RAG in Action

Building Intelligent Document-Based QA Systems

Kian Kashfipour
MSc Student, Politecnico di Milano

January 2025

About me

- Computer science MSc student at Politecnico di Milano
- BSc in Computer Engineering, Amirkabir University of Technology (2023).
- Data Scientist at TAPSI (2021-2023)
- ML Engineer (RAG systems and LLMs)
- Research Interests: Deep Learning, Generative Models, Large Language Models

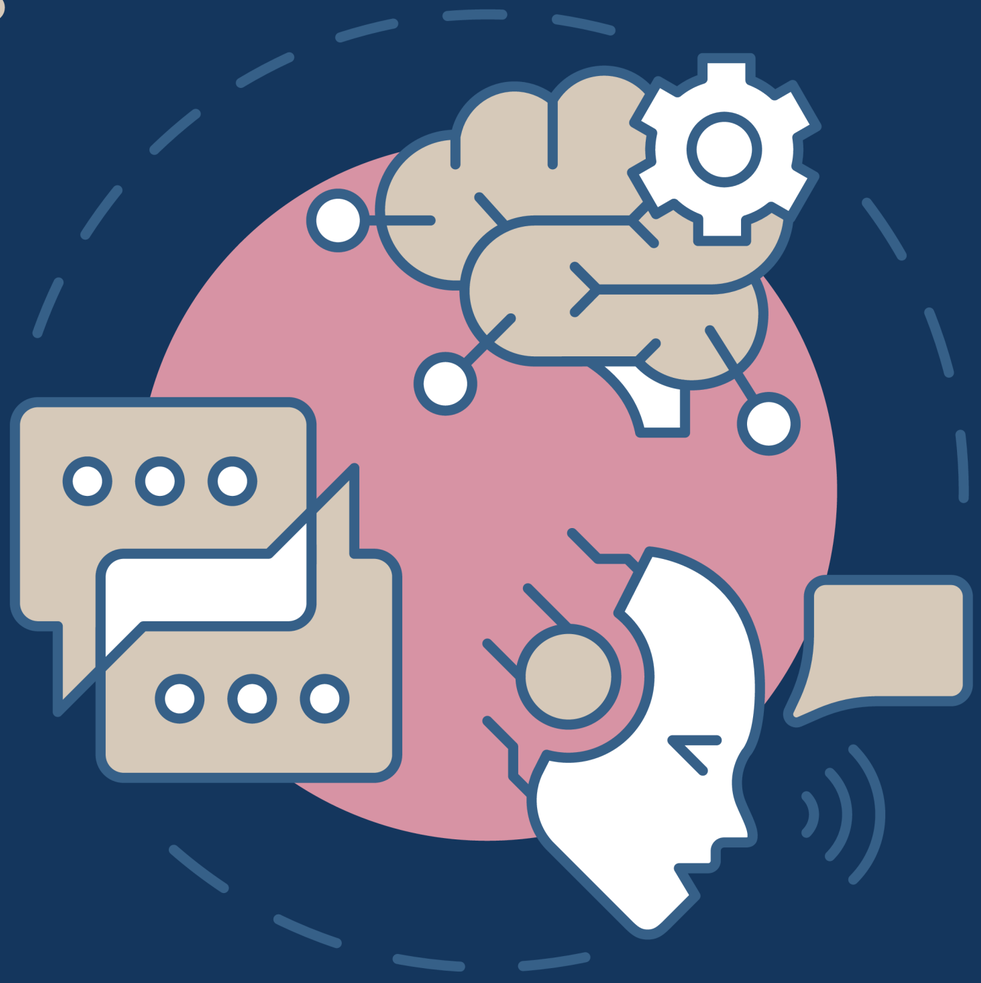
Overview

- LLMs
- Text Generation before LLMs
- Transformers
- Prompt Engineering
- Fine-Tuning
- Retrieval-Augmented Generation

Large Language Models

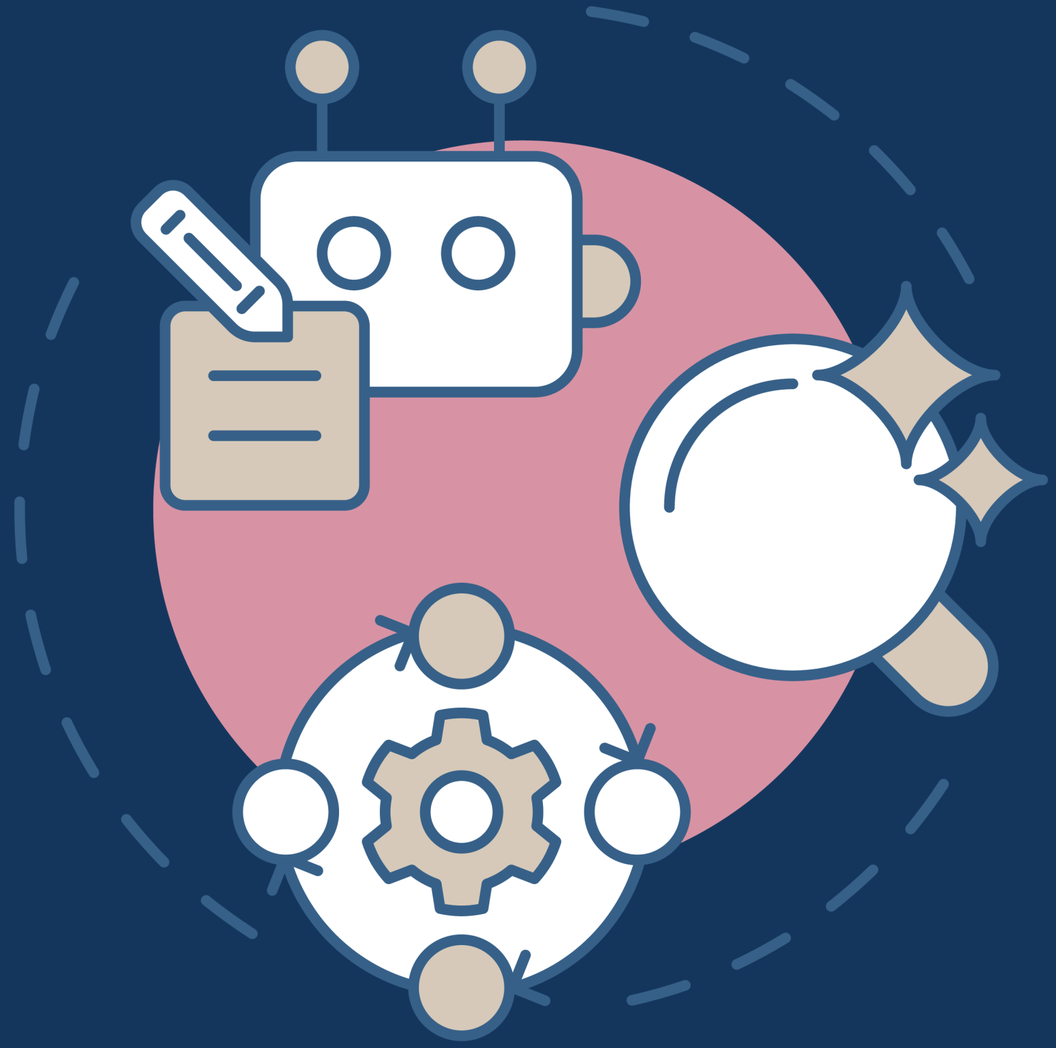
- **Definition:**
  - Type of generative AI
  - Trained on massive data
  - Transformer-based NN

- **Examples:** GPT, BERT, T5

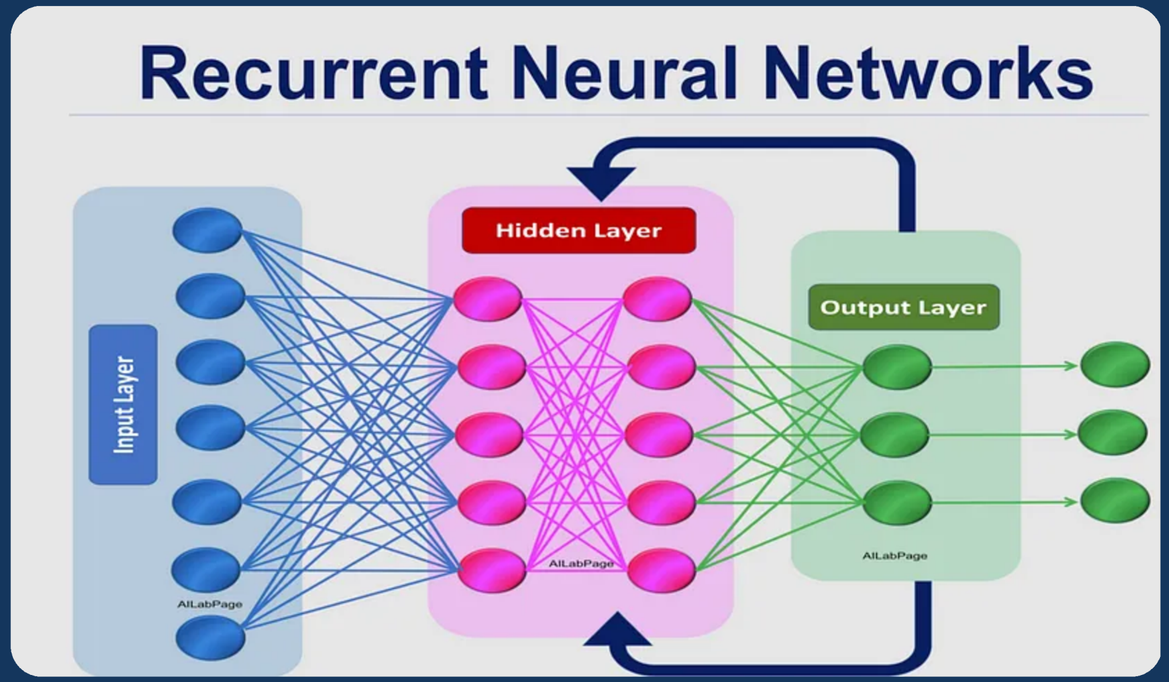


LLM Key Use Cases

- Text Summarization
- Chatbots / Virtual Assistants
- Translation
- Code Generation
- Question Answering



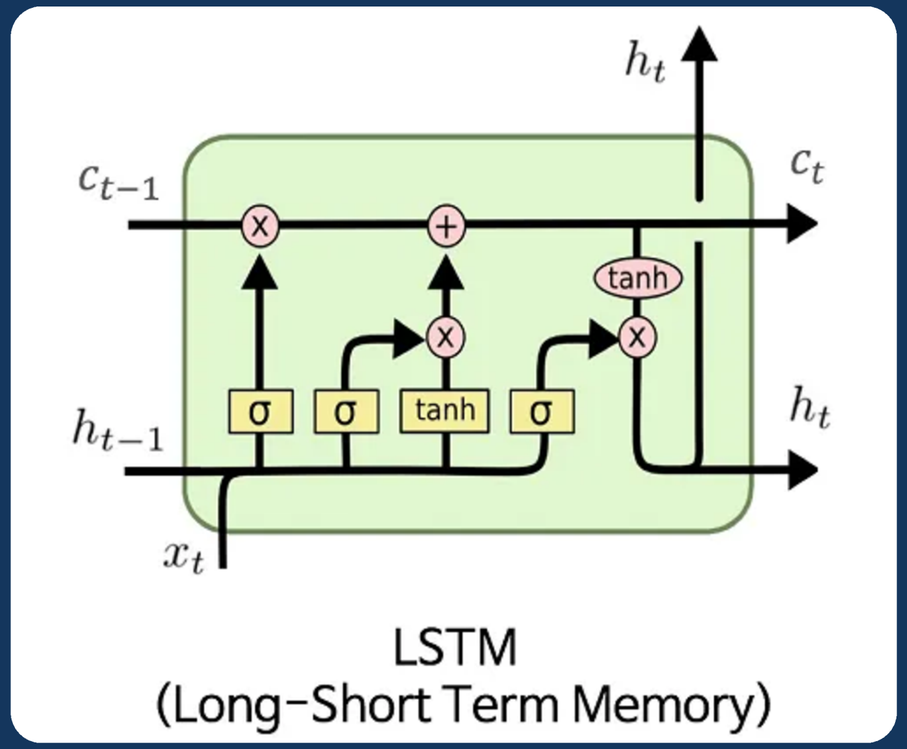
Text Generation before LLMs



- **What is sequential data?**
  - Order matters
  - Data points depend on each other
  - It is not fixed-size

- **RNN**
  - Processes sequences token by token
  - Shared hidden state captures context

Text Generation before LLMs



LSTM
(Long-Short Term Memory)

- **LSTM Improvements**
  - Gating mechanisms (input, forget, output)
  - Mitigates vanishing/exploding gradients
- **Challenges**
  - Long-range dependencies still hard
  - Process information sequentially
  - Slower training for large sequences
  - Prone to vanishing/exploding gradients (despite LSTM improvements)

Transformers

- Key Paper: "**Attention Is All You Need**" (Vaswani et al., 2017)
- Parallel Processing: Entire sequence at once
- Self-Attention (Multi-Head): Captures token relationships in parallel
- Encoder-Decoder Structure: Or encoder-only/decoder-only variants
- Scalability: Enables large-scale models (billions+ parameters)

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com

Noam Shazeer[*] Google Brain noam@google.com

Niki Parmar[*] Google Research nikip@google.com

Jakob Uszkoreit[*] Google Research usz@google.com

Llion Jones[*] Google Research llion@google.com

Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com

Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and

[*] Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†] Work performed while at Google Brain.

[‡] Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Attention is All You Need!

- Two main components: an encoder and a decoder
- Self-attention allows the model to understand the relationships
- Multi-head attention helps to focus on different aspects of the relationships
- Parallel Processing: Process entire sequences at once
- Long-Range Dependencies: self-attention mechanism helps
- Computational Cost: All that parallel processing and attention comes at a price.

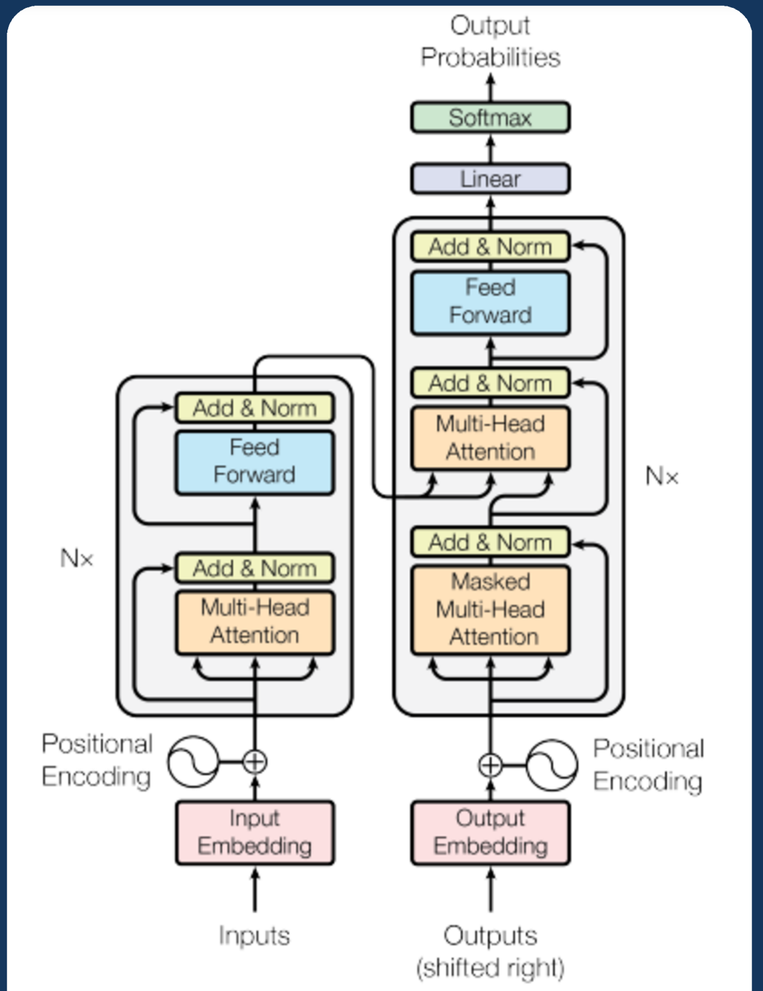


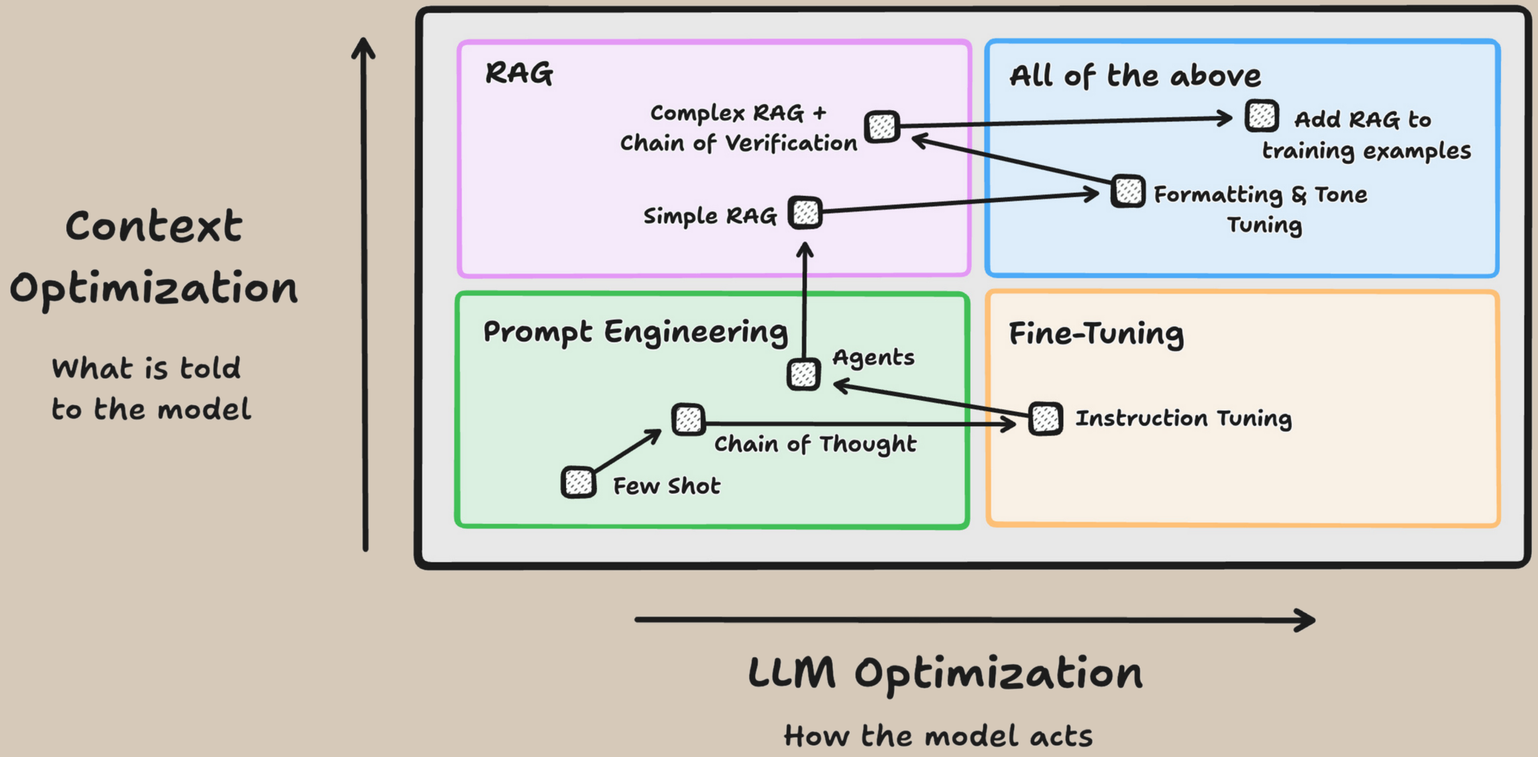
Figure 1: The Transformer - model architecture.

RNNs vs Transformers

Aspect	RNN (LSTM/GRU)	Transformer
Processing	Sequential	Parallel (Self-Attention)
Handling Long Context	Limited (Gradient Issues)	Very Effective (Attention)
Scalability	Difficult	Highly Scalable
Training Speed	Slower	Faster (GPU-friendly)
Typical Use Cases	Smaller-scale tasks	SOTA for NLP, LLMs

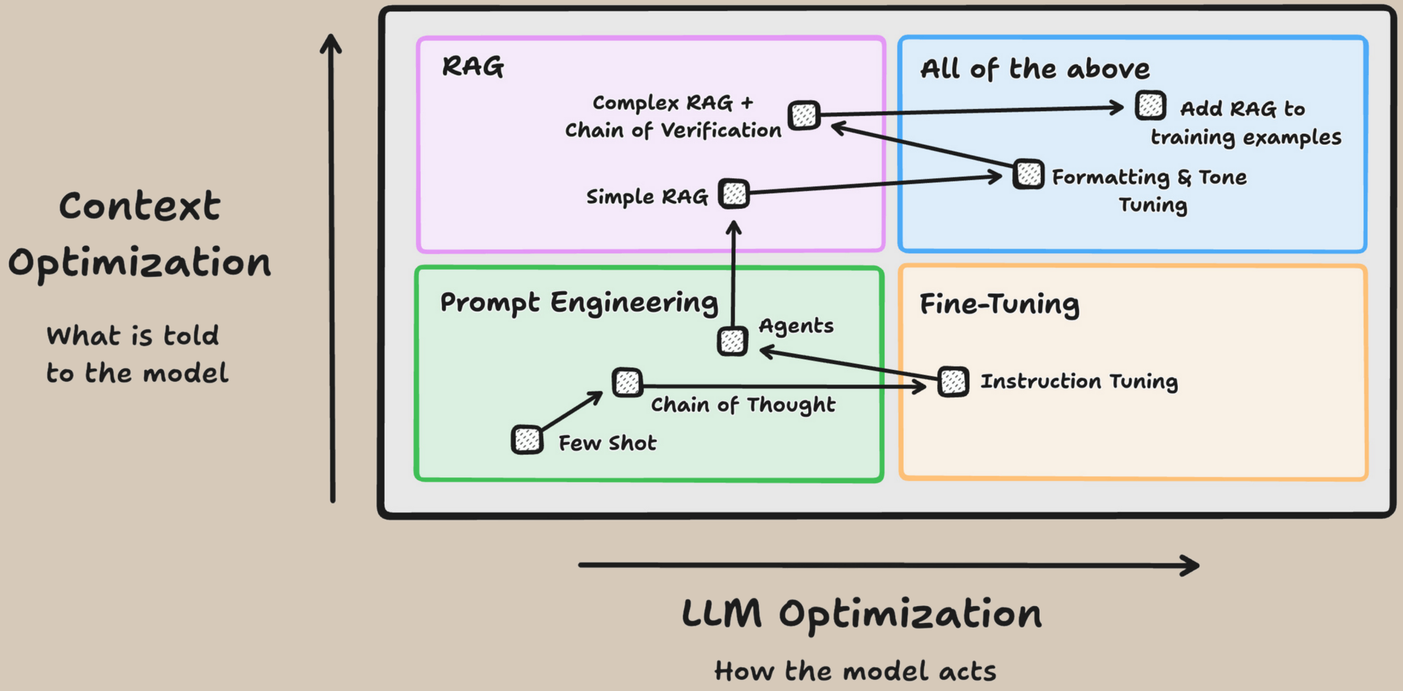
Improving LLMs

- **Prompt Engineering & In-Context Learning**
  - Tailor prompts to guide responses
- **Fine-Tuning**
  - Domain-specific training to internalize new data



Improving LLMs

- **Retrieval-Augmented Generation (RAG)**
  - Inject external knowledge at inference time
- **Why These Matter?**
  - Reduce hallucinations
  - Enhance domain accuracy
  - Minimize re-training costs



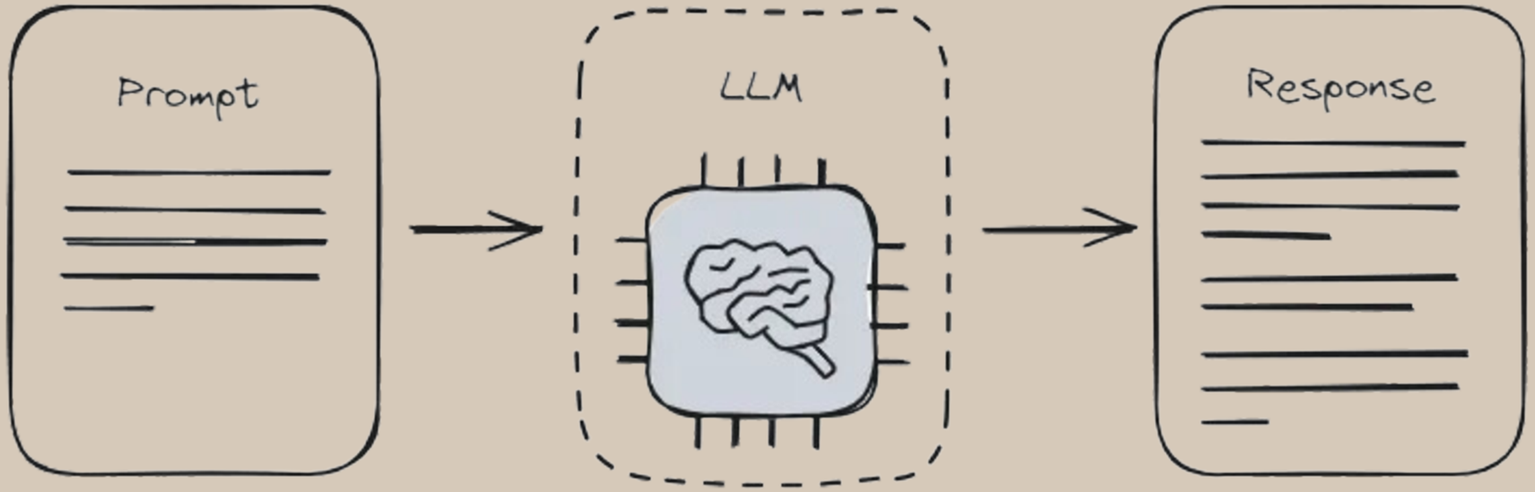
Prompt engineering

- **Prompt Enhancement**
  - Crafting inputs to elicit better responses
  - Small changes, big outcome differences

- **In-Context Learning**
  - Zero-Shot: No example provided
  - One-Shot: Single exemplar
  - Few-Shot: Multiple exemplars guiding output style

Practical Benefits Prompt Engineering

- No retraining
- Fast to implement
- Doesn’t require technical knowledge
- Domain adaptation via examples
- Often sufficient for many tasks



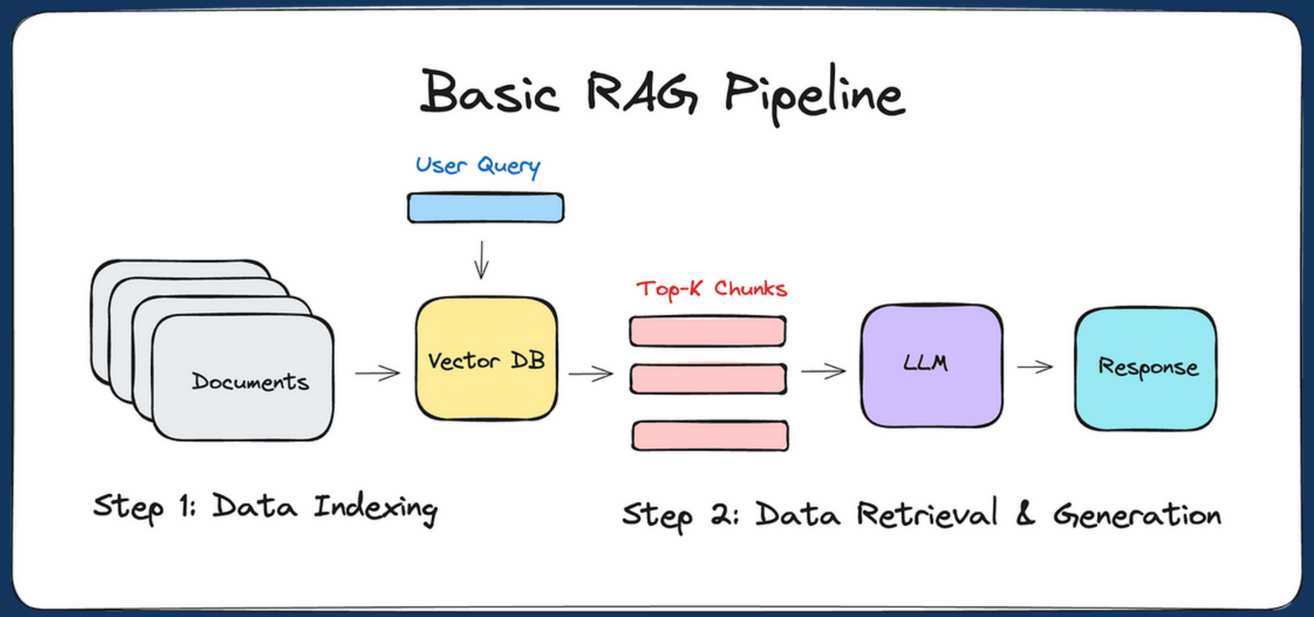
Fine Tuning

- Definition: Training a pre-trained model further on a specific dataset
- Pros: Domain specialization, consistent style, higher accuracy
- Cons: Requires labeled data, computationally expensive
- Typical Workflow: Prepare data Training Validation Deployment

- When to Use:
  - Highly specialized tasks
  - Long-term stable domains
  - Insufficient results from prompts alone

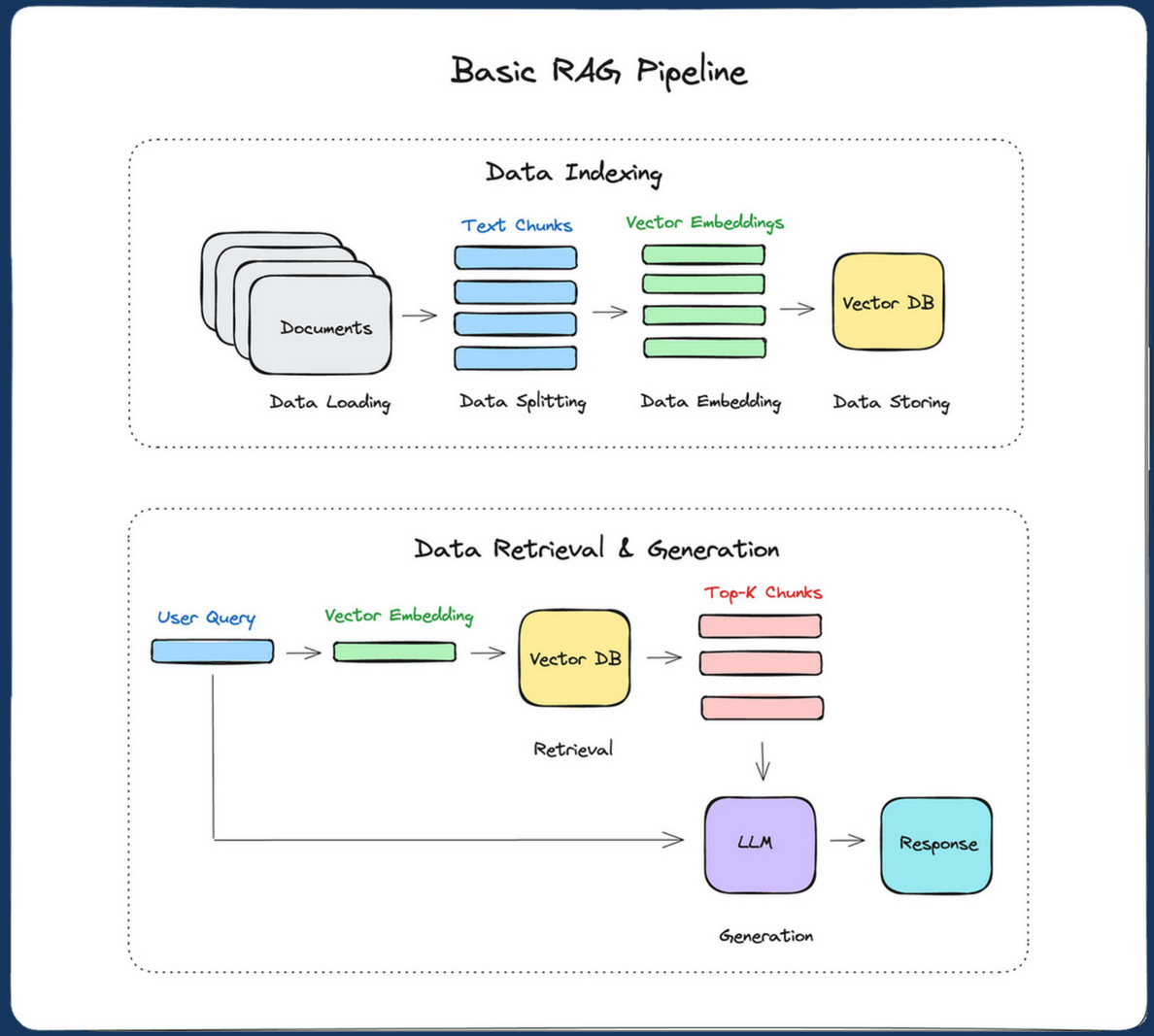
Retrieval-Augmented Generation (RAG)

- High-Level Steps:
  - User Query Ask a question.
  - Retrieve Relevant Documents From a Vector Database (e.g., Pinecone, Weaviate).
  - Combine Query + Context Include retrieved chunks in the prompt.
  - Generate Answer Pass to an LLM for context-aware output



Why RAG?

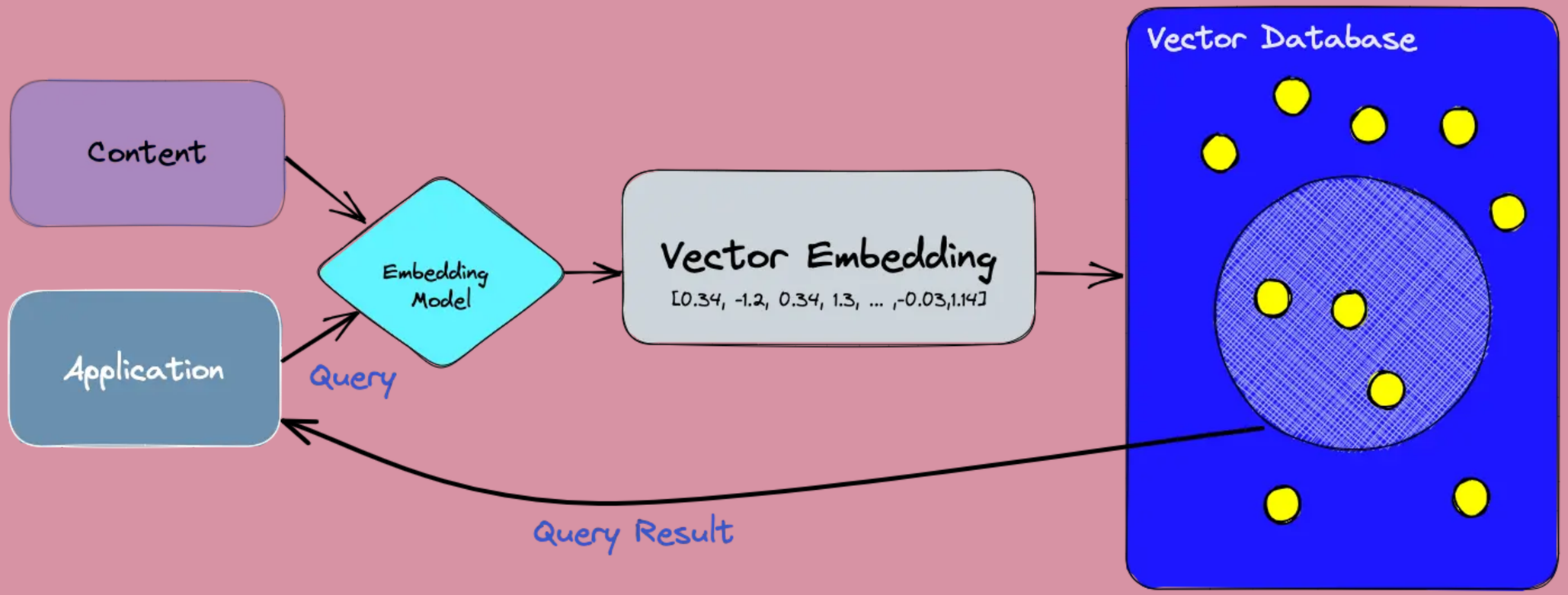
- **Dynamic Knowledge Injection**: Keeps responses updated without retraining.
- **Enhanced Accuracy**: Uses external data to ground answers.
- **Domain-Specific Context**: Tailors results to your proprietary or specialized data.

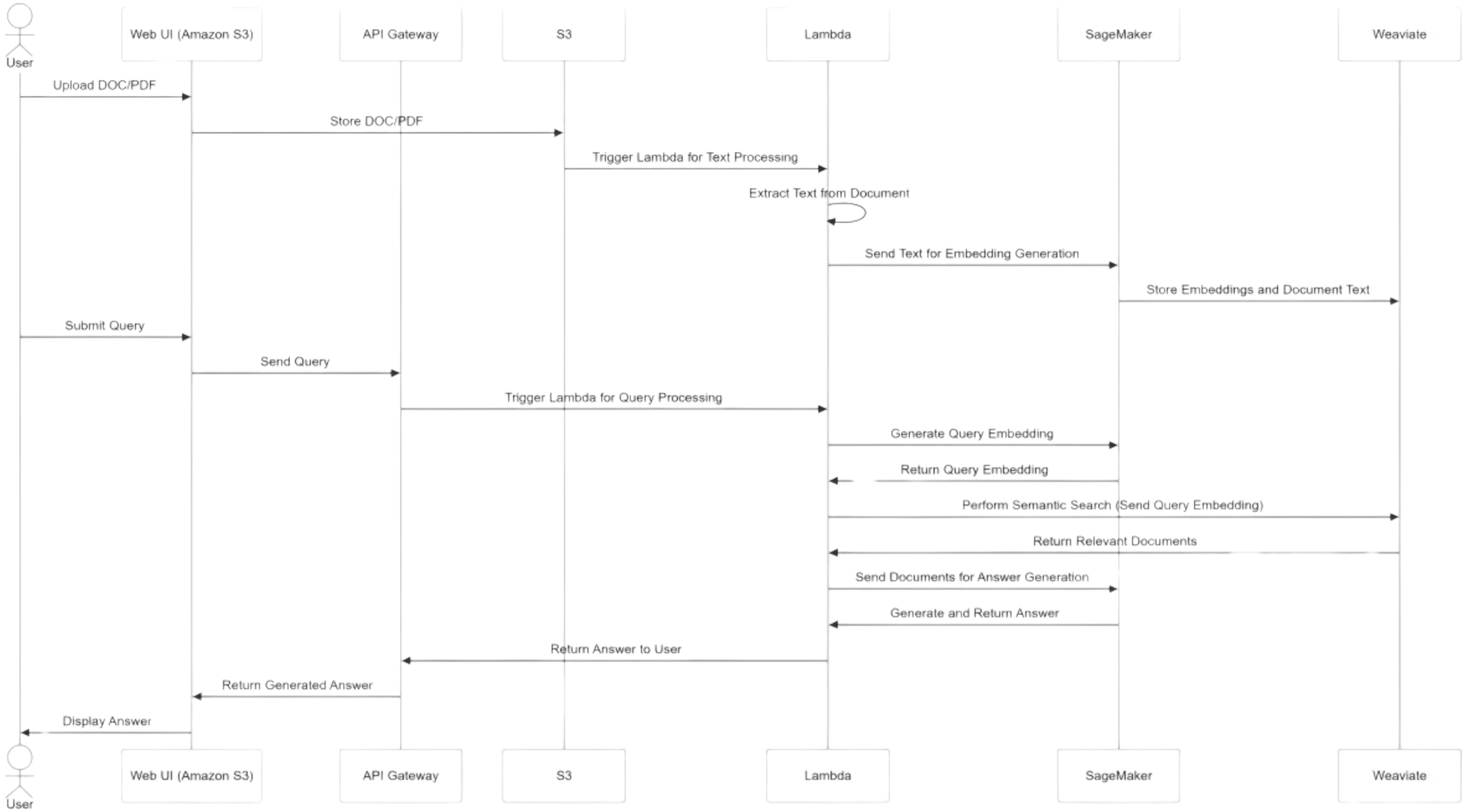


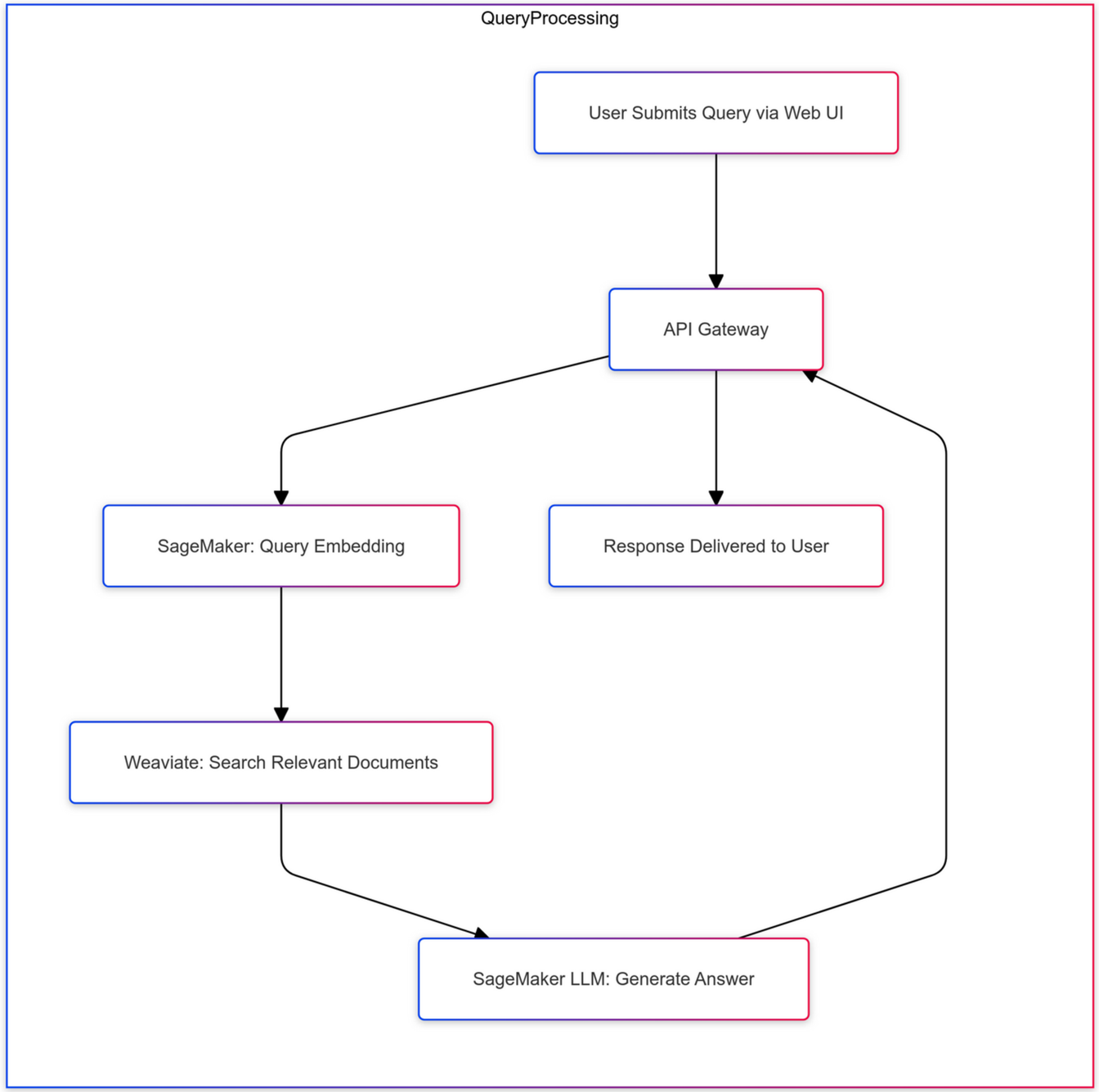
Retrieval-Augmented Generation (RAG):

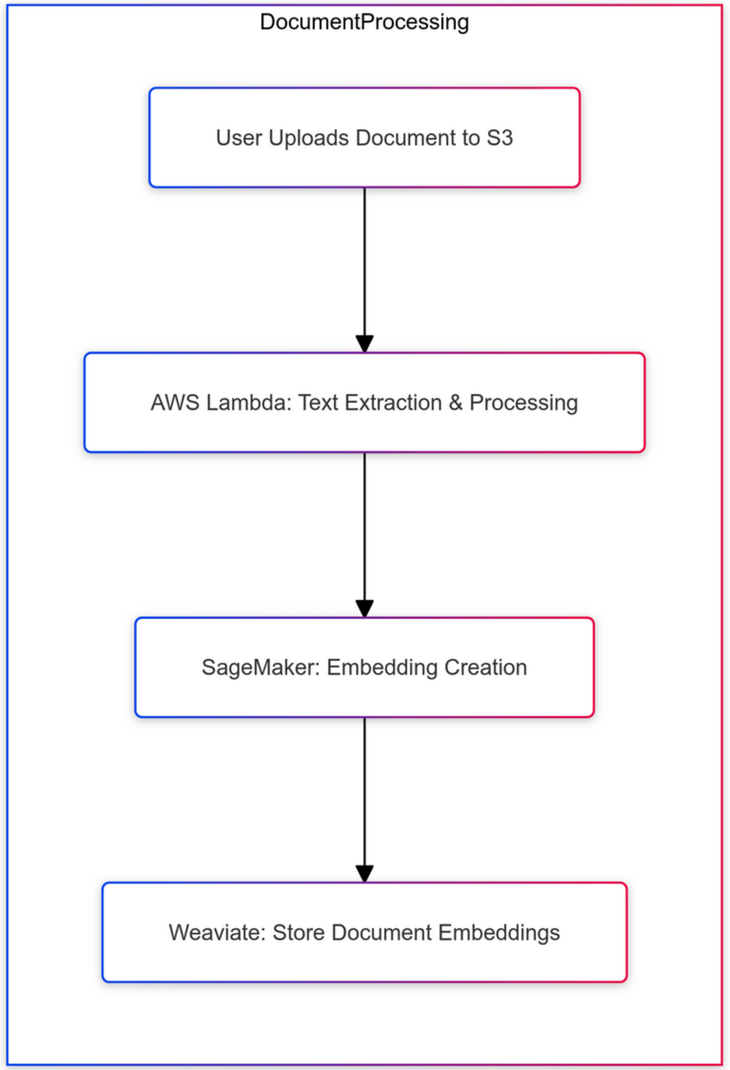
- Key Components:
  - Vector Store: Semantic search (e.g., Pinecone, FAISS).
  - Embedding Model: Converts text into vectors.
  - LLM: Generates answers based on retrieved context.
- Use Cases:
  - Customer Support: Provide real-time answers using FAQs or knowledge bases.
  - Enterprise Search: Retrieve internal policies or documents.
  - Education: Summarize textbooks or research papers.

Vector Databases – The Backbone of RAG




User
Web UI (Amazon S3)
API Gateway
S3
Lambda
SageMaker
Weaviate
Upload DOC/PDF
Store DOC/PDF
Trigger Lambda for Text Processing
Extract Text from Document
Send Text for Embedding Generation
Store Embeddings and Document Text
Submit Query
Send Query
Trigger Lambda for Query Processing
Generate Query Embedding
Return Query Embedding
Perform Semantic Search (Send Query Embedding)
Return Relevant Documents
Send Documents for Answer Generation
Generate and Return Answer
Return Answer to User
Return Generated Answer
Display Answer
User
Web UI (Amazon S3)
API Gateway
S3
Lambda
SageMaker
Weaviate


QueryProcessing
User Submits Query via Web UI
API Gateway
SageMaker: Query Embedding
Response Delivered to User
Weaviate: Search Relevant Documents
SageMaker LLM: Generate Answer


DocumentProcessing
User Uploads Document to S3
AWS Lambda: Text Extraction & Processing
SageMaker: Embedding Creation
Weaviate: Store Document Embeddings

RAG VS Fine-tuning

Category	RAG (Retrieval-Augmented Generation)	Fine-Tuning
Pros	- Dynamic: Supports rapidly changing or large datasets.	- Domain-Specific: Model internalizes specialized knowledge.
	- Cost-Effective: Avoids expensive re-training.	- Improved Performance: Tailored responses in specific contexts.
	- Scalable: Works with any external knowledge source.	
Cons	- Dependent on Retrieval: Quality of answers relies on embeddings and retrieval performance.	- Expensive: Requires labeled data, compute resources.
	- Latency: Inference depends on retrieval + generation.	- Static: Re-training needed for data updates.
When to Use	- Frequent data updates.	- Stable datasets or highly specific tasks (e.g., medical, legal).
	- Need dynamic or real-time knowledge injection.	
Hybrid Approach	Fine-tuned models combined with RAG for dynamic updates and enhanced domain knowledge.	

Thank You


KianKashfipour@gmail.com


Linkedin.com/in/kian79


Github.com/kian79