Structured Output in LangChain

22 May 2025 22:24

Structured Output

28 February 2025 00:13

In LangChain, structured output refers to the practice of having language models return responses in a well-defined data format (for example, JSON), rather than free-form text. This makes the model output easier to parse and work with programmatically.

[Prompt] - *Can you create a one-day travel itinerary for Paris?*

[LLM's Unstructured Response]

Here's a suggested itinerary: *Morning: Visit the Eiffel Tower.*

Afternoon: Walk through the Louvre Museum.

Evening: Enjoy dinner at a Seine riverside café.

[JSON enforced output]

```
[
  {"time": "Morning", "activity": "Visit the Eiffel Tower"},
  {"time": "Afternoon", "activity": "Walk through the Louvre Museum"},
  {"time": "Evening", "activity": "Enjoy dinner at a Seine riverside café"}
]
```

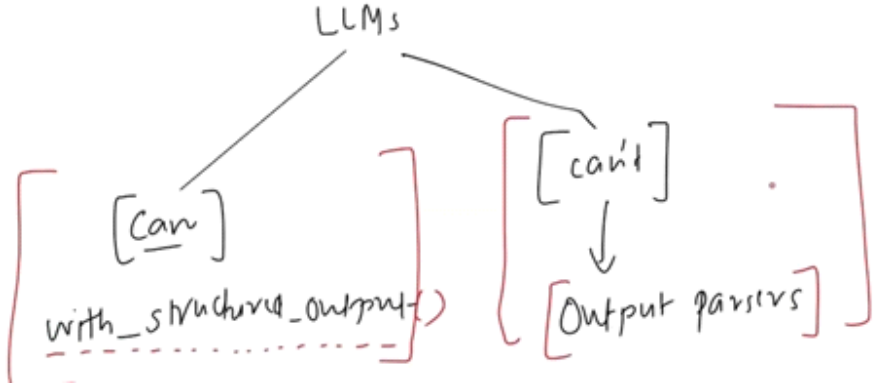
Why do we need Structured Output

28 February 2025 00:13

- Data Extraction
- API building
- Agents

Ways to get Structured Output

28 February 2025 00:14



TypedDict

01 March 2025 12:59

TypedDict is a way to define a dictionary in Python where you specify what keys and values should exist. It helps ensure that your dictionary follows a specific structure.

Why use TypedDict?

- It tells Python what keys are required and what types of values they should have.
- It does not validate data at runtime (it just helps with type hints for better coding).

-> simple TypedDict
-> Annotated TypedDict
-> Literal
-> More complex -> with pros and cons

Pydantic

01 March 2025 12:59

Pydantic is a data validation and data parsing library for Python. It ensures that the data you work with is correct, structured, and type-safe.

Basic example
Default values
Optional fields
Coerce
Builtin validation
Field Function -> default values, constraints, description, regex expressions
Returns pydantic object -> convert to json/dict

When to use what?

01 March 2025 12:59

✅ **Use `TypedDict` if:**

- You **only need type hints** (basic structure enforcement).
- You **don't need validation** (e.g., checking numbers are positive).
- You **trust the LLM** to return correct data.

✅ **Use `Pydantic` if:**

- You need **data validation** (e.g., sentiment must be `"positive"`, `"neutral"`, or `"negative"`).
- You need **default values** if the LLM **misses fields**.
- You want **automatic type conversion** (e.g., `"100"` → `100`).

✅ **Use `JSON Schema` if:**

- You **don't want to import extra Python libraries** (Pydantic).
- You need **validation but don't need Python objects**.
- You want to define structure **in a standard JSON format**.

🚀 **When to Use What?**

Feature	TypedDict ✅	Pydantic 🚀	JSON Schema 🌐
Basic structure	✅	✅	✅
Type enforcement	✅	✅	✅
Data validation	❌	✅	✅
Default values	❌	✅	❌
Automatic conversion	❌	✅	❌
Cross-language compatibility	❌	❌	✅

Output Parsers in LangChain

Output Parsers

06 March 2025 16:29

Output Parsers in LangChain help convert raw LLM responses into structured formats like JSON, CSV, Pydantic models, and more. They ensure consistency, validation, and ease of use in applications.

StrOutputParser

06 March 2025 16:29

The StrOutputParser is the simplest output parser in LangChain. It is used to parse the output of a Language Model (LLM) and return it as a plain string.

result.content

```
content='A black hole is a region in space where gravity is so strong that nothing, not e
ven light, can escape its pull. It is formed when a massive star collapses upon itself.'
additional_kwargs={'refusal': None} response_metadata={'token_usage': {'completion_tokens
': 37, 'prompt_tokens': 15, 'total_tokens': 52, 'completion_tokens_details': {'accepted_p
rediction_tokens': 0, 'audio_tokens': 0, 'reasoning_tokens': 0, 'rejected_prediction_toke
ns': 0}, 'prompt_tokens_details': {'audio_tokens': 0, 'cached_tokens': 0}}, 'model_name':
 'gpt-3.5-turbo-0125', 'system_fingerprint': None, 'finish_reason': 'stop', 'logprobs': N
one} id='run-a7b90203-58f8-47c5-a01b-01184b6aec14-0' usage_metadata={'input_tokens': 15,
'output_tokens': 37, 'total_tokens': 52, 'input_token_details': {'audio': 0, 'cache_read'
: 0}, 'output_token_details': {'audio': 0, 'reasoning': 0}}
```

StructuredOutputParser

06 March 2025 16:29

StructuredOutputParser is an output parser in LangChain that helps extract structured JSON data from LLM responses based on predefined field schemas.

It works by defining a list of fields (ResponseSchema) that the model should return, ensuring the output follows a structured format.

PydanticOutputParser

06 March 2025 16:30

🔹 What is `PydanticOutputParser` in LangChain?

`PydanticOutputParser` is a **structured output parser** in LangChain that uses **Pydantic models** to enforce schema validation when processing LLM responses.

🚀 Why Use `PydanticOutputParser`?

- ✅ **Strict Schema Enforcement** → Ensures that LLM responses follow a well-defined structure.
- ✅ **Type Safety** → Automatically converts LLM outputs into Python objects.
- ✅ **Easy Validation** → Uses Pydantic's built-in validation to catch incorrect or missing data.
- ✅ **Seamless Integration** → Works well with other LangChain components.