Research Document: Implementation of an AI Agent with OpenRouter

May 26, 2025

1 Introduction

This document analyzes a Python script that implements an AI agent using the OpenRouter platform, a unified interface for accessing large language models (LLMs). The script leverages the OpenAI Python SDK and the `openai-agents` package to interact with OpenRouter's API, specifically using the DeepSeek model. The purpose is to create a conversational agent that processes user input and responds in English.

2 Code Overview

The script is a command-line interface for interacting with an AI agent powered by OpenRouter's DeepSeek model. Below is the code:

```python
import asyncio
from openai import AsyncOpenAI
from agents import Agent, OpenAIChatCompletionsModel, Runner, set_tracing_disabled
from dotenv import load_dotenv
import os

load_dotenv()
set_tracing_disabled(disabled=True)

client = AsyncOpenAI(
    api_key=os.getenv("OPENAI_API_KEY"),
    base_url="https://openrouter.ai/api/v1"
)

async def main():
    agent = Agent(
        name="deepseek",
        instructions="You␣only␣respond␣in␣english.",
        model=OpenAIChatCompletionsModel(model="deepseek/deepseek-chat-v3-0324:free", openai_client=client),
    )
    print("Agent␣created,␣Type␣'exit'␣to␣quit.\n")
    while True:
        user_input = input("User: ")
        if user_input.lower() == "exit":
            break
        result = await Runner.run(
            agent,
            user_input,
        )
        print(f"Agent: {result.final_output}\n")

asyncio.run(main())
```

3 Key Components

The script uses several Python libraries:

- `asyncio`: Enables asynchronous programming for non-blocking API calls.
- `openai`: Provides the `AsyncOpenAI` client for interacting with OpenRouter's API, which is compatible with OpenAI's ChatCompletion API.
- `agents`: Supplies `Agent`, `OpenAIChatCompletionsModel`, and `Runner` for agent orchestration, likely from a custom package (`panaversity/learn-agentic-ai`).
- `dotenv`: Loads environment variables from a `.env` file.
- `os`: Accesses environment variables.

The `set_tracing_disabled(disabled=True)` function disables logging of internal operations (e.g., API calls) in the `openai-agents` package, reducing overhead and log clutter.

The `AsyncOpenAI` client is initialized with OpenRouter's API endpoint (`https://openrouter.ai/ap`
and the API key, enabling asynchronous requests to the DeepSeek model (`deepseek/deepseek-chat-v3`

The `Agent` is configured with:

- `name`: "deepseek"
- `instructions`: Restricts responses to English.
- `model`: Uses `OpenAIChatCompletionsModel` with the DeepSeek model.

The `main` function runs an asynchronous loop, prompting the user for input, exiting on "exit", and processing input via `Runner.run`, which returns the agent's response.

4 Functionality

The script creates a command-line interface where users input queries, and the DeepSeek agent responds. It uses OpenRouter's free model, limited to 200 requests per day and 20 requests per minute. The `openai-agents` package orchestrates the agent's behavior, leveraging OpenRouter's compatibility with OpenAI's API.

5 Limitations

- **No History Persistence**: The script does not save or use conversation history, so the agent cannot recall prior interactions.
- **API Key Issue**: The use of `OPENAI_API_KEY` may be incorrect; `DEEPSEEK_OPENROUTER_API_KEY` is likely needed.
- **Model Name**: The model ID (`deepseek/deepseek-chat-v3-0324:free`) must be verified via OpenRouter's API.
- **Dependency**: Relies on the `openai-agents` package, which may have specific requirements or compatibility issues.

6 Recommendations

- Add history persistence using a file (e.g., `history.json`) to maintain conversation context.
- Test with a raw `AsyncOpenAI` call to isolate `openai-agents` issues.

7 Conclusion

The script demonstrates a simple AI agent implementation using OpenRouter's DeepSeek model. While functional for basic interactions, adding history persistence and fixing configuration issues would enhance its utility. OpenRouter's unified API simplifies access to LLMs, making this a valuable starting point for agent-based applications.