Gen AI agent for data engineering

GenAI-Powered Data Engineering Agent

Project Overview

In real-world data engineering workflows, data is often stored in **multiple formats and schemas**, making interoperability a challenge. This project will explore how **Generative AI (GenAI) can automate data format detection, schema inference, and serialization/deserialization** by generating Python code dynamically.

The goal is to build an **autonomous data engineering agent** that:

1. **Scans an S3 prefix (or a local folder for simplicity)** to identify files and their formats (e.g., JSON, CSV, Avro, XML, Parquet, etc.).
2. **Infers schemas** and groups files into distinct schema categories.
3. **Dynamically generates Python classes** that handle serialization and deserialization for each schema.
4. **Converts data into standardized formats** (e.g., Parquet for analytics, Avro for streaming).
5. **Performs data cleaning and transformation** before storage.
6. **Writes transformed data** to an S3 prefix (or a local folder for simplicity), partitioned by schema type.

This project integrates **LLM-based code generation, data engineering, and automation**, creating a pipeline that **writes and executes its own serialization/deserialization logic** based on the data it encounters.

Why Synthetic Data?

To **test and evaluate** the data engineering agent without relying on sensitive or proprietary datasets, we will use **synthetic data**—artificially generated datasets that mimic real-world data structures. Synthetic data allows us to:

* **Control complexity** (e.g., structured vs. semi-structured data).
* **Test edge cases** (e.g., nested JSON, inconsistent schemas, missing values).
* **Benchmark performance** under different scenarios.

Example Synthetic Data Files

The project will generate synthetic data files in various formats, such as:

1\. JSON Files (Nested & Flat)

`nested_data.json`

json
CopyEdit
`{`
`"user": {"id": 123, "name": "Alice", "email": "alice@example.com"},`
`"transactions": [`
`{"id": 1, "amount": 50.75, "timestamp": "2024-01-01T10:00:00Z"},`
`{"id": 2, "amount": 20.00, "timestamp": "2024-01-02T12:30:00Z"}`
`]`
`}`

2\. CSV Files (Simple & Messy)

`users.csv`

csv
CopyEdit
`id,name,email,age`
`1,John Doe,john@example.com,25`
`2,Jane Smith,jane@example.com,`
`3,Bob,,30`

3\. XML Data

`products.xml`

xml
CopyEdit
`<products>`
`<product>`
`<id>101</id>`
`<name>Smartphone</name>`
`<price>699.99</price>`
`</product>`
`<product>`
`<id>102</id>`
`<name>Laptop</name>`
`<price>1299.99</price>`
`</product>`

`</products>`

4\. Avro & Parquet Files

Binary files containing structured data, which the agent will generate and process dynamically.

Technical Components & Implementation

1. **Generative AI for Code Generation**
    * Use an **LLM (e.g., Llama 3.3, Nova, Claude 3.5 Haiku) to generate Python code** to write code for determining the schema of a file, write code for serialization/deserialization.
    * Prompt engineering: "Write a Python class that reads XML and converts it to Parquet."
    * Self-correction & validation using LLM-based evaluation.
    * Have the LLM write all code as per PEP8, use PyDantic, and the usual good programming practices.
2. **Schema Inference & Transformation**
    * Extract schema from JSON, CSV, XML, and other formats.
    * Identify **duplicate fields, nested structures, and missing values**.
    * Standardize data into a **clean, structured format**.
3. **Execution & Automation**
    * **Write & execute** the LLM-generated Python code dynamically.
    * Automate the pipeline for **schema-based partitioning and storage** in S3.

Evaluation & Success Metrics

* **Accuracy of schema detection** (comparison with ground truth).
* **Correctness of generated serialization/deserialization code**.
* **Efficiency of format conversion** (latency, data size reduction).
* **Scalability tests** with large datasets.

Why This Project Matters?

This project demonstrates the potential of **GenAI in automating complex data engineering workflows**, reducing manual effort, and enabling AI-driven **data pipeline orchestration**.