Transforming Natural Language to SQL and Insights for E-commerce with LlamaIndex, GPT3.5, and Streamlit

Harshad Suryawanshi · Follow
Published in LlamaIndex Blog
5 min read · Dec 17, 2023

In the dynamic world of AI and data analytics, the ability to bridge the gap between complex data queries and non-technical users is a game-changer. My latest project, Na2SQL, showcases this exciting advancement. Leveraging the power of LlamaIndex and OpenAI's GPT-3.5, this app allows users, regardless of their SQL knowledge, to derive valuable insights from a database using simple natural language.

Features

- **Intuitive Natural Language Queries:** The core of this application is its ability to understand and process natural language queries. Users can ask questions in plain English and receive SQL queries and insights in return, all without any prior SQL experience.
- **Advanced Data Processing:** The app doesn't just stop at generating SQL queries; it executes these queries and analyzes the results to provide meaningful insights, making it a powerful tool for data analysis.
- **User-Friendly Interface with Streamlit:** I chose Streamlit for its simplicity and effectiveness in creating interactive web applications. The app's interface is straightforward, ensuring a smooth user experience.
- **Database Viewer:** An interactive database viewer in the sidebar on the left allows users to explore the database structure, enhancing their understanding and interaction with the data.

The Tech Stack

This project harmoniously integrates several advanced technologies:

1. **OpenAI's GPT-3.5:** At the heart of the application is GPT-3.5, enabling the app to understand natural natural language queries and transform them into valid SQL queries. Furthermore, it also generates the final analysis considering both the

user's query and the SQL output, thereby providing a comprehensive and relevant response.

2. **LlamaIndex:** A pivotal component of the app is LlamaIndex's SQLTableQueryEngine. This powerful tool translates natural language queries into SQL, handles the execution of these queries, and plays a significant role in the subsequent analysis using GPT 3.5. Its integration ensures a smooth transition from user inputs to database insights, culminating in a meaningful final analysis that encapsulates the entire natural language-to-SQL-to-execution process.

3. **LlamaIndex's Streamlit LlamaPack:** Using LlamaIndex's Streamlit LlamaPack, we quickly assemble and highly functional Streamlit UI. This framework significantly simplifies the UI development process, allowing for rapid deployment and an enhanced user experience.

4. **SQLite Database:** The app interacts with an dummy SQLite ecommerce database, showcasing its ability to work with real-world data.

Deep Dive into the Code

In the heart of the application lies `app.py`, a script that brings to life the seamless interaction between natural language processing and SQL query generation.

This code is an evolution of the **Streamlit chatbot LlamaPack** available on Llama Hub, further tailored to meet the specific needs of ecommerce data analytics. Let's dive into some key portions of the `app.py` script:

1. Initial Imports and Setup

The script begins by importing necessary modules such as Streamlit, SQLAlchemy for database interaction, LlamaIndex for language model services, and other essential libraries.

```
import streamlit as st
from sqlalchemy import create_engine, inspect
from typing import Dict, Any

from llama_index import (
    VectorStoreIndex,
    ServiceContext,
    download_loader,
)
```

```
from llama_index.llama_pack.base import BaseLlamaPack
from llama_index.llms import OpenAI
import openai
import os
import pandas as pd
```

2. StreamlitChatPack Class

The `StreamlitChatPack` class extends the base LlamaPack, setting up the page and modules necessary for the app's functionality.

```
class StreamlitChatPack(BaseLlamaPack):

    def __init__(
        self,
        page: str = "Natural Language to SQL Query",
        run_from_main: bool = False,
        **kwargs: Any,
    ) -> None:
        """Init params."""
        self.page = page

    # ... other methods ...
```

3. The `run` Method

This method is where the magic happens. It sets up the Streamlit page configuration and initializes the chat functionality.

```
def run(self, *args: Any, **kwargs: Any) -> Any:
    """Run the pipeline."""
    import streamlit as st

    st.set_page_config(
        page_title=f"{self.page}",
        layout="centered",
        initial_sidebar_state="auto",
        menu_items=None,
    )

    # ... rest of the run method ...
```

4. Database Schema Viewer in the Sidebar

A helpful feature is the Database Schema Viewer, conveniently located in the sidebar. This viewer serves as a reference tool, allowing users to see the structure and content of the database tables, enhancing their understanding of the data they're querying.

```
# Sidebar for database schema viewer
st.sidebar.markdown("## Database Schema Viewer")

# Create an inspector object
inspector = inspect(engine)

# Get list of tables in the database
table_names = inspector.get_table_names()

# Sidebar selection for tables
selected_table = st.sidebar.selectbox("Select a Table", table_names)

db_file = 'ecommerce_platform1.db'
conn = sqlite3.connect(db_file)

# Display the selected table
if selected_table:
    df = get_table_data(selected_table, conn)
    st.sidebar.text(f"Data for table '{selected_table}':")
    st.sidebar.dataframe(df)

# Close the connection
conn.close()
```

5. Database Interaction and LLM Integration

This part of the code loads the database from disk and initializes the LLM and the service context for use with Llamaindex. I've used GPT3.5 here, but you can easily swap it out with any other LLM of your choice.

```
# Function to load database and LLM
def load_db_llm():
    engine = create_engine("sqlite:///ecommerce_platform1.db")
    sql_database = SQLDatabase(engine)  # Include all tables
    llm2 = OpenAI(temperature=0.1, model="gpt-3.5-turbo-1106")
    service_context = ServiceContext.from_defaults(llm=llm2)
    return sql_database, service_context, engine
```

```
sql_database, service_context, engine = load_db_llm()
```

6. Initializing the NLSQLTableQueryEngine

One of the most critical aspects of the application is the initialization of the `NLSQLTableQueryEngine`. This is where the app sets up the engine responsible for converting natural language queries into SQL queries, executing them and generating the final response, all with the help of GPT 3.5.

```
# Initializing the query engine
if "query_engine" not in st.session_state:
    st.session_state["query_engine"] = NLSQLTableQueryEngine(
        sql_database=sql_database,
        synthesize_response=True,
        service_context=service_context
    )
```

7. User Interaction and Displaying Results

The script provides an interactive interface for users to input natural language queries, which are then translated into SQL queries and executed.

The app concludes by displaying the SQL queries and responses, offering an informative and engaging user experience

```
if prompt := st.chat_input(
    "Enter your natural language query about the database"
):  # Prompt for user input and save to chat history
    with st.chat_message("user"):
        st.write(prompt)
    add_to_message_history("user", prompt)

# If last message is not from assistant, generate a new response
if st.session_state["messages"][-1]["role"] != "assistant":
    with st.spinner():
        with st.chat_message("assistant"):
            response = st.session_state["query_engine"].query("User Question:"+
            sql_query = f"```sql\n{response.metadata['sql_query']}\n```\n**Resp
            response_container = st.empty()
```

```
            response_container.write(sql_query)
            add_to_message_history("assistant", sql_query)
```

Wrapping Up

This app is more than just a tool; it's a step towards making data analytics accessible to a broader audience. It embodies the potential of AI in simplifying complex data interactions. I invite you to explore this application, witness its capabilities, and join me in this journey towards a more inclusive data-driven future.

Link to Github Repo

Connect with Me on LinkedIn

Linkedin Post:

Harshad S. on LinkedIn: #ai #llamaindex #streamlit #largelanguagemodels...

AI Prototype 6: Transforming Natural Language to SQL and Insights for E-commerce with LlamaIndex, OpenAI GPT3.5, and...

www.linkedin.com

Natural Language To Sql | Llamaindex | Llm | Gpt35 | OpenAI

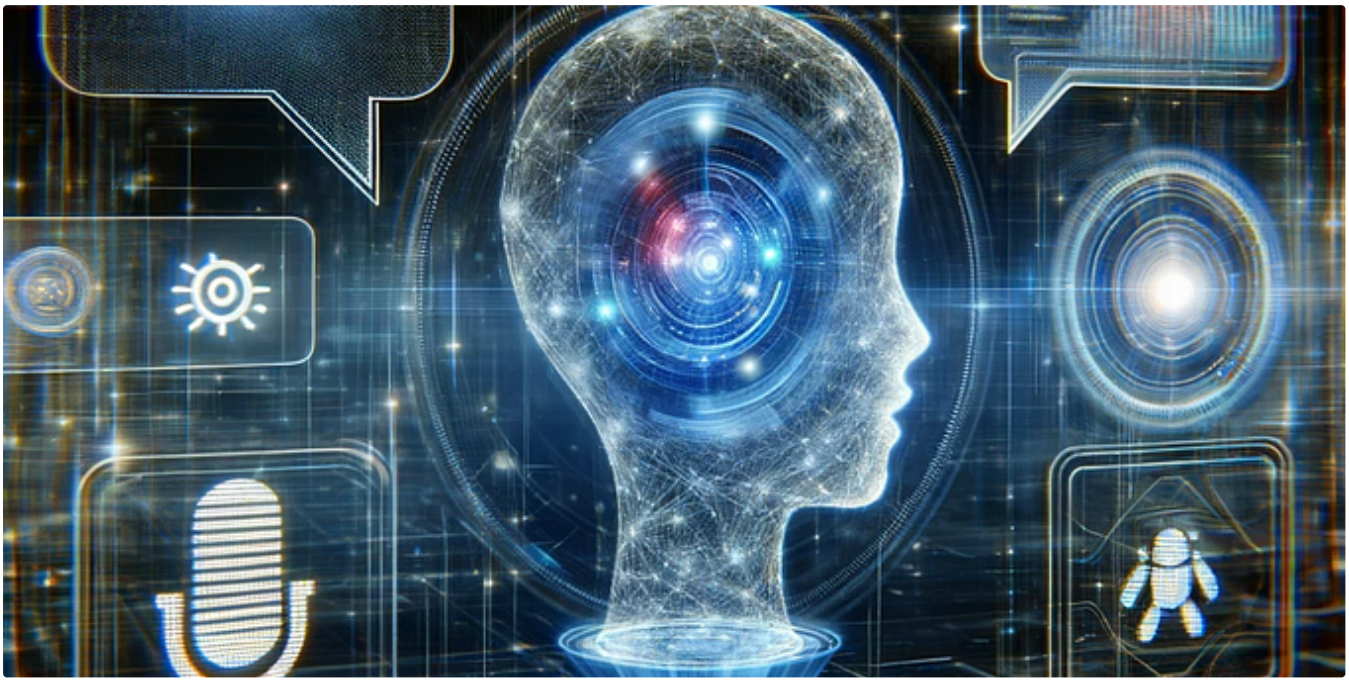
Follow

Written by Harshad Suryawanshi

42 Followers · Writer for LlamaIndex Blog

IIM Trichy | Driving Analytics Product Content at MSCI | Gen-AI Enthusiast | Large Language Models (LLM) Explorer

More from Harshad Suryawanshi and LlamaIndex Blog



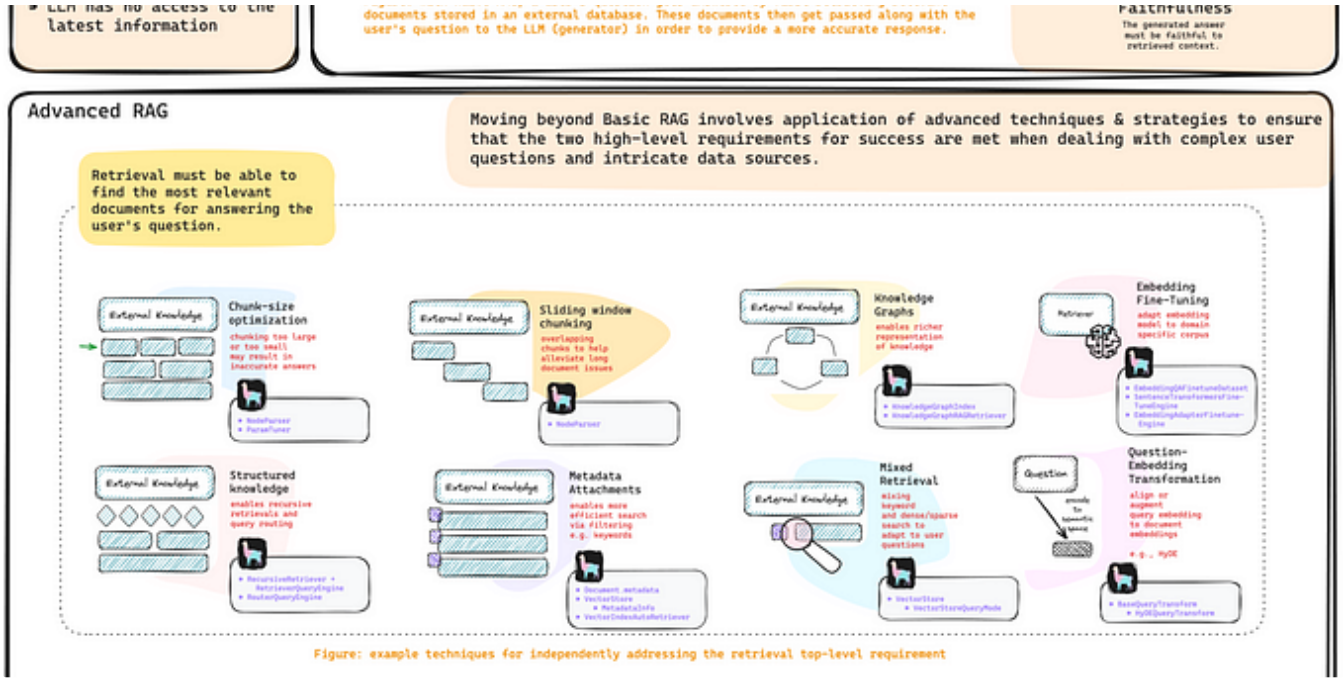
Harshad Suryawanshi in LlamaIndex Blog

AI Voice Assistant: Enhancing Accessibility in AI with LlamaIndex and GPT3.5

Introduction

5 min read · Jan 14

182 1



Andrei in LlamaIndex Blog

A Cheat Sheet and Some Recipes For Building Advanced RAG

It's the start of a new year and perhaps you're looking to break into the RAG scene by building your very first RAG system. Or, maybe...

7 min read · Jan 5

1.3K 5

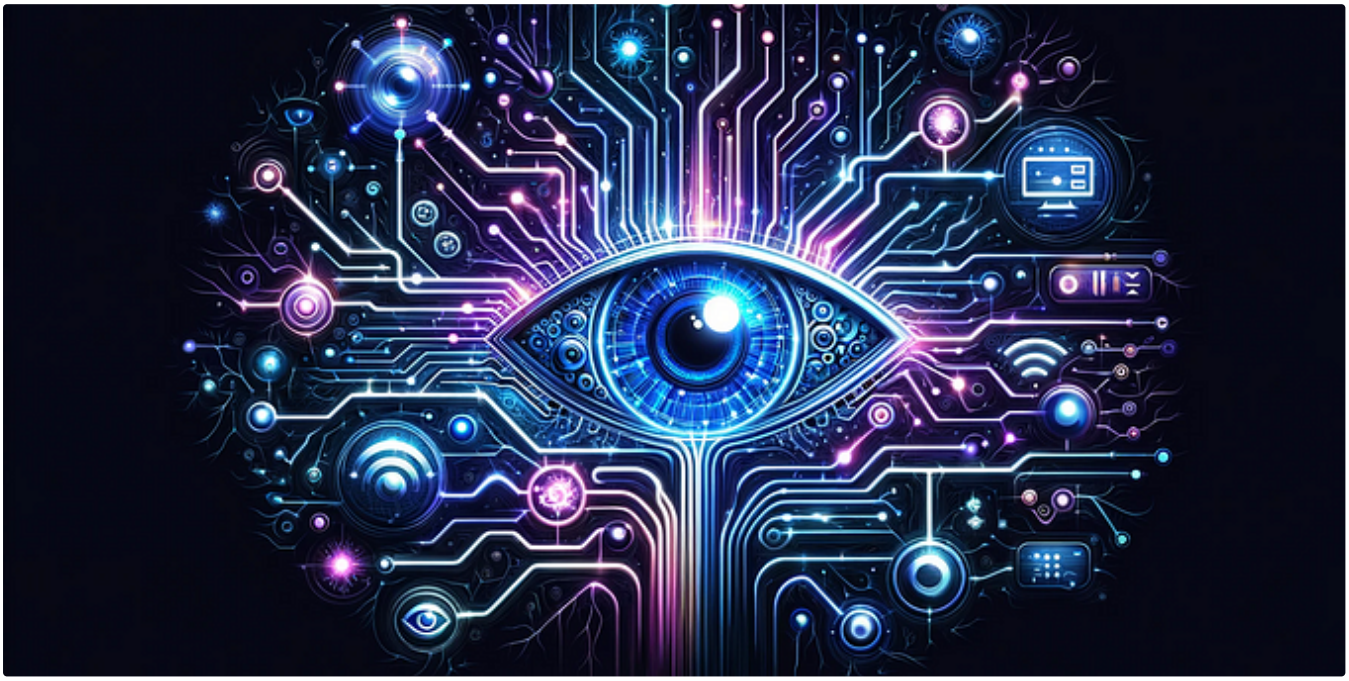
LlamaIndex in LlamaIndex Blog

Running Mixtral 8×7 locally with LlamaIndex and Ollama

You may have heard the fuss about the latest release from European AI powerhouse Mistral AI: it's called Mixtral 8×7b, a "mixture of...

6 min read · Dec 22, 2023

701 8



Harshad Suryawanshi in LlamaIndex Blog

Building My Own ChatGPT Vision with PaLM, KOSMOS-2 and LlamaIndex

In the ever-evolving landscape of AI, OpenAI's ChatGPT with vision capabilities has opened a new chapter. It's an exciting time for...

5 min read · Nov 8, 2023

37

See all from Harshad Suryawanshi

See all from LlamaIndex Blog

Recommended from Medium

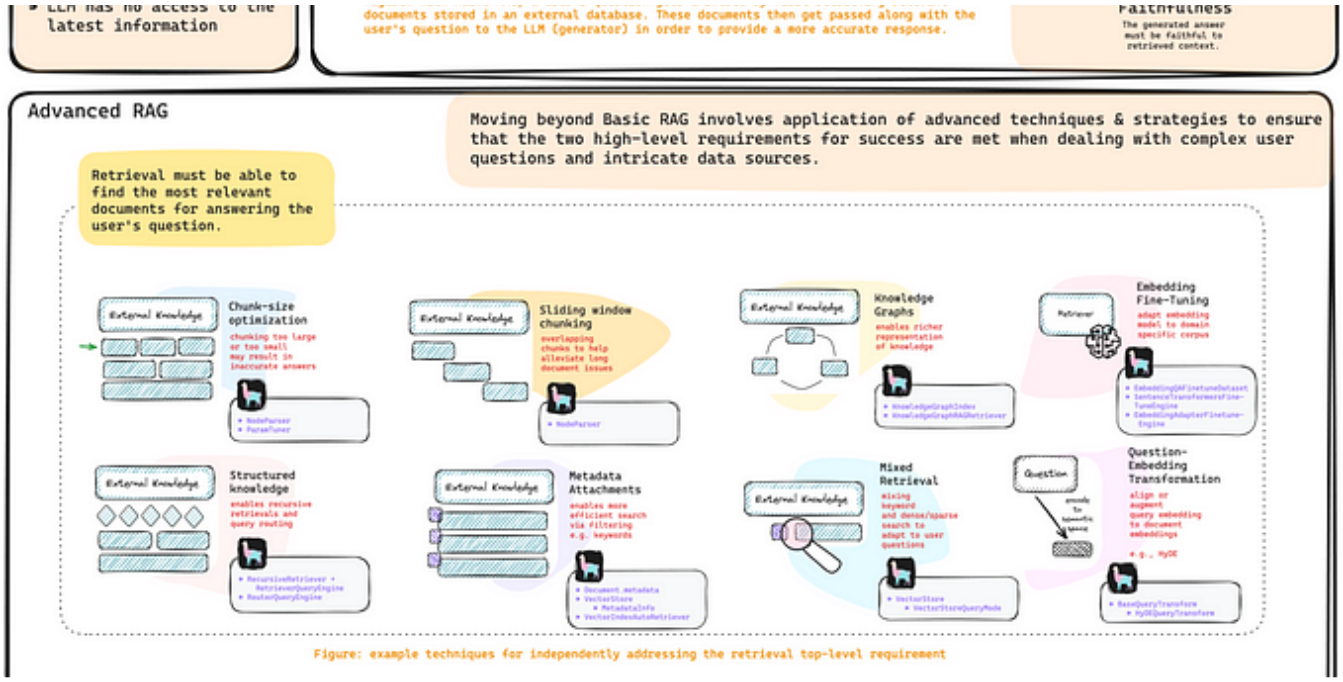
IVAN ILIN in Towards AI

Advanced RAG Techniques: an Illustrated Overview

A comprehensive study of the advanced retrieval augmented generation techniques and algorithms, systemising various approaches. The article...

19 min read · Dec 17, 2023

4.1K 24



Andrei in LlamaIndex Blog

A Cheat Sheet and Some Recipes For Building Advanced RAG

It's the start of a new year and perhaps you're looking to break into the RAG scene by building your very first RAG system. Or, maybe...

7 min read · Jan 5

1.3K 5

Lists

Natural Language Processing

1132 stories · 602 saves

AI Regulation

6 stories · 296 saves

data science and AI

39 stories · 54 saves

Coding & Development

11 stories · 408 saves

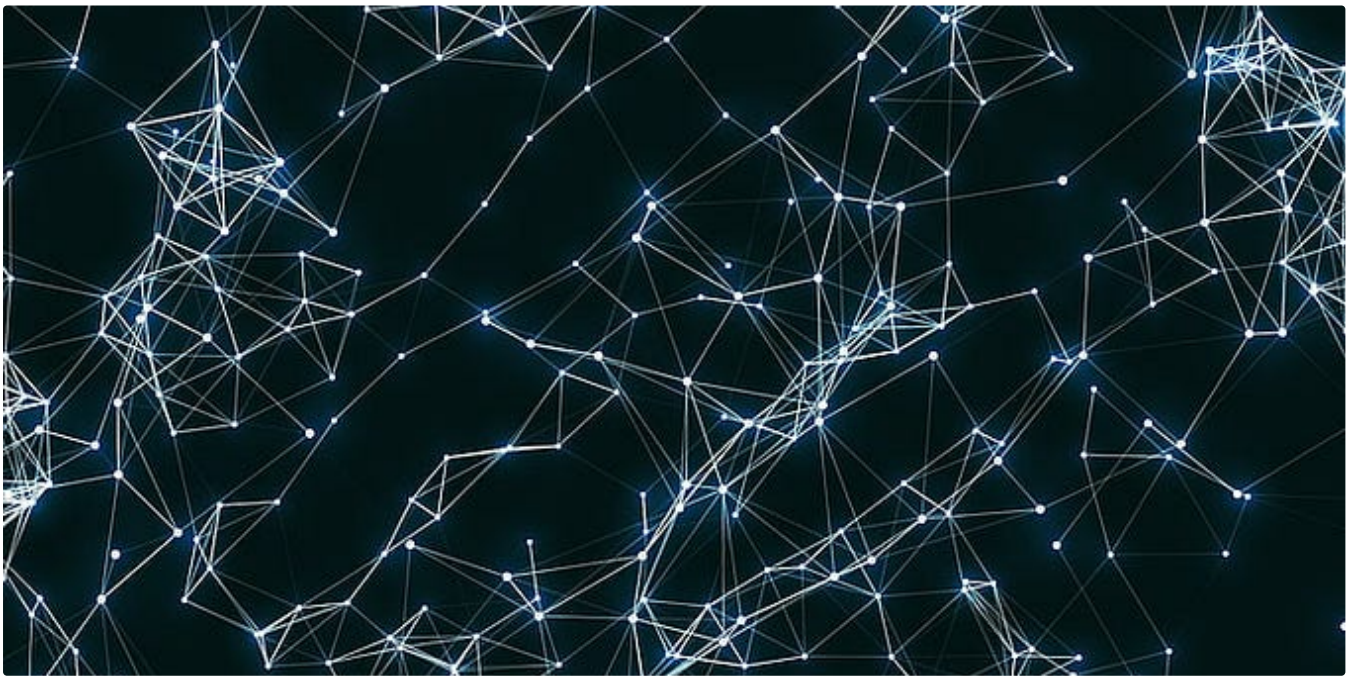
Ingrid Stevens

Extract Structured Data from Unstructured Text using LLMs

Using LangChain's create_extraction_chain and PydanticOutputParser

5 min read · Jan 1

355 2



Ahmedabdullah in Tensor Labs

Taking your RAG pipelines to a next level ! LangGraphs

Discover power of LangGraph: Revolutionize AI with intelligent, iterative RAG pipelines that think, adapt, and deliver precision solutions

5 min read · Jan 19

318 2

Ming

Comparing LangChain and LlamaIndex with 4 tasks

LangChain v.s. LlamaIndex — How do they compare? Show me the code!

10 min read · Jan 11

490 3

Akash Mathur

Advanced RAG: Query Augmentation for Next-Level Search using LlamaIndex 🦙

Using Open Source LLM Zephyr-7b-alpha and BGE Embeddings bge-large-en-v1.5

13 min read · Jan 18

583 3