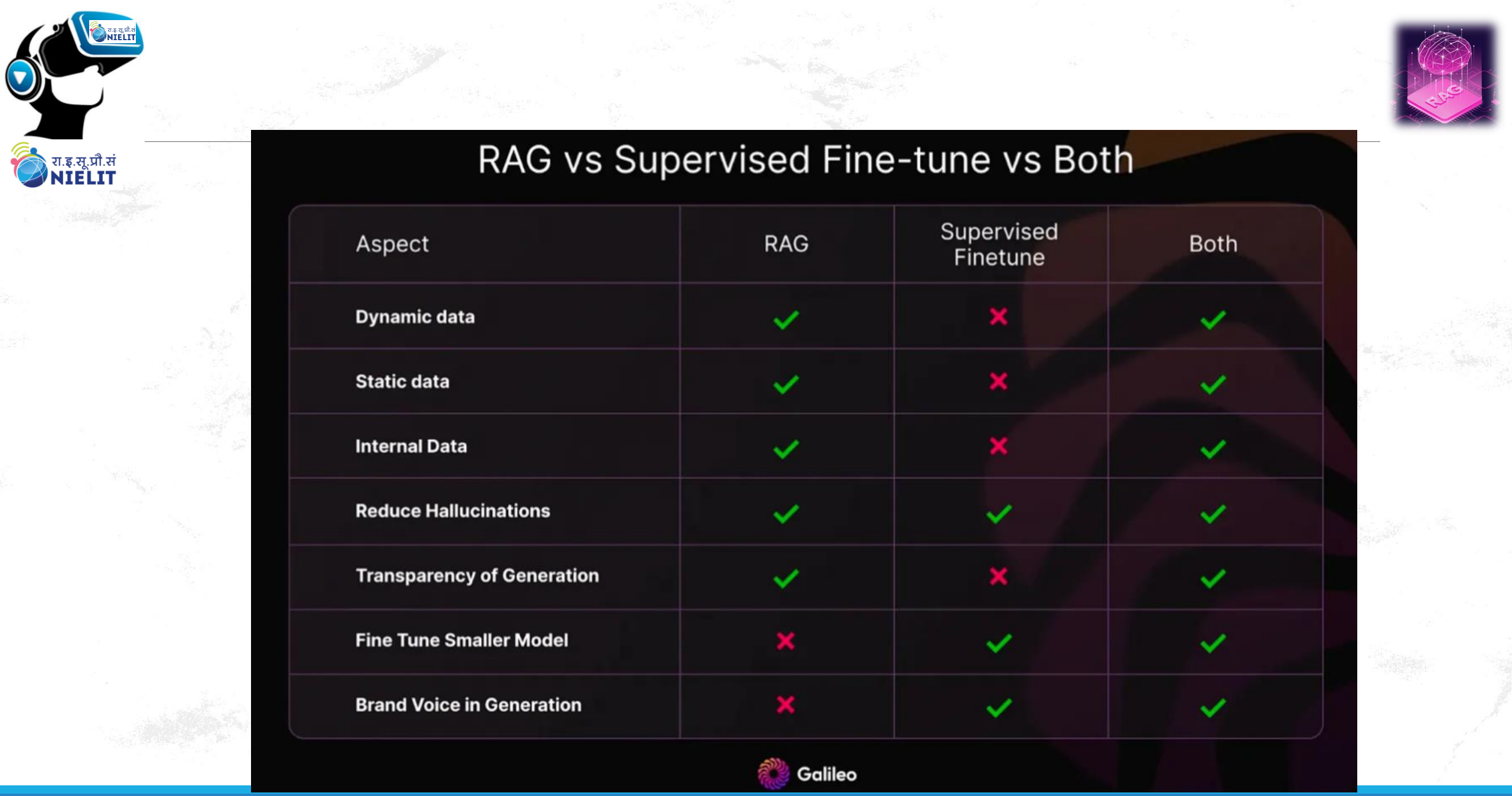
- Anecdotally, enterprises are most excited to use RAG systems to demystify their messy, unstructured internal documents.
- The main unlock with LLM technology has been the ability to handle large corpus of messy unstructured internal documents (a likely representation of the large majority of companies with messy internal drives, etc), which has traditionally led employees to ask for information from other humans rather than trying to navigate poorly-maintained document file storage systems.
- This is unlike in customer service chatbots where customers have been accustomed to either call-centers or simple FAQs to find information.

RAG vs Fine-Tuning

- Feature : Advantages of using RAG
- Ease of introducing new data to the RAG system : RAG accesses current active data stores, providing current and comprehensive answers, surpassing the static knowledge base of a fine-tuned model. RAG also increases the utility of your existing structured and unstructured data-stores (i.e. SQL databases for quantitative data, vector databases for documents, etc).
- Reduced Need for Extensive Training Data : Leveraging external databases, RAG bypasses the need for training data or modifying data pipelines to update a model. RAG is flexible enough to adapt easily to varied or rare topics.
- Minimize Hallucinations : RAG allows LLMs to draw upon external knowledge sources to supplement their internal representation of information, reducing the risk of hallucinations caused by models generating factually inaccurate responses.

practical considerations

- Aspect : Description
- Limitations in RAG for Access-Controlled Data: If your RAG system relies on sensitive information, data privacy is a concern. Scalably and flexibly applying fine-grained access control can be challenging in RAG.
- Limitations in Fine-Tuning to remove data : Teaching fine-tuned models to disregard certain data is challenging, while RAG can simply avoid retrieving irrelevant information.
- Limitations in Fine-Tuning in introducing data that contradicts existing data in the model : Implementing new, contradictory information in fine-tuned models is difficult due to the influence of the base model's training, which may not align with a company's current perspective or worldview.


RAG vs Supervised Fine-tune vs Both
Aspect
RAG
Supervised Finetune
Both
Dynamic data
Static data
Internal Data
Reduce Hallucinations
Transparency of Generation
Fine Tune Smaller Model
Brand Voice in Generation
Galileo