Member-only story

LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : KG basics : 2

Aniket Hingane | Day Manager, Night Coder · Follow

8 min read · 2 hours ago

Before we build lets learn KG basics Today ! Tomorrow let’s build it !

LLM Apps : Why Knowledge Graphs are super critical to know if you care about RAG : 1

AI Apps Need RAG, RAG Need Knowledge Graph ! Next part we will code RAG on KG.

medium.com

First what is a graph?

A knowledge graph is fundamentally a type of graph — but not the kind you'd see with X and Y axes. Instead, it uses the mathematical concept of a graph structure:

- ***Nodes*** **:** These are the building blocks. They represent entities like people, places, things, concepts — anything you want to model
- ***Edges*** **:** The connections between nodes. These are labeled relationships giving context and meaning to how things are related (e.g., "is a friend of," "lives in," "is a type of").

Why a Graph Model?

1. ***Mimics Real-World Connections*** **:** The way we intuitively understand the world is often through interconnections. A graph structure beautifully captures these relationships, unlike traditional databases that rely on rigid tables.
2. ***Semantic Representation*** **:** Graphs, especially knowledge graphs, don't just show connections; the labels on those edges encode *meaning*. This adds a layer of understanding that computers can process.
3. ***Flexibility*** **:** You're not confined to pre-determined structures. As your knowledge evolves, you can easily add new types of nodes or define new kinds of relationships. This adaptability is perfect for modeling dynamic information domains.

Let's visualize a Knowledge Graph

Imagine a simplified knowledge graph:

Nodes

- Person (Tom Hanks)
- Movie (Forrest Gump)
- City (Los Angeles)

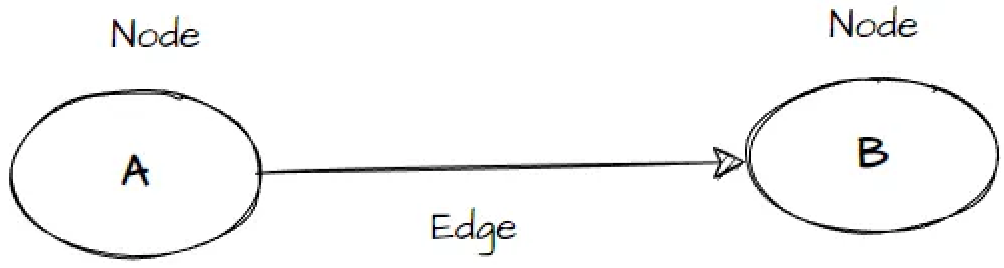
Edges

- (Tom Hanks)-[:ACTED_IN]-> (Forrest Gump)
- (Tom Hanks)-[:BORN_IN]->(Los Angeles)
- (Forrest Gump)-[:FILMED_IN]->(Los Angeles)

Notice how even this tiny graph tells a little story! You can visualize it like a network with circles (nodes) and lines (edges).

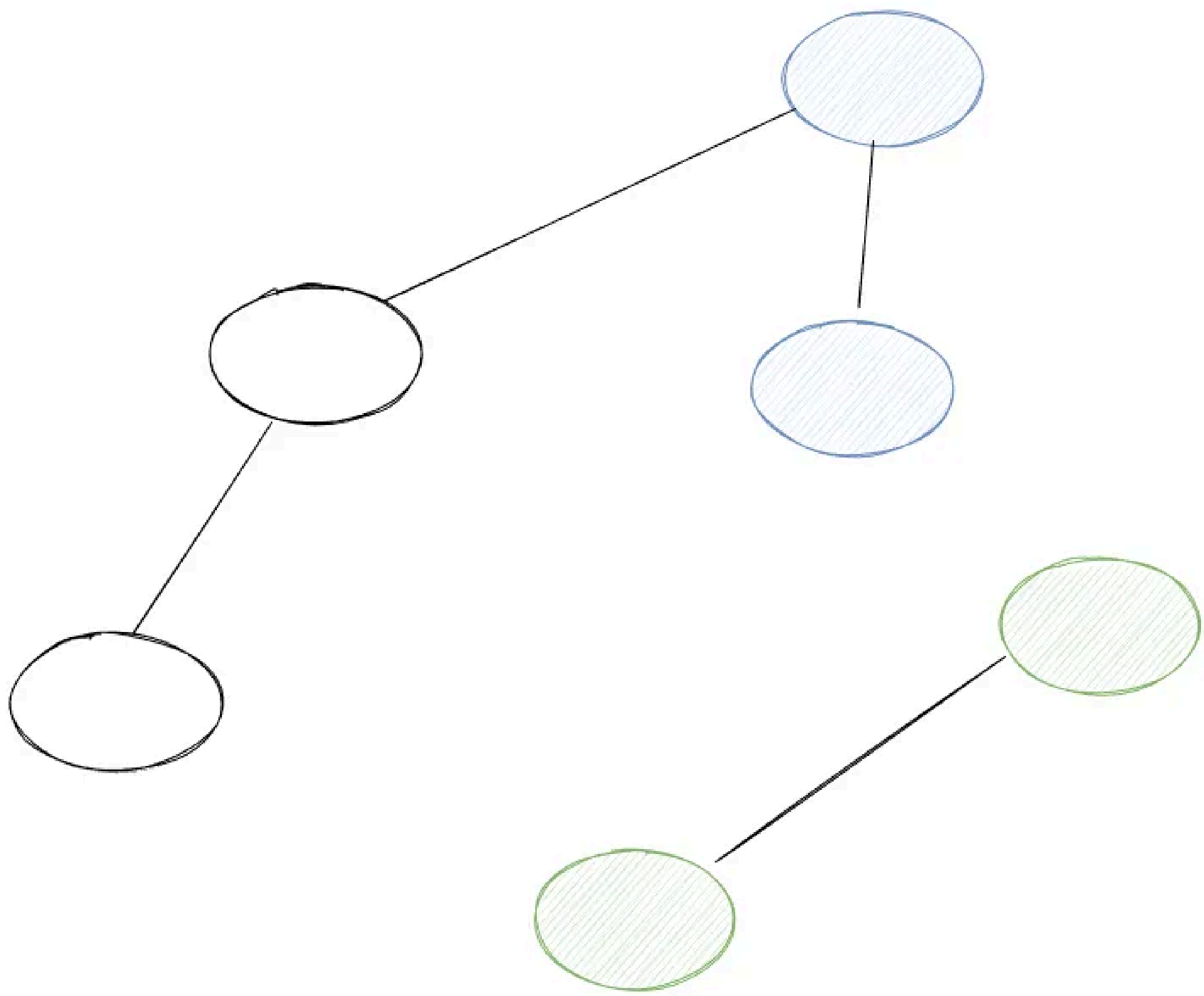
Alright ! let's get into basics

Basic building blocks



- **Nodes:** These are the building blocks. They represent entities like people, places, things, concepts — anything you want to model
- **Edges:** The connections between nodes. These are labeled relationships giving context and meaning to how things are related (e.g., "is a friend of," "lives in," "is a type of").

Isolated and no direction



Isolated Graphs

The Loners :

An isolated graph consists of nodes that have no connections whatsoever to other nodes. It's a collection of entities without any defined relationships between them.

Real-World Example:

Imagine a dataset of newly registered users on a social media platform before they've formed any friendships or connections. Each user would be represented as an isolated node.

Why They Matter — Isolated graphs might represent:

- Starting points for a growing knowledge graph, where relationships are added over time.
- Entities that are temporarily disconnected or with no known associations within the scope of your data.

Undirected Graphs

No One-Way Streets :

In undirected graphs, edges (relationships) lack inherent directionality. If a node A is connected to node B, it implies a mutual connection where B is also connected to A.

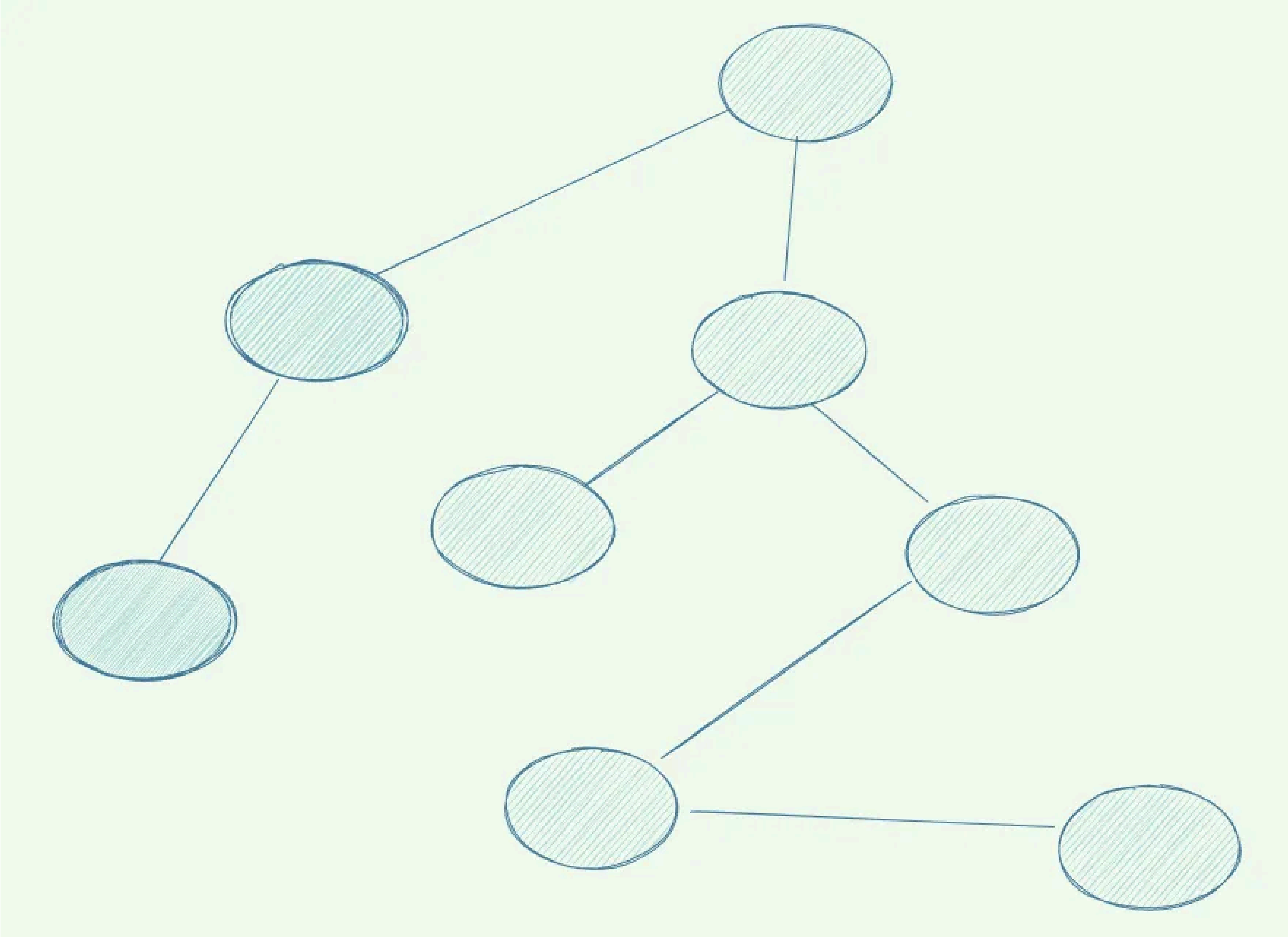
Real-World Examples:

- A friendship network: If Alice is friends with Bob, the friendship is reciprocal.
- A road map of a city with two-way streets: You can travel in either direction between locations.

The Intersection: Isolated Nodes in Undirected Graphs

It's completely possible to have isolated nodes within an undirected graph. Imagine a graph where some nodes represent people with friendships, and some remain isolated — people with no friends represented in that graph.

Fully connected graph



In a fully connected graph, every single node has a direct or indirect edge (relationship) to every other node within the graph. This creates a densely interconnected web of entities.

Imagine a group of friends where everyone is friends with everyone else. There are no isolated individuals or unconnected pairs.

Real-World Examples (Often Smaller Scale):

- A close-knit team where everyone collaborates directly with all other members.
- A communication network where any device can directly send messages to any other device.
- A sports round-robin tournament where every player competes against every other player.

Key Properties of Fully Connected Graphs

1. **High Connectivity:** It's impossible to disconnect a fully connected graph by removing just a few edges.

2. **Calculating Edges:** For a graph with 'n' nodes, a fully connected graph will have n(n-1)/2 edges. This grows quickly! Imagine every node needing to connect to all the remaining nodes.

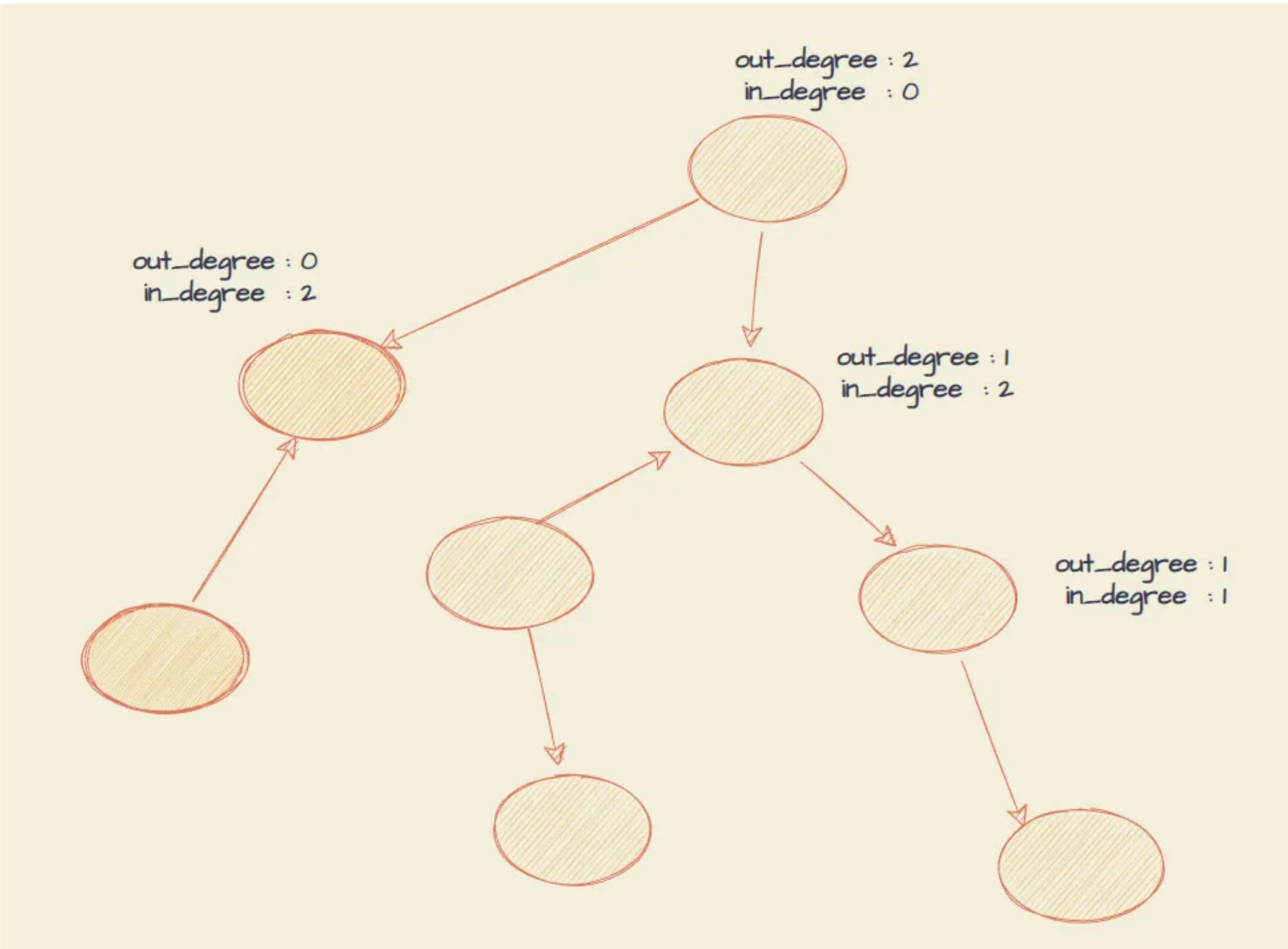
3. **Undirected vs. Directed:** Fully connected graphs can exist in both forms:

- **Undirected:** Relationships are reciprocal (a friendship network).
- **Directed:** Relationships have directionality (a hierarchy where every person reports to every other person, which might get a bit messy!)

Why Fully Connected Graphs Matter

- **Modeling Tightly Coupled Systems:** When direct interaction or influence between all components is critical, a fully connected model might apply.
- **Robustness (to a point):** They can be somewhat resistant to disruption. Even if some nodes fail, many paths of communication or influence may remain.

Directed graph and degree



Where Relationships Have Direction

One-Way Streets:

In a directed graph, edges (relationships) have an intrinsic direction, indicated by arrows. This signifies a 'from' and 'to' aspect to the relationship.

Real-World Examples:

- Financial transactions: Money flows *from* a payer *to* a payee.
- Social media follows: You follow someone, but they might not follow you back.
- A family tree: Parent-child relationships have a clear directionality.

Degrees in Directed Graphs: It's About In and Out

In directed graphs, we need to refine the concept of the degree of a node:

- ***Indegree***: The number of edges directed *towards* a node. In our financial example, this would be how much money a person has received.
- ***Outdegree***: The number of edges directed *away* from a node. This represents how much money a person has spent in the transaction example.
- **Visualizing It:** Imagine a node with several arrows pointing into it (high indegree) and a few pointing out (lower outdegree).

Why This Distinction Matters

Indegree and outdegree provide more nuanced insights when analyzing directed graphs:

- **Influence:** Nodes with high indegree might be viewed as influential in the network (many followers, lots of incoming links).
- **Sources vs. Sinks:** Nodes with high outdegree can be considered sources or distributors (spreading information, initiating transactions).
- **Flow Analysis:** By analyzing indegree and outdegree, you can track how entities, information, or resources move through the network.

Example: Twitter Network

Consider a directed graph representing Twitter.

- **Node:** User Account

- **Relationship:** “follows” (directed)

A celebrity account likely has:

- **High Indegree:** Lots of people follow them.
- **Lower Outdegree:** They follow a smaller set of users.

With that, let’s see Knowledge Graph

At its core, a knowledge graph is a graph-based data structure that represents knowledge in a structured and machine-readable format. It organizes information as a collection of entities (nodes) and relationships (edges) between them. This graph structure allows for the representation of complex relationships and semantic connections, making it an effective model for capturing and understanding knowledge.

Components of a Knowledge Graph

1. Nodes (Entities): Nodes represent entities or concepts in the knowledge domain. These entities can range from real-world objects such as people, places, and events to abstract concepts, classes, or categories.

2. Edges (Relationships): Edges define the relationships or connections between entities in the knowledge graph. They capture the semantic associations, dependencies, or properties that exist between different entities.

3. Properties: Properties are attributes or characteristics associated with nodes or edges in the knowledge graph. They provide additional context or information about the entities and relationships.

Construction of a Knowledge Graph:

Building a knowledge graph involves several steps, including data extraction, entity recognition, relationship extraction, and graph construction.

The process typically starts with data collection from various sources such as structured databases, unstructured text, or web pages. Next, entities are identified and extracted from the data using techniques like named entity recognition (NER) or entity linking. Relationships between entities are then inferred or extracted based on the contextual information and semantic connections within the data.

Finally, the extracted entities and relationships are organized into a graph structure, forming the knowledge graph.

Applications of Knowledge Graphs:

Knowledge graphs find applications across various domains and industries, revolutionizing the way we interact with data and information. Some common applications include:

1. Semantic Search: Knowledge graphs power semantic search engines that understand the meaning and context of user queries, enabling more relevant and precise search results.

2. Recommendation Systems: Knowledge graphs drive recommendation systems by modeling user preferences, item attributes, and semantic relationships to deliver personalized recommendations.

3. Question Answering Systems: Knowledge graphs support question answering systems by providing structured knowledge repositories that can be queried to retrieve relevant information and answer user questions.

4. Data Integration and Linkage: Knowledge graphs facilitate data integration and linkage by connecting disparate data sources and harmonizing heterogeneous data formats and schemas.

5. Knowledge Representation and Reasoning: Knowledge graphs serve as a formal representation of knowledge that can be leveraged for automated reasoning, inference, and decision-making in intelligent systems.

Future Prospects of Knowledge Graphs: (In my opinion)

As the volume and complexity of data continue to grow, the role of knowledge graphs in organizing and leveraging this data will become increasingly important. Future advancements in knowledge graph technology are expected to focus on:

1. Scalability and Performance: Improving the scalability and performance of knowledge graph systems to handle large-scale, real-time data processing and analytics.

2. Interoperability and Standards: Enhancing interoperability and standardization efforts to enable seamless integration and exchange of knowledge graphs across different platforms and domains.

3. Knowledge Graph Reasoning: Advancing techniques for knowledge graph reasoning and inference to support more complex and context-aware decision-making in intelligent systems.

4. Domain-Specific Knowledge Graphs: Creating domain-specific knowledge graphs tailored to specific industries or use cases to capture and leverage specialized knowledge and expertise.

Conclusion:

Knowledge graphs represent a fundamental shift in how we organize, represent, and leverage knowledge in the digital age. By capturing complex relationships and semantic connections in a structured graph format, knowledge graphs enable more intelligent and context-aware applications across various domains.

As we continue to harness the power of knowledge graphs, their impact on artificial intelligence, data science, and information technology will only continue to grow, shaping the future of intelligent systems and data-driven decision-making.

Next part , its time to build one !

Machine Learning | Artificial Intelligence | Knowledge Graph | Llm | System Design Interview

Follow

Written by Aniket Hingane | Day Manager, Night Coder

26 Followers

Passionate about simplifying software concepts and design through concise articles. Making complexity accessible, one short piece at a time.

More from Aniket Hingane | Day Manager, Night Coder

Aniket Hingane | Day Manager, Night Coder

LLM apps : Why you MUST know Semantic Router in 2024 : Part 1

Must know LLM apps decision making layer, don't miss out !

· 6 min read · Jan 23, 2024

--

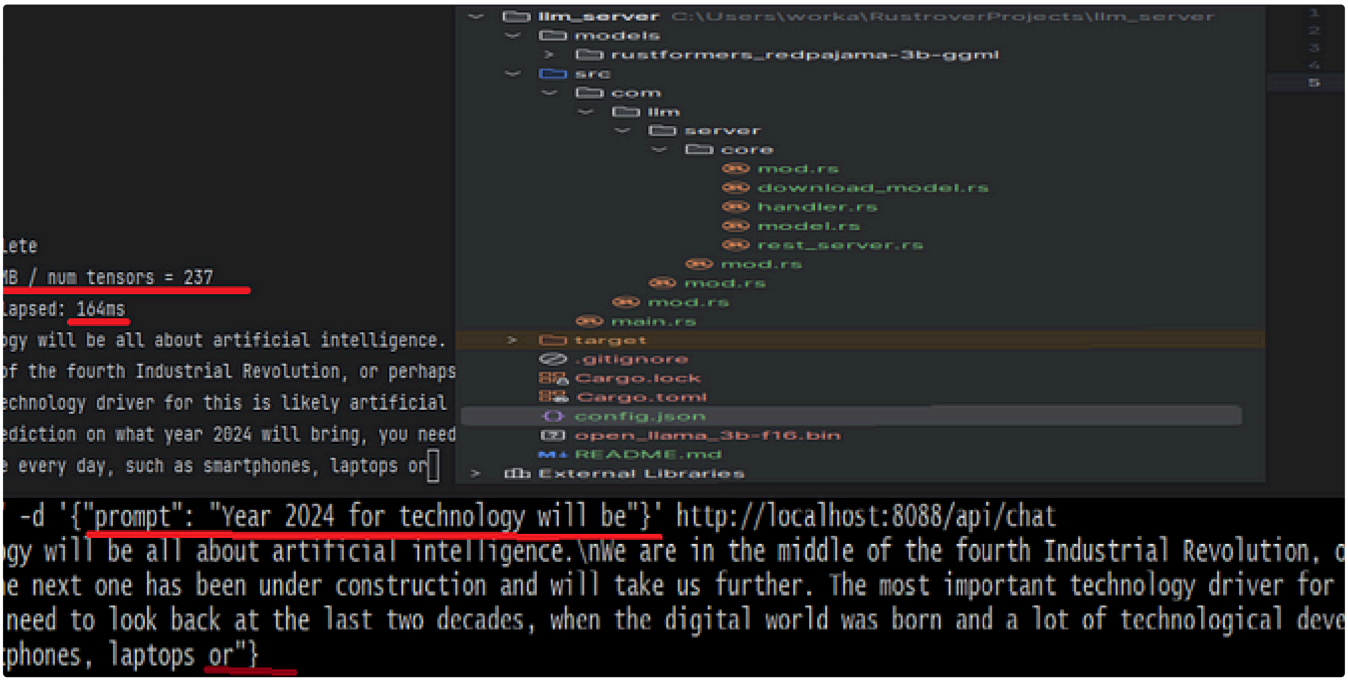
Aniket Hingane | Day Manager, Night Coder

LLM apps : Why you MUST know Semantic Router in 2024 : Part 2

Leveraging Semantic Router for Enhanced Prompting in Your Applications

· 5 min read · Jan 24, 2024

--



Aniket Hingane | Day Manager, Night Coder

Must know for AI App Dev : Let's serve LLMs via Rust programing : LLM Handler Build

Let's build LLM handler and run super fast inference!

· 5 min read · 5 days ago

--

Aniket Hingane | Day Manager, Night Coder

Refined RAG : Duo of Merge Retrieval & Long Context Reorder : Part 2

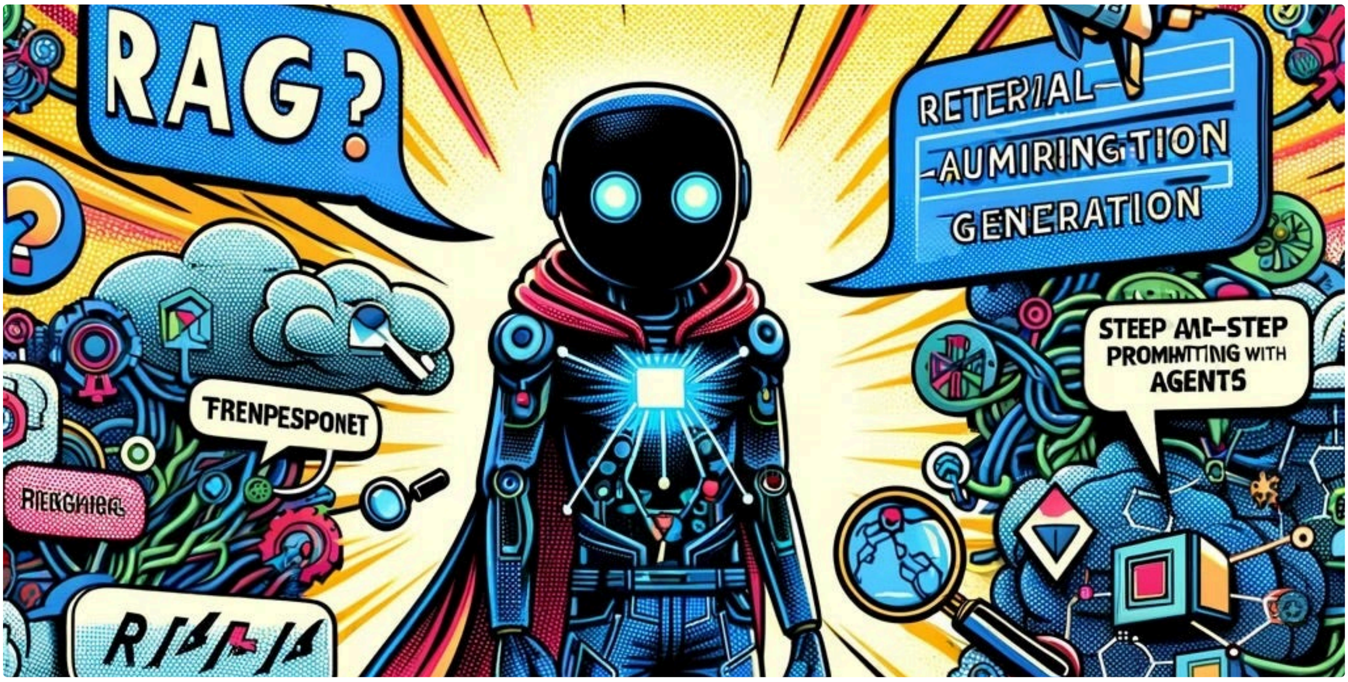
It's time to code !

· 7 min read · Jan 21, 2024

--

See all from Aniket Hingane | Day Manager, Night Coder

Recommended from Medium



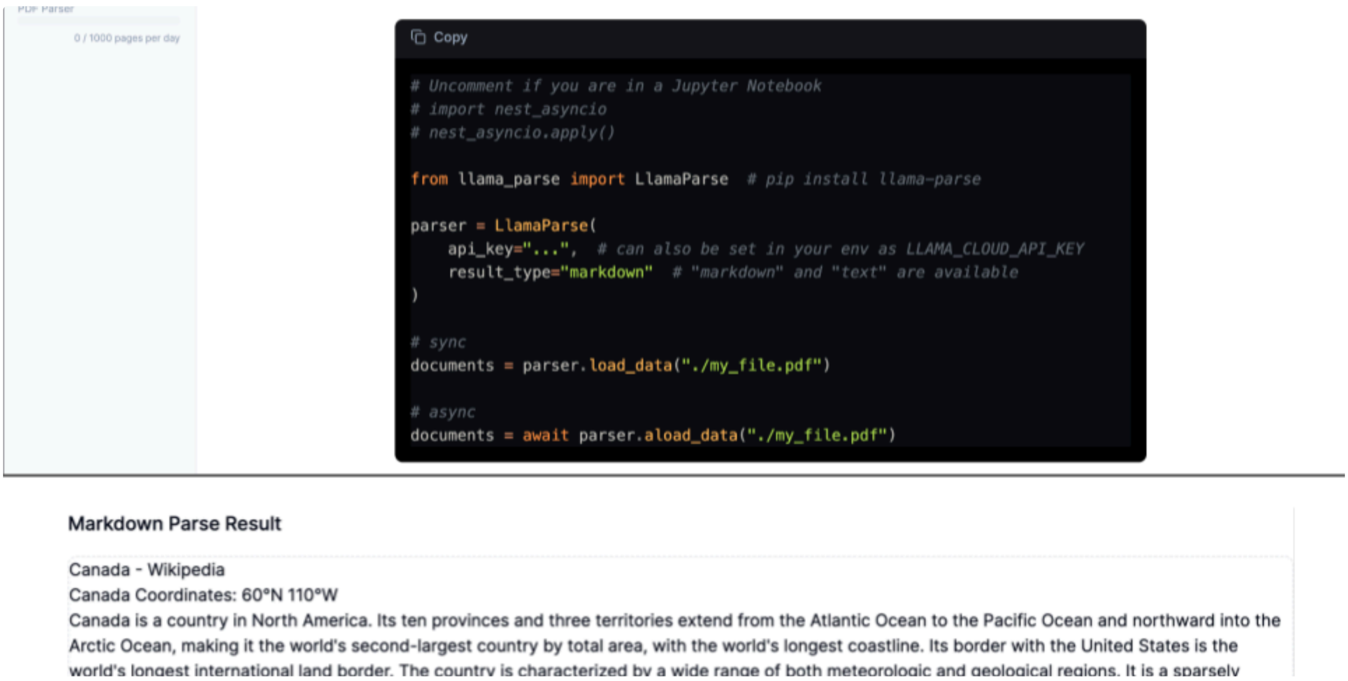
Dr. Varshita Sher in Towards Data Science

Using LangChain ReAct Agents for Answering Multi-hop Questions in RAG Systems

Useful when answering complex queries on internal documents in a step-by-step manner with ReAct and Open AI Tools agents.

42 min read · Feb 15, 2024

-- 4



Jerry Liu in LlamaIndex Blog

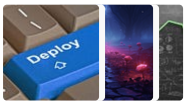
Introducing LlamaCloud and LlamaParse

Today is a big day for the LlamaIndex ecosystem: we are announcing LlamaCloud, a new generation of managed parsing, ingestion, and…

8 min read · 2 days ago

-- 11

Lists



Predictive Modeling w/ Python

20 stories · 926 saves

Natural Language Processing

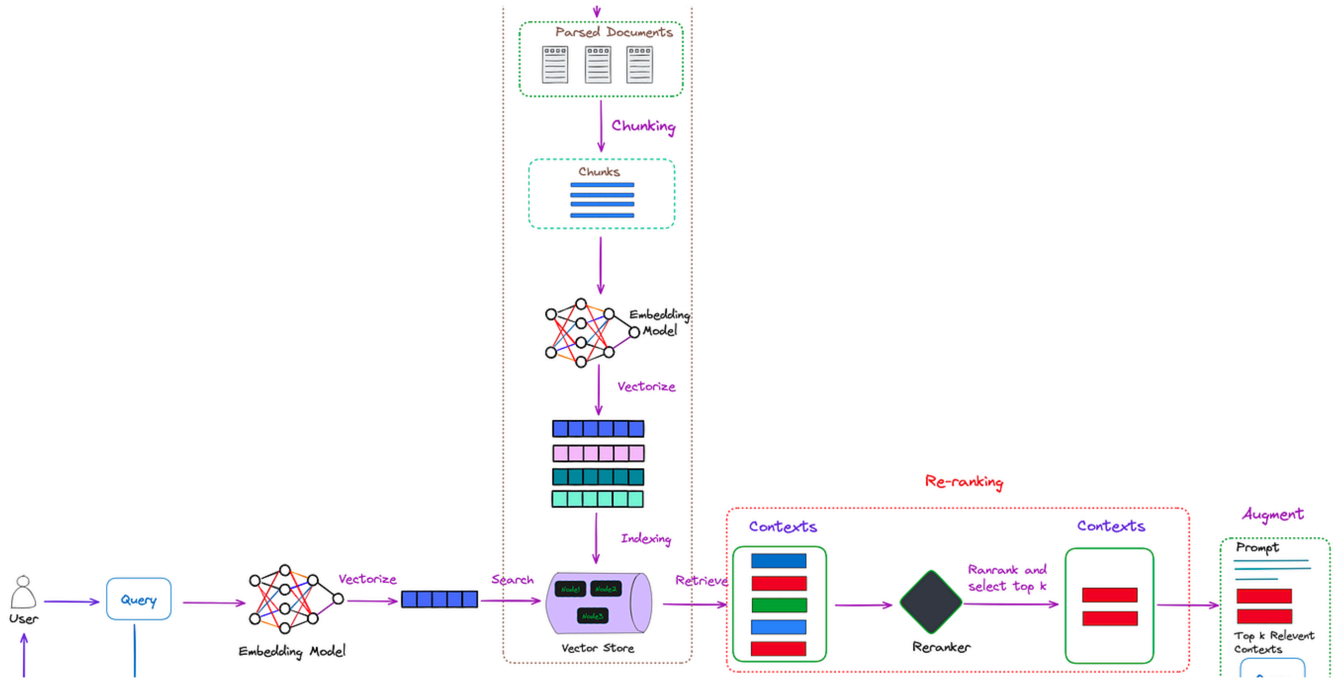
1223 stories · 702 saves

AI Regulation

6 stories · 324 saves

Practical Guides to Machine Learning

10 stories · 1090 saves



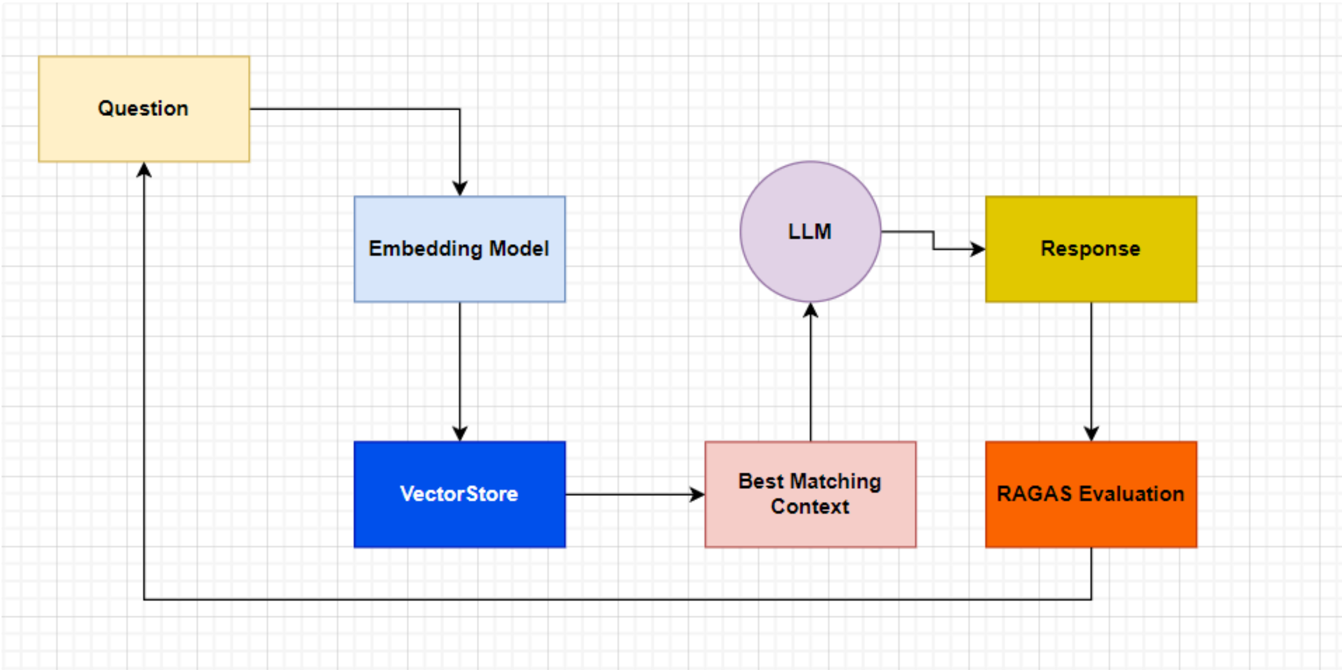
 Florian June in Towards AI

Advanced RAG 04: Re-ranking

From Principles to Two Mainstream Implementation Methods

· 11 min read · Feb 14, 2024

--



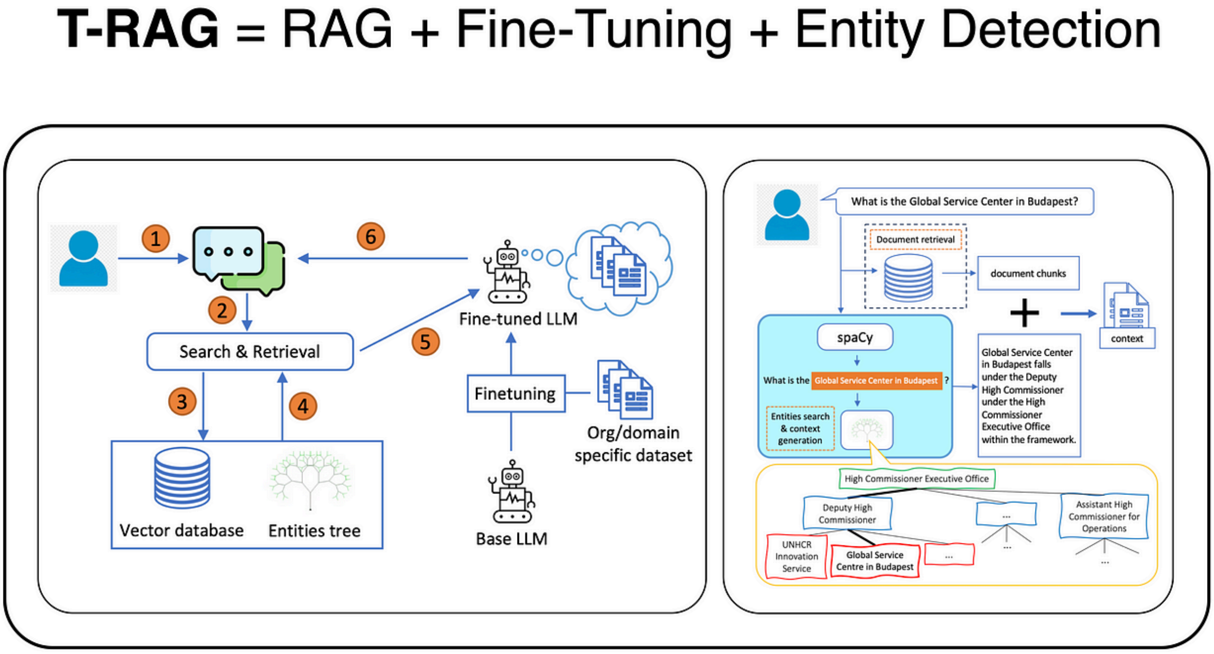
Plaban Nayak in AI Planet

Evaluating Naive RAG and Advanced RAG pipeline using langchain v.0.1.0 and RAGAS

What is RAG(Retrieval Augmented Generation) ?

25 min read · Feb 11, 2024

-- 2



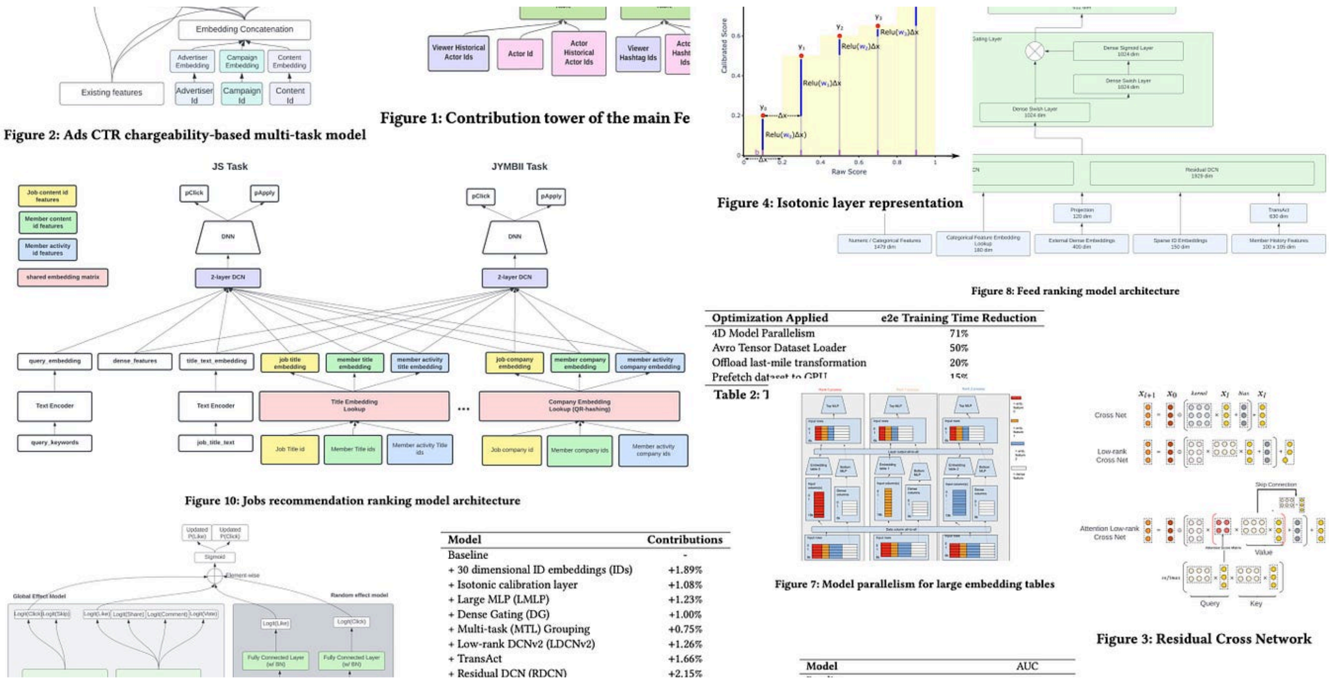
Cobus Greyling

T-RAG = RAG + Fine-Tuning + Entity Detection

The T-RAG approach is premised on combining RAG architecture with an open-source fine-tuned LLM and an entities tree vector database. The...

5 min read · Feb 15, 2024

-- 8



Andrew Lukyanenko in GoPenAI

Paper Review: LiRank: Industrial Large Scale Ranking Models at LinkedIn

Ranking model at LinkedIn

8 min read · 4 days ago

--

See more recommendations