Develop an AI Research Agent with AutoGen

Assignment

Developing an AI Healthcare Research Assistant using AutoGen

Scenario

MediSyn Labs, a clinical research and medical innovation company, supports medical researchers, healthcare analysts, and clinicians working on treatment efficacy and drug outcomes. These professionals often spend long hours reviewing medical literature, summarizing findings, and comparing therapies across diseases and populations.

To streamline this process**, MediSyn Labs** is building an AI-Powered Healthcare Research Assistant that can interpret clinical queries, summarize large volumes of literature, generate comparative studies, and assist with ongoing research using a natural language interface. The assistant should leverage AutoGen for orchestration, Gemini as the core LLM, and a Streamlit UI to ensure real-time interaction between human researchers and the agent.

Objectives

- Develop an intelligent research agent using AutoGen and Gemini tailored for healthcare.
- Build a flexible query-response system that supports summarization, session memory, and query comparison.
- Integrate browser based Streamlit UI to enable healthcare researchers to interact with the agent.
- Incorporate HITL feature, especially for generating critical summaries, subtopic generation and functionality to download the report.

Problem Statement

- **Time-Consuming Manual Review**: Medical professionals at MediSyn Labs spend excessive hours manually reviewing vast amounts of medical literature.
- **Inefficient Data Synthesis**: The process of summarizing findings and comparing therapies across diseases and populations is labor-intensive and inefficient.
- **Research Bottleneck**: This manual effort creates a significant bottleneck, impeding the efficiency and speed of critical research into treatment efficacy and drug outcomes.
- **Prone to Human Error:** Current methods are susceptible to human error, potentially impacting the accuracy of research findings.
- **Limited Scope and Depth of Analysis**: The time constraints and manual nature of the work restrict the scope and depth of analysis that can be realistically achieved.
- **Delayed Medical Advancement**: Ultimately, these limitations delay the progression of medical understanding and innovation.

Tasks to Be Implemented

1. Define a Healthcare Memory Schema

Design an AgentState with:

- Short-term memory: Active session queries and responses
- Long-term memory: Stored literature summaries, comparative findings, or case history
- Metadata fields like researcher_id, project_id, and disease_focus

2. Implement State Management

- Use StateGraph to manage state transitions.
- Define a state reducer to update memory with each interaction.

3. Manage Memory

- Use short-term memory to hold current queries (e.g., “What does recent literature say about mRNA vaccines?”)
- Use long-term memory (via MemorySaver or a DB) to store historical responses and summaries across sessions.

4. Handle Medical Queries

- Create an LLM node for medical literature summarization, treatment comparisons, and clinical question answering.
- Add tool nodes for memory retrieval or custom medical knowledge tools (e.g., PubMed fetchers or metadata parsers).

5. Trim and Filter Messages

- Limit the short-term memory to 5–7 clinically relevant queries.
- Filter out non-informational inputs (e.g., greetings or vague queries) to optimize processing.

6. Configuration and Deployment

- Manage credentials via environment variables.
- Write comprehensive documentation (README.md) including usage, setup, and customization steps that will help you in future scaling.

Feature Suggestions (Optional)

- **Multiple Schemas:** Separate schemas for treatment summaries, clinical trial notes, and user-specific projects.
- **External Memory Store:** Use an external database to persist long-term memory across research projects (e.g., summaries of PubMed searches or drug trial notes).
- **Human-in-the-Loop Prompts:** Prompt researchers for approval:
    - “Approve this summary of the latest COPD treatment trial?”

- “Flag this literature comparison between Remdesivir and Paxlovid?”

- **Schema Validation:** Validate fields like disease_focus (string), messages (list of dicts), and summary_rating (int or optional).
- **Enhanced UI Components:** Use collapsible sections, charts, or tabs to show treatment outcomes, dosage differences, or adverse event rates.

Note:

Make use of Virtual Environment to Avoid Version issues:

- Create a Virtual Environment:
  - Navigate to the project directory (e.g., D:\mod\). o Run: python -m venv venv.
  - Activate the virtual environment:
    - Windows: venv\Scripts\activate
    - macOS/Linux: source venv/bin/activate
- Install Dependencies:
  - Create a requirements.txt with necessary packages.
  - Run: pip install -r requirements.txt to install dependencies in the virtual environment.