Chapter 2: Routing

Routing Pattern Overview

While sequential processing via prompt chaining is a foundational technique for executing deterministic, linear workflows with language models, its applicability is limited in scenarios requiring adaptive responses. Real-world agentic systems must often arbitrate between multiple potential actions based on contingent factors, such as the state of the environment, user input, or the outcome of a preceding operation. This capacity for dynamic decision-making, which governs the flow of control to different specialized functions, tools, or sub-processes, is achieved through a mechanism known as routing.

Routing introduces conditional logic into an agent's operational framework, enabling a shift from a fixed execution path to a model where the agent dynamically evaluates specific criteria to select from a set of possible subsequent actions. This allows for more flexible and context-aware system behavior.

For instance, an agent designed for customer inquiries, when equipped with a routing function, can first classify an incoming query to determine the user's intent. Based on this classification, it can then direct the query to a specialized agent for direct question-answering, a database retrieval tool for account information, or an escalation procedure for complex issues, rather than defaulting to a single, predetermined response pathway. Therefore, a more sophisticated agent using routing could:

1. Analyze the user's query.
2. **Route** the query based on its *intent*:
    - If the intent is "check order status", route to a sub-agent or tool chain that interacts with the order database.
    - If the intent is "product information", route to a sub-agent or chain that searches the product catalog.
    - If the intent is "technical support", route to a different chain that accesses troubleshooting guides or escalates to a human.
    - If the intent is unclear, route to a clarification sub-agent or prompt chain.

The core component of the Routing pattern is a mechanism that performs the evaluation and directs the flow. This mechanism can be implemented in several ways:

- **LLM-based Routing:** The language model itself can be prompted to analyze the input and output a specific identifier or instruction that indicates the next step or destination. For example, a prompt might ask the LLM to "Analyze the following user query and output only the category: 'Order Status', 'Product Info', 'Technical

Support', or 'Other'." The agentic system then reads this output and directs the workflow accordingly.
- **Embedding-based Routing:** The input query can be converted into a vector embedding (see RAG, Chapter 14). This embedding is then compared to embeddings representing different routes or capabilities. The query is routed to the route whose embedding is most similar. This is useful for semantic routing, where the decision is based on the meaning of the input rather than just keywords.
- **Rule-based Routing:** This involves using predefined rules or logic (e.g., if-else statements, switch cases) based on keywords, patterns, or structured data extracted from the input. This can be faster and more deterministic than LLM-based routing, but is less flexible for handling nuanced or novel inputs.
- **Machine Learning Model-Based Routing**: it employs a discriminative model, such as a classifier, that has been specifically trained on a small corpus of labeled data to perform a routing task. While it shares conceptual similarities with embedding-based methods, its key characteristic is the supervised fine-tuning process, which adjusts the model's parameters to create a specialized routing function. This technique is distinct from LLM-based routing because the decision-making component is not a generative model executing a prompt at inference time. Instead, the routing logic is encoded within the fine-tuned model's learned weights. While LLMs may be used in a pre-processing step to generate synthetic data for augmenting the training set, they are not involved in the real-time routing decision itself.

Routing mechanisms can be implemented at multiple junctures within an agent's operational cycle. They can be applied at the outset to classify a primary task, at intermediate points within a processing chain to determine a subsequent action, or during a subroutine to select the most appropriate tool from a given set.

Computational frameworks such as LangChain, LangGraph, and Google's Agent Developer Kit (ADK) provide explicit constructs for defining and managing such conditional logic. With its state-based graph architecture, LangGraph is particularly well-suited for complex routing scenarios where decisions are contingent upon the accumulated state of the entire system. Similarly, Google's ADK provides foundational components for structuring an agent's capabilities and interaction models, which serve as the basis for implementing routing logic. Within the execution environments provided by these frameworks, developers define the possible operational paths and the functions or model-based evaluations that dictate the transitions between nodes in the computational graph.