Day93_GenAI_Financial_Data_Extraction

September 30, 2025

GenAI: Financial Data Extraction

Turn the theory from Day 90 into practice: **build a working financial data extractor** using LangChain, Prompt Templates, Chains, and Structured Output Parsers.

Goal: Extract revenue, profit, and outlook from financial texts using LLMs.

1 Notebook Setup Reminder

- We will **reuse Ollama or Groq LLM** already initialized in Day 92.
- Ensure required libraries are installed (`langchain`, `ollama`, `langchain_ollama`, `langchain_groq`).
- The code will follow a **step-by-step pipeline**: import → define schema → create prompt → create chain → run sample input.

2 Import Libraries

- `Ollama` → Local LLM
- `PromptTemplate` → Reusable prompts
- `LLMChain → Connect prompt` → LLM → output
- `StructuredOutputParser` → Ensures structured output

[1]:
```
from langchain.llms import Ollama
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.output_parsers import StructuredOutputParser, ResponseSchema
```

3 Define Schema

- Each schema defines a **key** and **description**.
- Parser will **validate and structure LLM output** as JSON.

[2]:
```
# Define the structure of the extracted data
schemas = [
    ResponseSchema(name="revenue", description="Revenue details"),
    ResponseSchema(name="profit", description="Profit details"),
    ResponseSchema(name="outlook", description="Future outlook"),
]
```

```
# Initialize parser
parser = StructuredOutputParser.from_response_schemas(schemas)

# Get formatting instructions for LLM prompt
format_instructions = parser.get_format_instructions()
```

4 Create Prompt

- **text** → Variable to input financial data.
- **format_instructions** → Ensures LLM outputs in structured JSON.

[13]:
```
template = """You are a financial analyst.
Extract revenue, profit, and outlook from the following text.
Text: {text}

Return the output **strictly as JSON** with all keys:
"revenue", "profit", "outlook".
- Do NOT add any explanations, bullet points, or markdown.
- If any value is missing, put "N/A".

Example output:
{{
  "revenue": "value",
  "profit": "value",
  "outlook": "value"
}}
"""
prompt = PromptTemplate(template=template, input_variables=["text"])
```

5 Create LLM Chain

LLMChain links **prompt → LLM → structured output.**

We can switch `llm_local` or `llm_groq` depending on whether we want local or cloud execution.

[16]:
```
from langchain_ollama import ChatOllama
# Local LLM setup
llm_local = ChatOllama(
    model="llama3.2:1b",    # Model downloaded in Ollama
    temperature=1.8,         # Creativity
    num_predict=50          # Number of tokens generated
)

# Create chain using RunnableSequence style
chain = prompt | llm_local  # no LLMChain needed
```

6 Run with Sample Input

- `chain.run()` → Sends text to LLM
- `parser.parse()` → Converts LLM output into structured JSON

[17]:
```
text = "Apple reported revenue of $90B and profit of $25B. The outlook is␣
 ↪strong with growth in services."
# Run chain
result = chain.invoke({'text': text})

# Get string content from AIMessage
llm_output = result.content

# Parse output
parsed = parser.parse(llm_output)
print(parsed)
```

```
{'revenue': '$90B', 'profit': '$25B', 'outlook': 'Strong'}
```

[18]:
```
print("Revenue:", parsed['revenue'])
print("Profit:", parsed['profit'])
print("Outlook:", parsed['outlook'])
```

```
Revenue: $90B
Profit: $25B
Outlook: Strong
```

[19]:
```
text = """
Tesla Inc. announced its Q2 2025 earnings, reporting total revenue of $45.7B,␣
 ↪an increase of 20%
compared to the previous quarter. Net profit rose to $7.2B, reflecting strong␣
 ↪vehicle deliveries
and growing energy product sales. The company highlighted robust growth in its␣
 ↪solar and battery
business segments. Outlook for Q3 2025 remains optimistic, with plans to expand␣
 ↪production
capacity and launch new vehicle models globally. CEO Elon Musk emphasized focus␣
 ↪on sustainable
energy solutions and innovation across all divisions.
"""

# Run chain
result = chain.invoke({'text': text})

# Get string content from AIMessage
llm_output = result.content

# Parse output
```

```
parsed = parser.parse(llm_output)
print(parsed)
```

```
{'revenue': '$45.7B', 'profit': '$7.2B', 'outlook': 'optimistic'}
```

[20]:
```
print("Revenue:", parsed['revenue'])
print("Profit:", parsed['profit'])
print("Outlook:", parsed['outlook'])
```

```
Revenue: $45.7B
Profit: $7.2B
Outlook: optimistic
```

7 Summary

- Built a **Financial Data Extraction App** using LangChain.
- Used **Prompt Templates**, **Runnable Chains**, and **Structured Output Parsers**.
- Successfully extracted **revenue, profit, and outlook** from sample text.
  # Next Steps
- Try **Groq API** for faster cloud inference.
- Build a **Streamlit/Gradio interface** for user input.
- Handle **multiple companies** or **long reports**.
- Save extracted data to **CSV/Excel**.

8 Practice Ideas

- Extract additional fields like **CEO or expenses**.
- Compare results between **local LLM (Ollama)** and **cloud LLM (Groq)**.
- Make a mini **Financial Insights Bot** for Q&A using the same parser.