Retrieval-Augmented Generation (RAG)

LLMs and the Need for RAG

Bindeshwar Singh Kushwaha
PostNetwork Academy

Let's Start With a Question

Think About This

Suppose you ask a Large Language Model:

Let's Start With a Question

Think About This

Suppose you ask a Large Language Model:

"What is the medical history of my uncle?"

Let's Start With a Question

Think About This

Suppose you ask a Large Language Model:

"What is the medical history of my uncle?"

Will the LLM know the answer?

Let's Start With a Question

Think About This

Suppose you ask a Large Language Model:

"What is the medical history of my uncle?"

Will the LLM know the answer?

Answer

No.

Why Can't the LLM Answer?

- LLMs are trained on public Internet data.

Why Can't the LLM Answer?

- LLMs are trained on public Internet data.
- They do NOT have access to your private information.

Why Can't the LLM Answer?

- LLMs are trained on public Internet data.
- They do NOT have access to your private information.
- They do NOT know your family members.

Why Can't the LLM Answer?

- LLMs are trained on public Internet data.
- They do NOT have access to your private information.
- They do NOT know your family members.
- They cannot access your personal documents.

Why Can't the LLM Answer?

- LLMs are trained on public Internet data.
- They do NOT have access to your private information.
- They do NOT know your family members.
- They cannot access your personal documents.
- They may generate incorrect or assumed answers.

What Is Missing?

Core Missing Component

Access to external, real, and personal knowledge.

What Is Missing?

Core Missing Component

Access to external, real, and personal knowledge.

That is where **Retrieval-Augmented Generation (RAG)** comes in.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

- Store your documents in a database.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

- Store your documents in a database.
- Convert them into embeddings.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

- Store your documents in a database.
- Convert them into embeddings.
- Retrieve relevant information when a query is asked.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

- Store your documents in a database.
- Convert them into embeddings.
- Retrieve relevant information when a query is asked.
- Inject that information into the prompt.

How Do We Solve This?

Instead of Expecting Memory...

We give the LLM access to relevant information.

- Store your documents in a database.
- Convert them into embeddings.
- Retrieve relevant information when a query is asked.
- Inject that information into the prompt.
- Let the LLM generate a grounded response.

How Do We Solve This?

Instead of Expecting Memory...

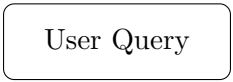
We give the LLM access to relevant information.

- Store your documents in a database.
- Convert them into embeddings.
- Retrieve relevant information when a query is asked.
- Inject that information into the prompt.
- Let the LLM generate a grounded response.

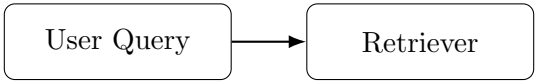
Core Idea

$$\text{Answer} = \text{LLM}(\text{Question} + \text{Retrieved Context})$$

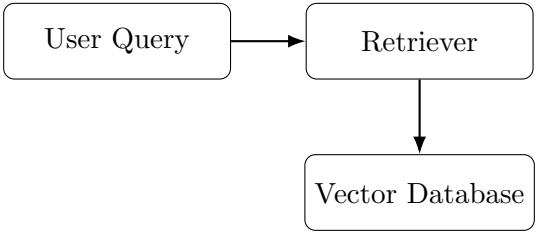
RAG Architecture Overview



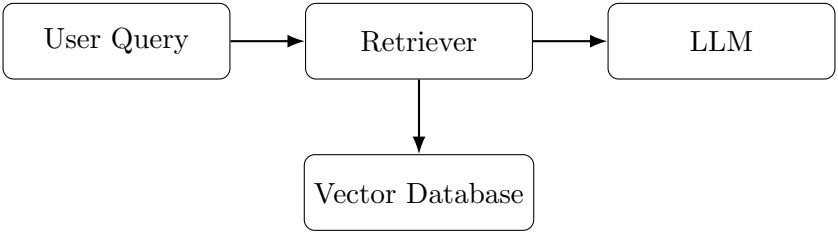
RAG Architecture Overview



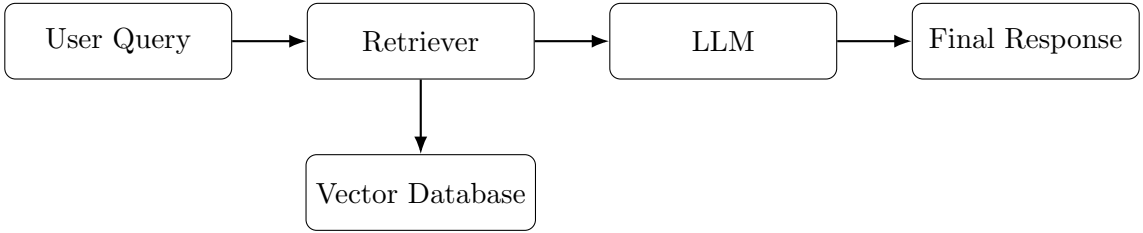
RAG Architecture Overview



RAG Architecture Overview



RAG Architecture Overview



Two Pipelines in RAG

1. Indexing Pipeline (Offline)

- Connect to data sources

Two Pipelines in RAG

1. Indexing Pipeline (Offline)

- Connect to data sources
- Extract and clean documents

Two Pipelines in RAG

1. Indexing Pipeline (Offline)

- Connect to data sources
- Extract and clean documents
- Split into chunks

Two Pipelines in RAG

1. Indexing Pipeline (Offline)

- Connect to data sources
- Extract and clean documents
- Split into chunks
- Convert into embeddings

Two Pipelines in RAG

1. Indexing Pipeline (Offline)

- Connect to data sources
- Extract and clean documents
- Split into chunks
- Convert into embeddings
- Store in vector database

Two Pipelines in RAG (Continued)

2. Generation Pipeline (Online)

- User query

Two Pipelines in RAG (Continued)

2. Generation Pipeline (Online)

- User query
- Retrieve similar chunks

Two Pipelines in RAG (Continued)

2. Generation Pipeline (Online)

- User query
- Retrieve similar chunks
- Augment prompt

Two Pipelines in RAG (Continued)

2. Generation Pipeline (Online)

- User query
- Retrieve similar chunks
- Augment prompt
- Generate contextual answer

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

- Converts words into numbers.

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

- Converts words into numbers.
- Similar meanings $\rightarrow$ similar vectors.

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

- Converts words into numbers.
- Similar meanings $\rightarrow$ similar vectors.
- Enables semantic search.

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

- Converts words into numbers.
- Similar meanings $\rightarrow$ similar vectors.
- Enables semantic search.
- Foundation of retrieval in RAG.

What is an Embedding?

Core Idea

An embedding is a numerical vector representation of text.

- Converts words into numbers.
- Similar meanings $\rightarrow$ similar vectors.
- Enables semantic search.
- Foundation of retrieval in RAG.

$$\text{"Diabetes Treatment"} \rightarrow [0.12, -0.88, 0.45, \ldots]$$

Why Not Simple Keyword Search?

- Keyword search matches exact words.

Why Not Simple Keyword Search?

- Keyword search matches exact words.
- But meaning can vary.

Why Not Simple Keyword Search?

- Keyword search matches exact words.
- But meaning can vary.
- Example:

Why Not Simple Keyword Search?

- Keyword search matches exact words.
- But meaning can vary.
- Example:

Example

Query: "Sugar disease"
Document: "Diabetes mellitus"

Why Not Simple Keyword Search?

- Keyword search matches exact words.
- But meaning can vary.
- Example:

Example

Query: "Sugar disease"
Document: "Diabetes mellitus"

Keyword search fails.

Why Not Simple Keyword Search?

- Keyword search matches exact words.
- But meaning can vary.
- Example:

Example

Query: "Sugar disease"
Document: "Diabetes mellitus"

Keyword search fails.
Embedding search succeeds.

What is a Vector Database?

Definition

A vector database stores embeddings and performs similarity search.

What is a Vector Database?

Definition

A vector database stores embeddings and performs similarity search.

- Stores high-dimensional vectors.

What is a Vector Database?

Definition

A vector database stores embeddings and performs similarity search.

- Stores high-dimensional vectors.
- Uses cosine similarity or distance metrics.

What is a Vector Database?

Definition

A vector database stores embeddings and performs similarity search.

- Stores high-dimensional vectors.
- Uses cosine similarity or distance metrics.
- Retrieves top-K similar documents.

What is a Vector Database?

Definition

A vector database stores embeddings and performs similarity search.

- Stores high-dimensional vectors.
- Uses cosine similarity or distance metrics.
- Retrieves top-K similar documents.

$$\text{Similarity} = \cos(\theta)$$

What Happens During Retrieval?

1. User asks question.

What Happens During Retrieval?

1. User asks question.
2. Question converted into embedding.

What Happens During Retrieval?

1. User asks question.
2. Question converted into embedding.
3. Compare with stored embeddings.

What Happens During Retrieval?

1. User asks question.
2. Question converted into embedding.
3. Compare with stored embeddings.
4. Select top-K similar chunks.

What Happens During Retrieval?

1. User asks question.
2. Question converted into embedding.
3. Compare with stored embeddings.
4. Select top-K similar chunks.
5. Return them as context.

What Happens During Retrieval?

1. User asks question.
2. Question converted into embedding.
3. Compare with stored embeddings.
4. Select top-K similar chunks.
5. Return them as context.

Important

The LLM does NOT search the database directly. The retriever does.

Prompt Augmentation

Core Mechanism

We append retrieved documents to the original question.

Prompt Augmentation

Core Mechanism

We append retrieved documents to the original question.

$$\text{Final Prompt} = \text{Question} + \text{Retrieved Documents}$$

Prompt Augmentation

Core Mechanism

We append retrieved documents to the original question.

$$\text{Final Prompt} = \text{Question} + \text{Retrieved Documents}$$

- LLM now sees relevant context.

Prompt Augmentation

Core Mechanism

We append retrieved documents to the original question.

$$\text{Final Prompt} = \text{Question} + \text{Retrieved Documents}$$

- LLM now sees relevant context.
- Reduces hallucination.

Prompt Augmentation

Core Mechanism

We append retrieved documents to the original question.

$$\text{Final Prompt} = \text{Question} + \text{Retrieved Documents}$$

- LLM now sees relevant context.
- Reduces hallucination.
- Produces grounded answers.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.
- Memory may be outdated.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.
- Memory may be outdated.
- May guess when uncertain.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.
- Memory may be outdated.
- May guess when uncertain.
- RAG provides evidence.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.
- Memory may be outdated.
- May guess when uncertain.
- RAG provides evidence.
- LLM answers based on retrieved text.

How RAG Reduces Hallucination

- Pure LLM relies on parametric memory.
- Memory may be outdated.
- May guess when uncertain.
- RAG provides evidence.
- LLM answers based on retrieved text.

Result

Higher factual accuracy and transparency.

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.

RAG

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.

RAG

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.

RAG

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.
- Hard to update.

RAG

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.
- Hard to update.

RAG

- Keep model fixed.

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.
- Hard to update.

RAG

- Keep model fixed.
- Update database only.

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.
- Hard to update.

RAG

- Keep model fixed.
- Update database only.
- Fast and scalable.

RAG vs Fine-Tuning (Conceptual Difference)

Fine-Tuning

- Modify model weights.
- Expensive.
- Needs retraining.
- Hard to update.

RAG

- Keep model fixed.
- Update database only.
- Fast and scalable.
- Enterprise friendly.

Real-World Applications of RAG

- Enterprise knowledge assistants.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.
- Medical information systems.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.
- Medical information systems.
- Customer support automation.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.
- Medical information systems.
- Customer support automation.
- Research paper assistants.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.
- Medical information systems.
- Customer support automation.
- Research paper assistants.
- Codebase question answering.

Real-World Applications of RAG

- Enterprise knowledge assistants.
- Legal document analysis.
- Medical information systems.
- Customer support automation.
- Research paper assistants.
- Codebase question answering.

Conclusion

RAG is the backbone of modern enterprise AI systems.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.
- Create AI assistants for law firms, hospitals, startups.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.
- Create AI assistants for law firms, hospitals, startups.
- Develop internal knowledge search systems.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.
- Create AI assistants for law firms, hospitals, startups.
- Develop internal knowledge search systems.
- Build customer support automation tools.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.
- Create AI assistants for law firms, hospitals, startups.
- Develop internal knowledge search systems.
- Build customer support automation tools.
- Convert PDFs and databases into intelligent AI systems.

Freelance Opportunities in RAG

High-Demand Skills (2025 and Beyond)

Companies need AI systems trained on their own data.

- Build enterprise chatbots using company documents.
- Create AI assistants for law firms, hospitals, startups.
- Develop internal knowledge search systems.
- Build customer support automation tools.
- Convert PDFs and databases into intelligent AI systems.

Opportunity

RAG Developers are in massive demand on freelancing platforms.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.
- Make decisions.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.
- Make decisions.
- Call APIs.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.
- Make decisions.
- Call APIs.
- Execute tools.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.
- Make decisions.
- Call APIs.
- Execute tools.
- Perform multi-step reasoning.

From RAG to Agentic AI

Next Evolution

RAG answers questions.

Agentic AI takes actions.

- Retrieve information.
- Make decisions.
- Call APIs.
- Execute tools.
- Perform multi-step reasoning.

Future

The future is RAG + Tools + Autonomous Agents.

Python Libraries for Building RAG Systems

- LangChain

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)
- ChromaDB

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)
- ChromaDB
- OpenAI / HuggingFace APIs

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)
- ChromaDB
- OpenAI / HuggingFace APIs
- FastAPI for deployment

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)
- ChromaDB
- OpenAI / HuggingFace APIs
- FastAPI for deployment
- Streamlit / Gradio for UI

Python Libraries for Building RAG Systems

- LangChain
- LlamaIndex
- Haystack
- FAISS (Vector Search)
- ChromaDB
- OpenAI / HuggingFace APIs
- FastAPI for deployment
- Streamlit / Gradio for UI

Stack Example

Python + LangChain + FAISS + GPT + FastAPI

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio
- Download local LLM (LLaMA / Mistral)

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio
- Download local LLM (LLaMA / Mistral)
- Use FAISS or Chroma for vector DB

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio
- Download local LLM (LLaMA / Mistral)
- Use FAISS or Chroma for vector DB
- Run everything on your machine

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio
- Download local LLM (LLaMA / Mistral)
- Use FAISS or Chroma for vector DB
- Run everything on your machine
- Deploy with FastAPI on localhost

Running RAG on Local Server (Offline Setup)

Why Local Server?

• No API cost • Full data privacy • Works without internet • Ideal for enterprise and research

- Install Ollama or LM Studio
- Download local LLM (LLaMA / Mistral)
- Use FAISS or Chroma for vector DB
- Run everything on your machine
- Deploy with FastAPI on localhost

Architecture

User → Retriever → Local LLM → Response

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.
4. Store in vector database.

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.
4. Store in vector database.
5. Build retrieval + LLM pipeline.

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.
4. Store in vector database.
5. Build retrieval + LLM pipeline.
6. Create web interface.

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.
4. Store in vector database.
5. Build retrieval + LLM pipeline.
6. Create web interface.
7. Deploy on cloud (AWS / Azure / GCP).

Deploy RAG on cloud

1. Collect client documents (PDF, CSV, Web).
2. Clean and split data.
3. Create embeddings.
4. Store in vector database.
5. Build retrieval + LLM pipeline.
6. Create web interface.
7. Deploy on cloud (AWS / Azure / GCP).

Result

Client gets a private AI assistant trained on their data.

Where to Find Freelance Work?

- Upwork

Where to Find Freelance Work?

- Upwork
- Freelancer

Where to Find Freelance Work?

- Upwork
- Freelancer
- Fiverr

Where to Find Freelance Work?

- Upwork
- Freelancer
- Fiverr
- Toptal

Where to Find Freelance Work?

- Upwork
- Freelancer
- Fiverr
- Toptal
- LinkedIn AI Consulting

Where to Find Freelance Work?

- Upwork
- Freelancer
- Fiverr
- Toptal
- LinkedIn AI Consulting
- Direct startup outreach

Where to Find Freelance Work?

- Upwork
- Freelancer
- Fiverr
- Toptal
- LinkedIn AI Consulting
- Direct startup outreach

Tip

Position yourself as: **"RAG and Agentic AI Developer"**

Why You Should Learn RAG Now

- AI is transforming every industry.

Why You Should Learn RAG Now

- AI is transforming every industry.
- Enterprises need private AI systems.

Why You Should Learn RAG Now

- AI is transforming every industry.
- Enterprises need private AI systems.
- RAG is practical and implementable.

Why You Should Learn RAG Now

- AI is transforming every industry.
- Enterprises need private AI systems.
- RAG is practical and implementable.
- Freelancers can build real solutions.

Why You Should Learn RAG Now

- AI is transforming every industry.
- Enterprises need private AI systems.
- RAG is practical and implementable.
- Freelancers can build real solutions.
- Startup founders can build AI products.

Why You Should Learn RAG Now

- AI is transforming every industry.
- Enterprises need private AI systems.
- RAG is practical and implementable.
- Freelancers can build real solutions.
- Startup founders can build AI products.

Your Advantage

If you master RAG + Agentic AI today, you become future-ready.

Reach PostNetwork Academy

Website

www.postnetwork.co

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

LinkedIn

www.linkedin.com/company/postnetworkacademy

Reach PostNetwork Academy

Website

www.postnetwork.co

YouTube Channel

www.youtube.com/@postnetworkacademy

LinkedIn

www.linkedin.com/company/postnetworkacademy

GitHub

www.github.com/postnetworkacademy

Thank You!