Auto-Integrate

Aditya Kunatharaju, Tanya Jain
Sponsor: Jags Krishnamurthy (ObjectEdge)

UNIVERSITY OF SAN FRANCISCO

Introduction

Auto-Integrate serves to address the challenges faced in integrating diverse data formats utilized by various applications. These formats often lack standardization, making it difficult to establish effective communication and integration between these applications.

The primary objective of Auto-Integrate is to automate the integration of application data, simplifying the process and enhancing interoperability.

Problem Statement

The core problem being addressed is the inherent difficulty in integration applications due to varying data formats and lack of standardization. Fields standing for the same feature may not have the same name. For example, "business_name" and "business". Currently, developers and analysts must manually map data from one field to another across applications for integration.

We seek to automate this process.

Intended Users: Developers, analysts, consultants, and other professionals needing to integrate application data.

Our Solution

We sought to achieve automation by:

- Automatically collecting source and target applications data using API endpoints, and comprehending the data formats.
- Prompting LLMs to map fields from the source API to the target API.
- Parsing the LLM response to generate an integration pipeline for successful data transfer.
- Providing mapping metrics such as coverage and accuracy for the user to assess integration feasibility.
- Logging each step to facilitate user auditing and review of the process details, aiding in debugging when errors are encountered.

Our Process

We started this project by researching the numerous LLM's available. We tested OpenAI's GPT3.5 Turbo, DaVinci, and GPT-4, Meta's Llama-2, Vicuna, and TooLlama.

Our initial goal was to deploy Llama-2 locally, however it's performance was not as reliable or accurate as GPT-3.5 Turbo or GPT-4. Our final solution leverages GPT-4 models, aligning with the expectations of our intended users.

In an effort to enhance the LLM's response, we implemented Microsoft's AutoGen framework to construct two bespoke AI driven multi-agent conversational frameworks. This strategic use of multiple AI agents allows each agent to focus on a distinct facet of the integration process.

Once we get a mapping, we setup an integration pipeline that allows a seamless transfer of data from the source API to the target API.

Multi Agent Framework

Our multi-agent framework leverages the power of LLMs to focus on the key processes of the integration. One of these concerns the data extraction and aggregation process, while the second with data mapping.

<u>Data Extraction and Aggreagation AI Agents:</u>

- **Information Extractor:** Use API documentation to extract information.
- **Data Format Analyzer:** Identify data types & structure of data models.

<u>Data Mapping AI Agents:</u>

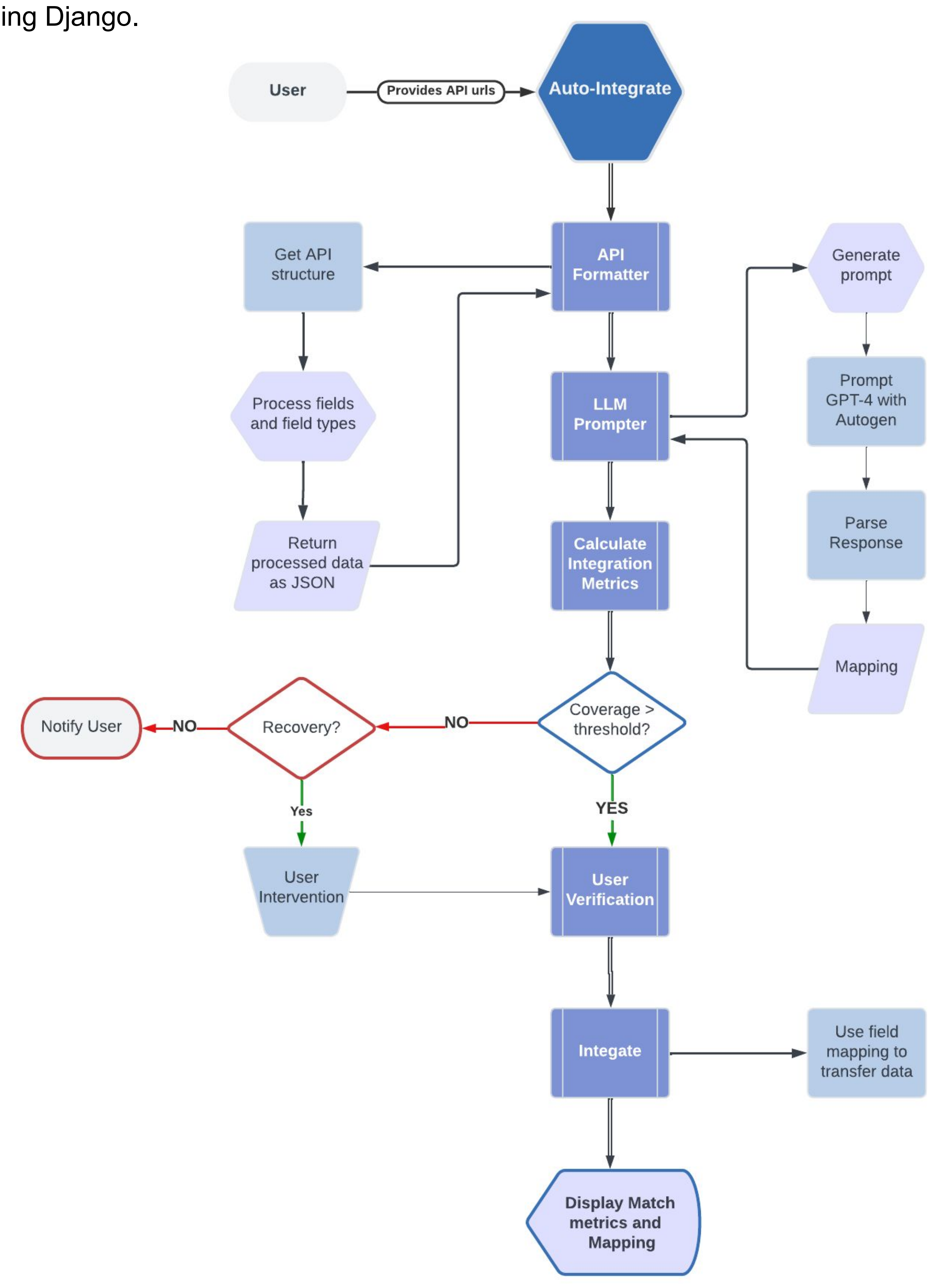
- **Schema Analyzer:** Analyzes the schema or data models of both APIs. Recommends based on types, constraints, and semantic meaning.
- **Naming Convention Analyzer:** Analyzes the naming conventions, recommends field mappings based on semantic meaning using NLP.
- **Data Format Compatibility Analyzer:** Ensures the data format compatibility. Validates the formatting of data after any transformation.

<u>Common AI Agents:</u>

- **Task Manager:** Manages the other agents by assigning tasks and monitoring progress. Ensures the agents are working on the right tasks.
- **User Proxy:** Initiates and terminates the chat on completion.

Data Flow

The user can interact with our application using either a command line or a UI built using Django.



Technology Stack

OpenAI
django
AutoGen
HTML
CSS

Conclusion

The best performing LLM was OpenAI's GPT-4. It consistently gave us reliable mappings, with minimal hallucinations, and in the required format.

Other LLMs such as Llama-2 were prone to hallucinations and inaccuracy, often changing field names or incorrectly mapping fields.

Using a multi-agent framework further increased reliability and allowed us to generate output formats with greater complexity, which made generating the integration pipeline easier.

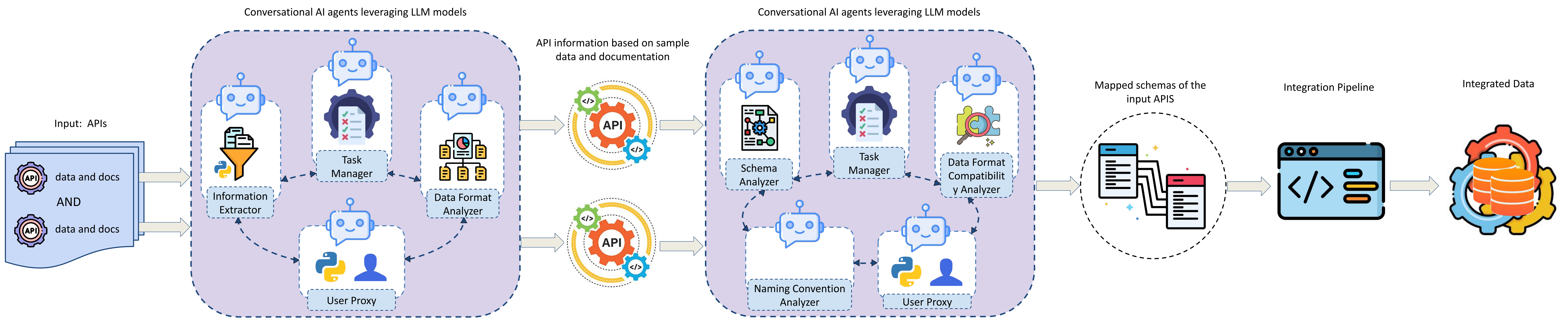
Within the multi-agent framework, we found the best results were given when there was a slight overlap in responsibilities between the agents, and when it was highlighted that the task at hand is of deterministic nature and the response received is of high significance in a real world scenario.

Contact Us

Aditya Kunatharaju
aditya17varma@gmail.com
(415) 999-0174

Tanya Jain
tanya.jain27may@gmail.com
(415) 867-7356

Design Overview


Conversational AI agents leveraging LLM models
Conversational AI agents leveraging LLM models
Input: APIs
data and docs
AND
data and docs
Information Extractor
Task Manager
Data Format Analyzer
User Proxy
API information based on sample data and documentation
API
API
Schema Analyzer
Task Manager
Data Format Compatibility Analyzer
Naming Convention Analyzer
User Proxy
Mapped schemas of the input APIS
Integration Pipeline
Integrated Data