Output Parsers

MUKESH KUMAR

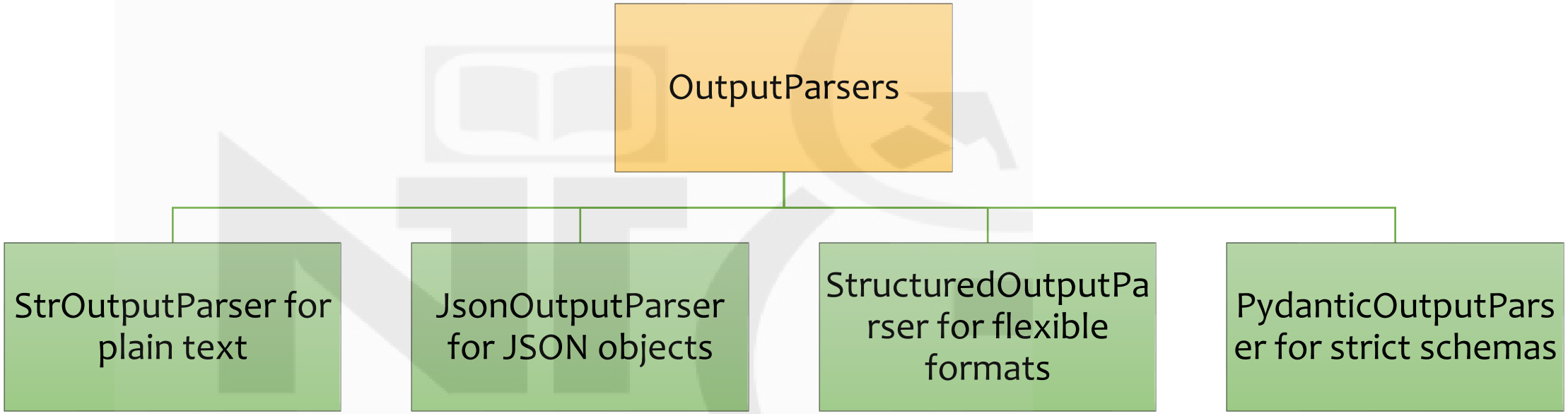
Introduction to Output Parsers

What are Output Parsers?

- Tools in LangChain that convert raw LLM output into structured formats
- Help ensure data is clean, predictable, and usable by other systems
- Bridge between free-form text and structured application data

Why Use Output Parsers?

- LLMs produce human-like natural language
- Applications require structured formats (JSON, lists, models)
- Output Parsers prevent ambiguity and parsing errors
- Enable LLM integration with APIs, databases, UIs



https://python.langchain.com/docs/concepts/output_parsers/

StringOutputParser

- **Purpose:**
- Basic parser that returns the raw text output from the LLM
- **Use Case:**
- Simple text generation tasks
- **Example:**

```
from langchain.output_parsers import StrOutputParser
parser = StrOutputParser()
result = parser.parse("This is a generated string.")
```

Why not just use result.content instead of stringoutput parser

- Result.content also gives the same result as stringouput parser
- However , its not compatible with chains

JsonOutputParser

- **Purpose:**
- Parses output from LLM as strict JSON
- **Use Case:**
- When downstream applications require exact key-value format
- **Example:**

```
from langchain.output_parsers import JsonOutputParser
parser = JsonOutputParser()
result = parser.parse('{"name": "Alice", "age": 25}')
```

Drawback

- Jsonouput parser cannot enforce schema i.e. we cannot enforce a predefined structure of the json output its decided by llm

- There is where Structuredouput parser comes in.

StructuredOutputParser

- **Purpose:**
- Guide the LLM to produce structured outputs via formatting instructions
- Often used with schema or descriptions
- **Use Case:**
- Semi-structured generation with user-defined format
- **Example:**

```
from langchain.output_parsers import StructuredOutputParser
from langchain.output_parsers.schema import ResponseSchema
schema = [ResponseSchema(name="title", description="Title of the book")]
parser = StructuredOutputParser.from_response_schemas(schema)
```

Drawbacks

- No data validation
- LLM may return different datatype which will go unchecked
- This is where pydantic ouput parser comes in

PydanticOutputParser

- **Purpose:**
- Uses Pydantic models to enforce structure and validate output
- **Use Case:**
- When data needs strong typing and validation
- **Example:**

```python
from pydantic import BaseModel
from langchain.output_parsers import PydanticOutputParser

class Movie(BaseModel):
    title: str
    rating: float

parser = PydanticOutputParser(pydantic_object=Movie)
```

Advantages of PydanticOutputParser

- Ensures LLM output matches a defined schema with automatic validation.
- Converts output into typed Python objects for easy access and fewer bugs.
- Provides clear error messages if output format is wrong.
- Generates prompt instructions to guide the LLM’s response format.
- Improves code maintainability by enforcing a clear data contract.

Summary

- Output Parsers convert raw LLM text to structured data
- LangChain provides various parsers for different needs:
  - StrOutputParser for plain text
  - JsonOutputParser for JSON objects
  - StructuredOutputParser for flexible formats
  - PydanticOutputParser for strict schemas
- Crucial for integrating LLMs with production software