Reading: Structured Outputs for Tool Calling

Estimated time: 5 mins

Learning Objectives

By the end of this reading, you will be able to:

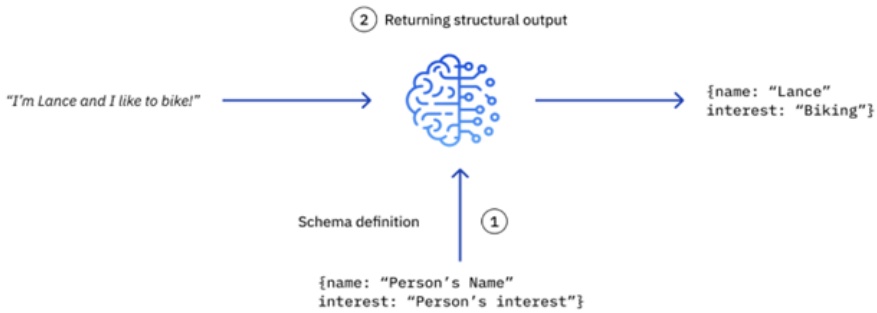
- Define structured output schemas using JSON-like formats and Pydantic models to enforce data consistency in LLM-generated responses.
- Implement tool calling and JSON mode to generate and validate structured outputs for use in databases, APIs, and application workflows.

This reading explores the use of structured outputs in large language model (LLM) applications, crucial for producing consistent, schema-compliant responses. Learners will discover how to implement output schemas using Pydantic models, tool calling, and JSON mode to support reliable database integration, API workflows, and data validation across various applications.

While many LLM applications respond to users in natural language, there are scenarios where we need models to output in a structured format. Structured outputs allow you to:

- Store model responses in databases with consistent formats
- Process outputs programmatically in your application
- Ensure responses include specific fields in a predictable structure
- Validate output against a defined schema

The process involves two main steps, as illustrated in the image:



1. **Schema Definition:** Defining the structure that we want the output to have.
2. **Returning structured output:** Having the model generate output conforming to the schema.

For example, given an input like *"I'm Lance and I like to bike,"* we can define a schema with fields for *"name"* and *"interest"*. The LLM processes this input against the schema and returns a structured output: {name: "Lance", interest: "Biking"}.

Schema definition

The schema defines the structure your model's output should follow. In LangChain, you have several options for defining schemas:

1. **JSON like structures:** The simplest format is a JSON-like structure, represented in Python as a dictionary or list.

2. **Pydantic models:** Pydantic is particularly useful for defining structured output schemas because it offers type hints and validation. Pydantic models offer several advantages as well, such as:

    - Type validation
    - Clear field descriptions
    - Built-in documentation
    - Integration with LangChain's tooling

Returning structured output

There are two main approaches to ensure models return structured outputs:

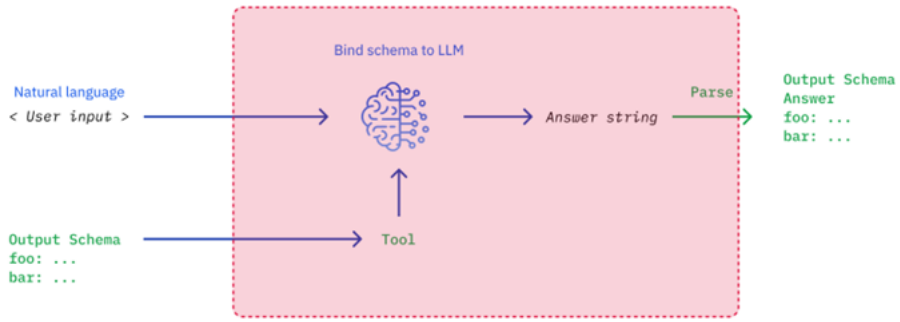
1. Tool calling

- Tool calling involves binding a schema to a model as a tool. When invoked, the model calls this tool and ensures its response conforms to the tool's schema. The process involves:
    - Creating a model
    - Binding the ResponseFormatter schema as a tool to the model
    - Invoking the model with a user query
    - Extracting the tool call arguments as a dictionary
    - Optionally, parsing the dictionary into a pydantic object

2. JSON mode

- Some model providers support JSON mode, which enforces the model to produce conforming JSON output. This approach:
    - Requires using a model that supports JSON mode
    - Uses the with_structured_output method with the "json_mode" parameter
    - Returns results already formatted as a Python dictionary

Structured Output Method: with_structured_output()



As shown in the image, LangChain provides a helper method with_structured_output() that streamlines the entire process:

1. The output schema is provided as a tool to the LLM
2. User input is processed by the LLM with this schema in mind
3. The LLM generates an answer string
4. This output is parsed into the structured schema format

This workflow handles several challenges automatically:

- Binding the schema to the model as a tool
- Instructing the model to always use the tool
- Parsing the output back to the specified schema

Use cases for structured outputs

- **Database storage:** Format responses for direct storage in databases
- **API integration:** Structure outputs to match API requirements
- **UI components:** Format data for display in specific UI elements
- **Multi-step workflows:** Break down complex tasks into structured steps
- **Data extraction:** Extract specific information from unstructured text

Summary

In this reading, you learned that:

1. **Structured outputs enable precision:** LLMs can produce responses in predictable formats, essential for database storage, API integration, and automating workflows.
2. **Schemas define output expectations:** Using JSON-like structures or Pydantic models helps enforce required fields and validate response data programmatically.
3. **Tool calling for schema enforcement:** Binding a schema as a "tool" to a model ensures outputs conform exactly to predefined formats, enabling robust application logic.
4. **JSON mode for reliable outputs:** Models supporting JSON mode can generate structured outputs automatically, reducing the need for post-processing.
5. **LangChain's with_structured_output() simplifies the process:** This helper method streamlines schema binding, model invocation, and output parsing, making structured responses easier to implement.

Author(s)

IBM Skills Network Team