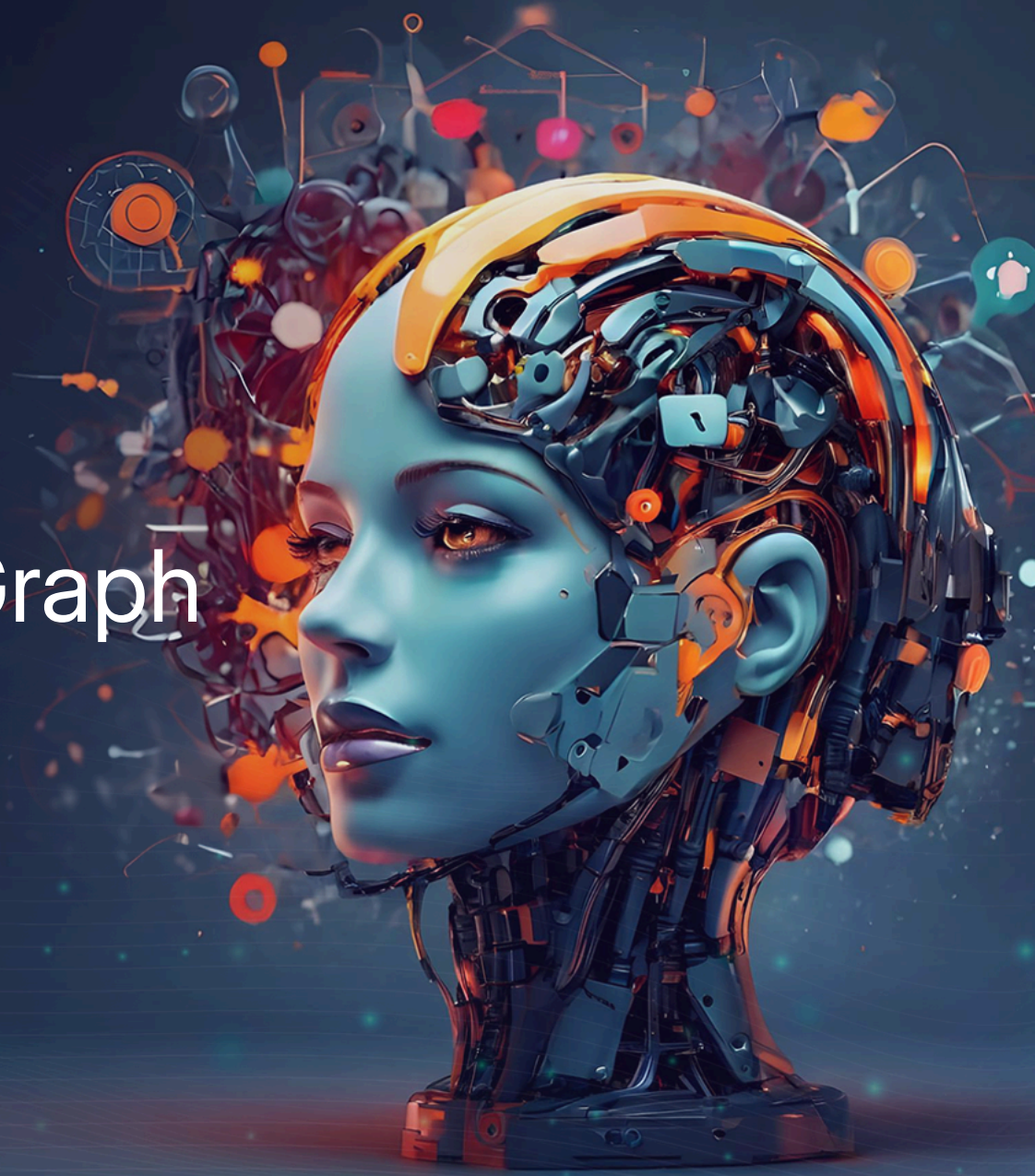
Analytics Vidhya

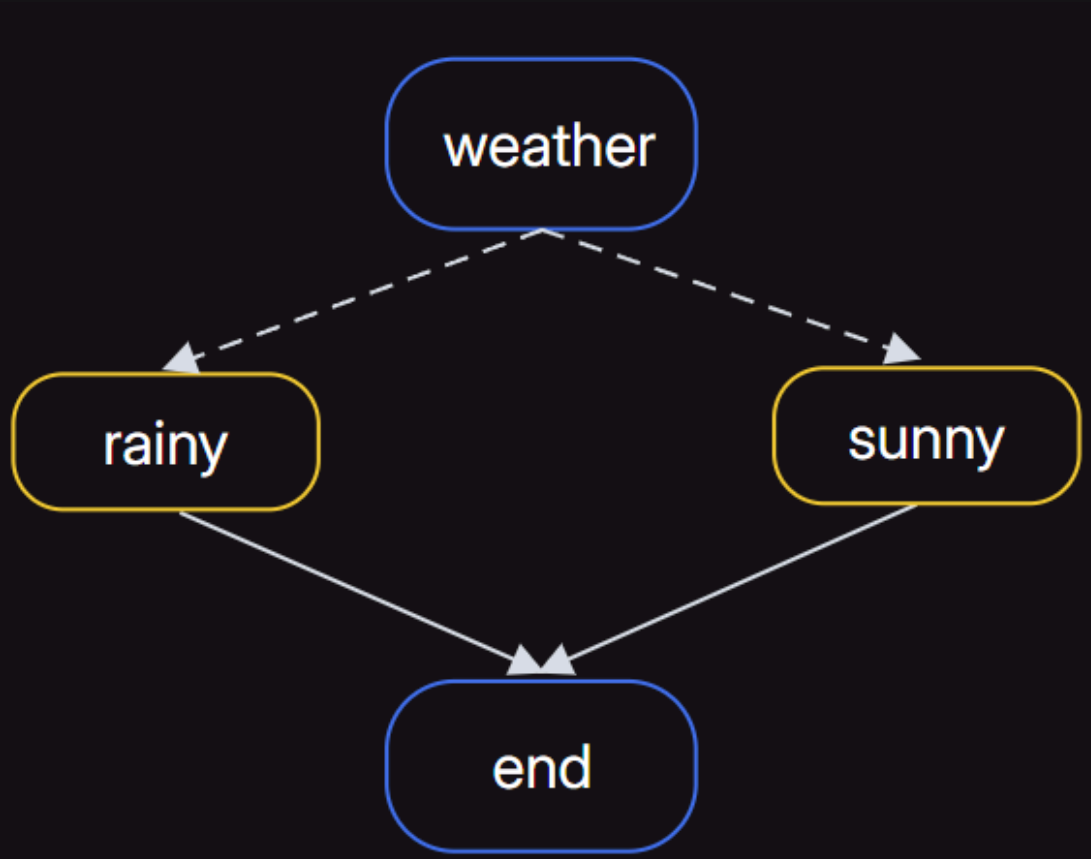
Conditional Routing in LangGraph



Dipanjan Sarkar
Head of Community & Principal AI Scientist at Analytics Vidhya
Google Developer Expert - ML & Cloud Champion Innovator
Published Author

Conditional Routing in LangGraph

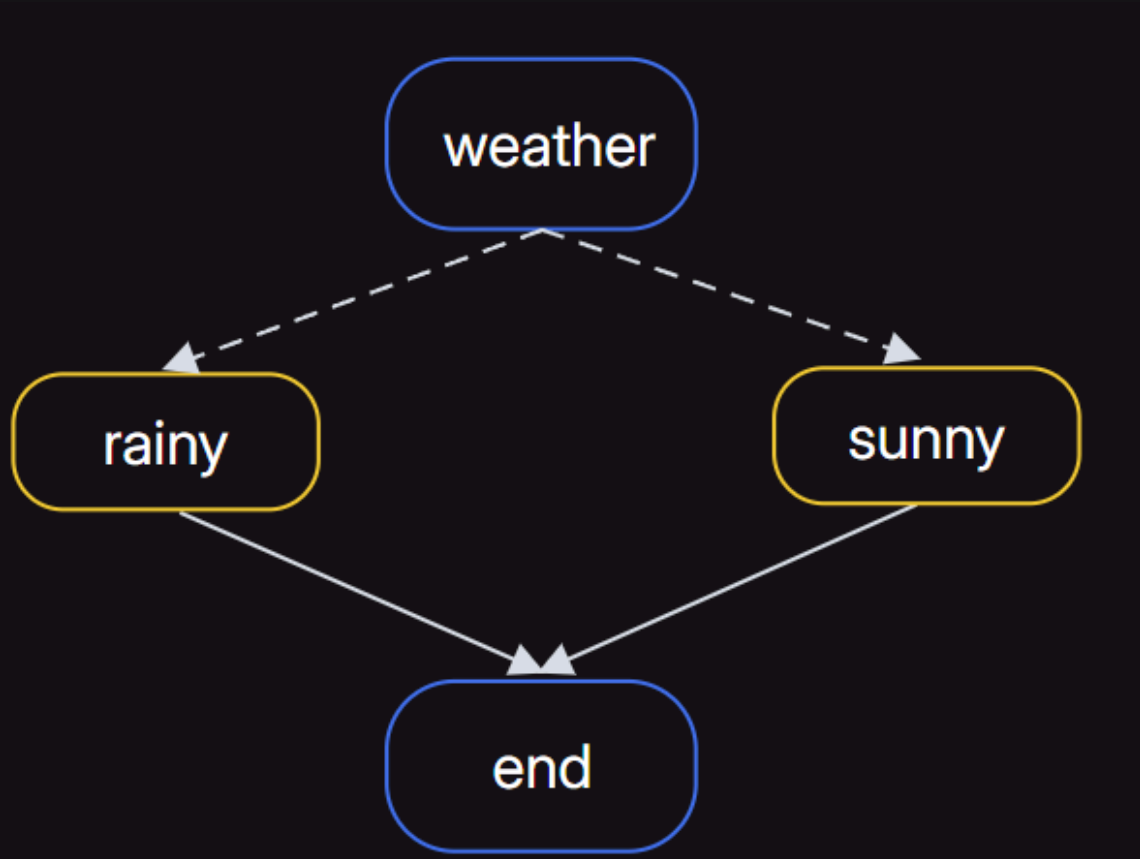
- In LangGraph, conditional routing allows the dynamic determination of the next node(s) to execute based on the current state.
- This is achieved using conditional edges, which evaluate a function using certain defined logic, to decide the subsequent path in the graph.
- This mechanism enables the creation of adaptable workflows that respond to varying conditions during execution.



Conditional Routing in LangGraph

Implementing Conditional Edges

- To implement conditional routing, you define a routing function that examines the current state and returns the name of the next node(s) to execute.
- This function is then associated with a node using the add_conditional_edges method.



Conditional Routing Example in LangGraph

State Definition

- The State schema includes a weather field, that specifies the weather condition (rainy or sunny)

```python
from typing import TypedDict
from langgraph.graph import StateGraph, START, END

# Define the state structure
class State(TypedDict):
    weather: str

# Define node functions
def weather_node(state: State) -> State:
    print("Checking the weather...")
    return state  # Just pass the state along

def rainy_node(state: State) -> State:
    print("It's rainy. Take an umbrella!")
    return state

def sunny_node(state: State) -> State:
    print("It's sunny. Wear sunglasses!")
    return state

# Define the routing function
def weather_routing(state: State) -> str:
    if state['weather'] == 'rainy':
        return "rainy"
    elif state['weather'] == 'sunny':
        return "sunny"
    else:
        raise ValueError("Invalid weather condition!")
```

Conditional Routing Example in LangGraph

Node Functions

- weather_node: A decision point where the weather is checked
- rainy_node: Prints a message for rainy weather
- sunny_node: Prints a message for sunny weather

```python
from typing import TypedDict
from langgraph.graph import StateGraph, START, END

# Define the state structure
class State(TypedDict):
    weather: str

# Define node functions
def weather_node(state: State) -> State:
    print("Checking the weather...")
    return state  # Just pass the state along

def rainy_node(state: State) -> State:
    print("It's rainy. Take an umbrella!")
    return state

def sunny_node(state: State) -> State:
    print("It's sunny. Wear sunglasses!")
    return state

# Define the routing function
def weather_routing(state: State) -> str:
    if state['weather'] == 'rainy':
        return "rainy"
    elif state['weather'] == 'sunny':
        return "sunny"
    else:
        raise ValueError("Invalid weather condition!")
```

Conditional Routing Example in LangGraph

Routing Function

- weather_routing: Routes to either rainy or sunny nodes based on the value of weather in the state.

```python
from typing import TypedDict
from langgraph.graph import StateGraph, START, END

# Define the state structure
class State(TypedDict):
    weather: str

# Define node functions
def weather_node(state: State) -> State:
    print("Checking the weather...")
    return state  # Just pass the state along

def rainy_node(state: State) -> State:
    print("It's rainy. Take an umbrella!")
    return state

def sunny_node(state: State) -> State:
    print("It's sunny. Wear sunglasses!")
    return state

# Define the routing function
def weather_routing(state: State) -> str:
    if state['weather'] == 'rainy':
        return "rainy"
    elif state['weather'] == 'sunny':
        return "sunny"
    else:
        raise ValueError("Invalid weather condition!")
```

Conditional Routing Example in LangGraph

Graph Construction

- The graph begins at START, passes through weather_node, and uses conditional edges to route to rainy or sunny
- Both rainy and sunny nodes are connected to END

```python
# Initialize the StateGraph
graph_builder = StateGraph(State)

# Add nodes to the graph
graph_builder.add_node("weather_node", weather_node)
graph_builder.add_node("rainy", rainy_node)
graph_builder.add_node("sunny", sunny_node)

# Define edges
graph_builder.add_edge(START, "weather_node")
graph_builder.add_conditional_edges("weather_node",
                                    weather_routing,
                                    ["rainy", "sunny"])
graph_builder.add_edge("rainy", END)
graph_builder.add_edge("sunny", END)

# Compile the graph
graph = graph_builder.compile()

print("\n--- Sunny Condition ---")
initial_state_sunny = {"weather": "sunny"}
result_sunny = graph.invoke(initial_state_sunny)

## OUTPUT
--- Sunny Condition ---
Checking the weather...
It's sunny. Wear sunglasses!
```

Conditional Routing Example in LangGraph

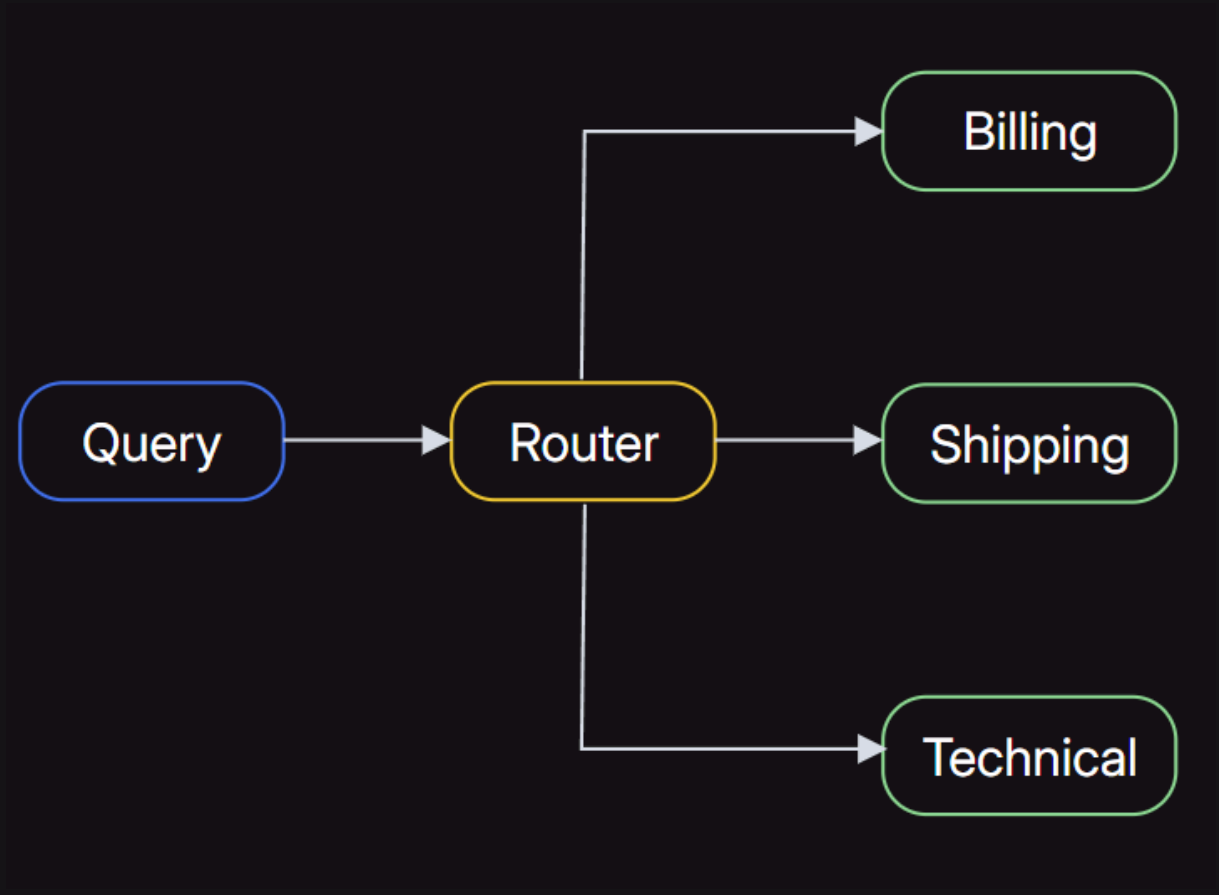
Execution

- The graph executes and goes to the relevant node based on the weather input, demonstrating dynamic routing based on the state.

```python
# Initialize the StateGraph
graph_builder = StateGraph(State)

# Add nodes to the graph
graph_builder.add_node("weather_node", weather_node)
graph_builder.add_node("rainy", rainy_node)
graph_builder.add_node("sunny", sunny_node)

# Define edges
graph_builder.add_edge(START, "weather_node")
graph_builder.add_conditional_edges("weather_node",
                                    weather_routing,
                                    ["rainy", "sunny"])
graph_builder.add_edge("rainy", END)
graph_builder.add_edge("sunny", END)

# Compile the graph
graph = graph_builder.compile()

print("\n--- Sunny Condition ---")
initial_state_sunny = {"weather": "sunny"}
result_sunny = graph.invoke(initial_state_sunny)

## OUTPUT
--- Sunny Condition ---
Checking the weather...
It's sunny. Wear sunglasses!
```

Router Agents Brief

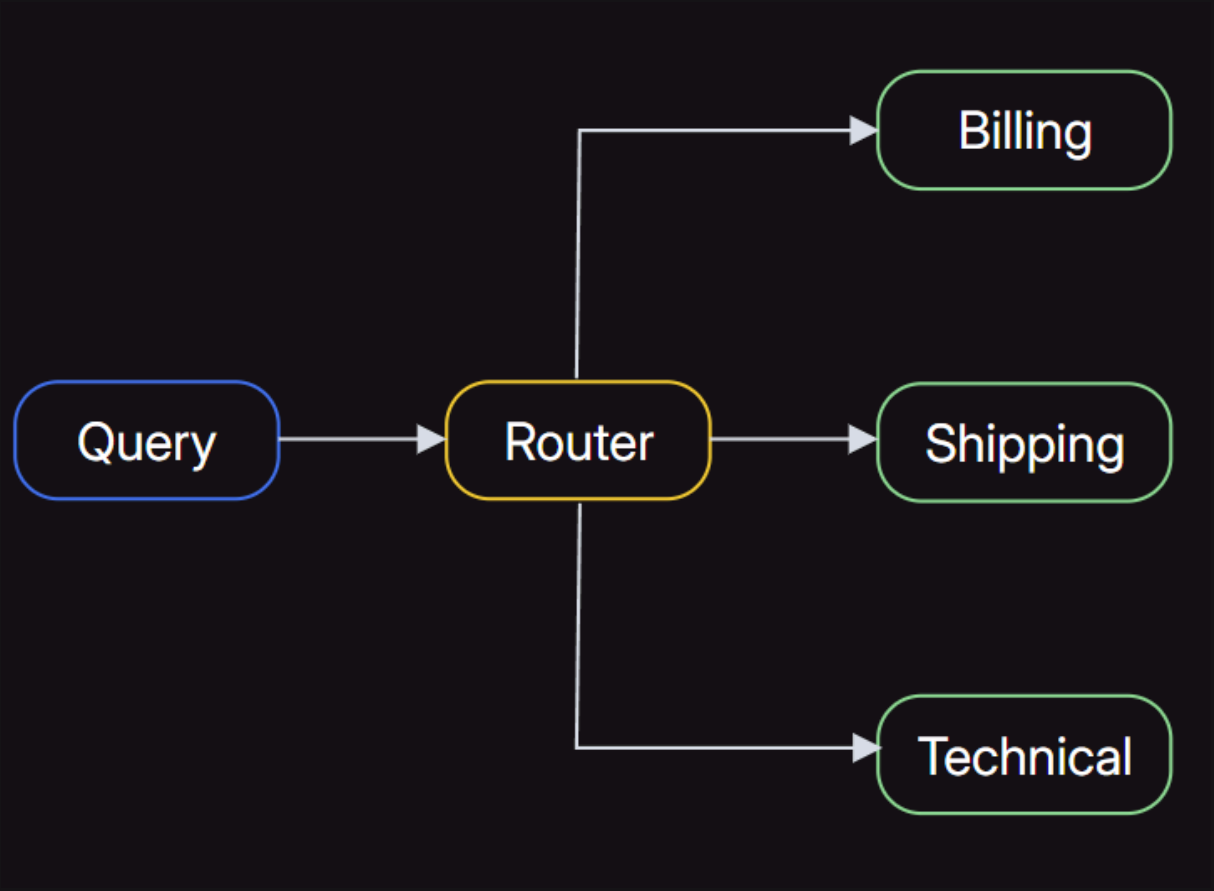
- Router Agents serve as decision-making components that analyze user inputs and direct them to the most appropriate nodes for further processing.
- This routing mechanism is powered by conditional edges and enables the construction of dynamic, adaptable workflows.



Router Agents Brief

Key Features of Router Agents

- Dynamic Query Classification
  - Router agents assess incoming queries and categorize them based on their content, facilitating targeted processing by specialized nodes or sub-agents.
- Conditional Routing
  - By implementing conditional edges, router agents determine the subsequent node or agent to handle a task, allowing for flexible and context-aware workflow management.
- Integration with Multi-Agent Systems
  - Router agents are integral to multi-agent architectures, coordinating interactions among various agents to efficiently manage complex tasks.

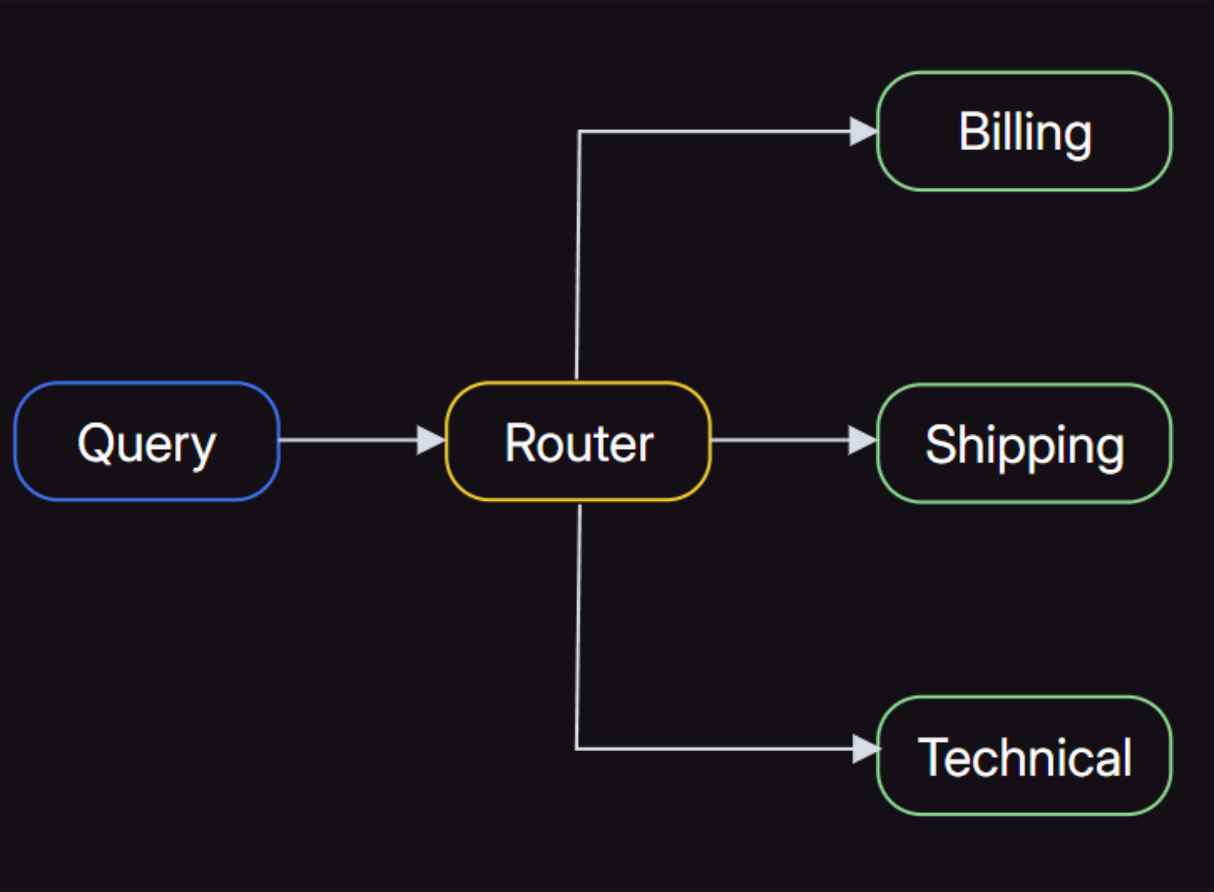


Analytics Vidhya

Router Agents Brief

Example Workflow

- User Query Reception
  - The router agent receives a user's query.
- Query Analysis
  - It analyzes the query to determine its nature and requirements.
- Routing Decision
  - Based on the analysis, the router agent directs the query to a specialized node function, agent, or tool equipped to handle that specific type of request.



Thanks