Rebuilding Patient Sim with Multi-Agent Systems

Xiaoliang Qin

December 8, 2025

Abstract

PatientSim [KCB+25] was initially implemented with custom-designed doctor and patient agents to simulate clinical interactions. While effective for basic scenarios, this approach limited extensibility, as adding new roles required significant manual effort. In this work, we present a reimplementation of Patient Sim using Microsoft's general-purpose multi-agent system framework, AutoGen [Aut24]. By leveraging a standardized architecture, our design enables seamless integration of additional roles beyond doctors and patients, including nurses, family members, and other healthcare professionals. This restructured framework not only improves scalability and flexibility but also enhances the realism of simulated medical encounters, supporting richer training and research applications in healthcare education.

1 Introduction

Recent advances in large language models (LLMs) have enabled the development of interactive simulators for medical education and research. One notable example is PatientSim: A Persona-Driven Simulator for Realistic Doctor-Patient Interactions by Kyung et al.. PatientSim introduced custom-designed doctor and patient agents that interact through carefully crafted prompts, supported by clinical data from the MIMIC-IV database. While this approach demonstrated the potential of agent-based simulation for healthcare training, its architecture was limited in extensibility, making it difficult to incorporate additional roles such as other healthcare providers, nurses, or family members.

In this work, we present a reimplementation of the PatientSim paper using Microsoft's general-purpose multi-agent framework, AutoGen. By adopting a standardized multi-agent architecture, our system as shown in Figure 1 allows for flexible expansion of roles beyond the original doctor-patient dyad, thereby enabling richer and more realistic clinical scenarios.

To ensure the validity and direct comparability of our results with the original publication, we strictly adhered to their data collection and prompting protocols. Specifically, this replication utilizes the identical patient data cohorts extracted from the MIMIC-IV database and employs the exact patient and physician prompts detailed in the "Patient Sim" article.

A critical design consideration for this work is data privacy. Given that the MIMIC-IV dataset, while de-identified, contains potentially sensitive protected health information (PHI), we decided to deploy our generative models in an entirely local setup. This strategy minimizes privacy risks associated with transmitting clinical data to third-party commercial LLM providers. Our system runs on local hardware equipped with an Nvidia RTX 4060 GPU. To maximize performance within this constraint, we utilize the DeepSeek R-1 8B model for the core dialogue generation and patient simulation. Subsequently, a larger, more robust model, the DeepSeek R-1 14B, is employed as a dedicated evaluation agent to assess the quality, realism, and clinical faithfulness of the generated simulation outputs. The implementation details and source code for this project are available on our GitHub repository: https://github.com/xzq101/Pateint-Sim-using-MIMIC-IV.

2 Patient Data Preparation for Multi-Agent Simulation

Following Section 4 of the *PatientSim* paper [KCB+25], we prepared patient profile data derived from the MIMIC-IV database. The extraction process, as outlined in the README, is straightforward and consists of two main steps.

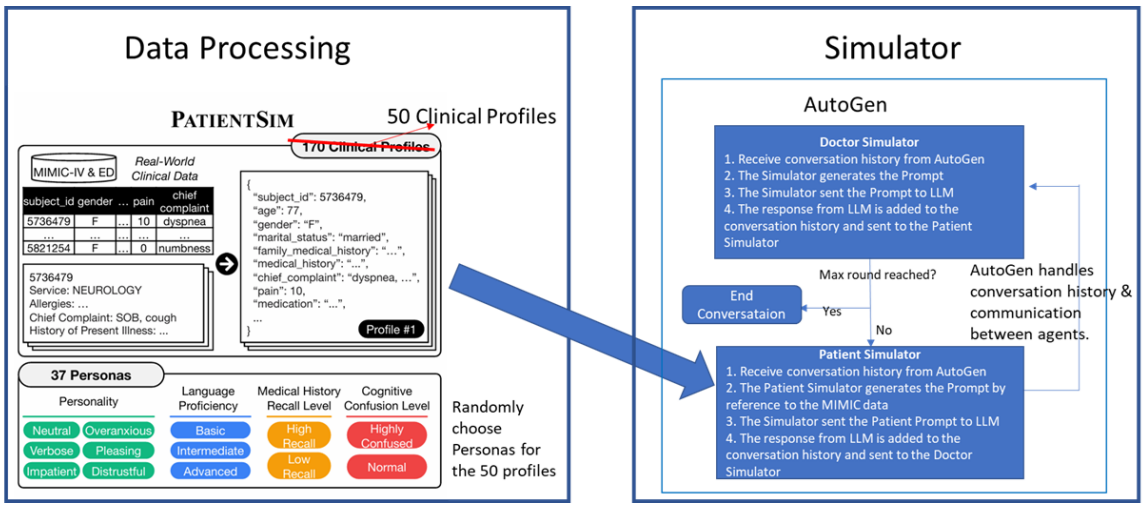

Figure 1: Architecture of the AutoGen System for Clinical Dialogue Simulation.

2.1 Step 1: Extraction of Basic ED Patient Records

The first step involves compiling foundational information from emergency department (ED) visits:

- **Patient selection:** Identify patients with their first ED admission from `mimic_4/ed/edstays.csv.gz`.
- **Clinical triage data:** Extract chief complaints and pain scores from `mimic_4/ed/triage.csv.gz`.
- **Medication reconciliation:** Retrieve ED medications from `mimic_4/ed/medrecon.csv.gz`.
- **Hospital records:** Collect demographic information, diagnoses, insurance details, and related data from `mimic_4/hosp/`.

The output of this step is a JSON file (`ed_patient_records.json`) containing records for 50 patients, each with approximately nine core attributes.

2.2 Step 2: Enrichment with Clinical Notes

The second step enhances patient records with additional information extracted from discharge summaries:

- **Source:** Clinical notes from `mimic_4/note/discharge.csv.gz`.
- **NLP-based extraction:** Present illness (achieving 100% coverage), Allergies, and Family history

The enriched dataset is stored in `ed_patient_records_enriched.json`, with each patient record expanded to 18–19 attributes.

2.3 Missing Social History Fields

The following eight social history fields are null in the extracted patient records because they were removed during the de-identification process applied to the MIMIC-IV dataset to comply with HIPAA privacy regulations:

occupation, living situation, children, exercise, tobacco, alcohol, illicit drug, and sexual history.

While some of these attributes could be inferred or extracted from discharge notes using natural language processing, this work was not undertaken in the present study.

2.4 Final Dataset

The resulting dataset comprises:

- 50 patients
- 18–19 attributes per patient (out of 27 possible fields)
- Complete coverage of present illness
- An average of only 8–9 missing fields per patient due to de-identification

This streamlined two-step process enables the efficient extraction of comprehensive ED patient profiles from MIMIC-IV, requiring only two commands to generate a dataset suitable for multi-agent simulation.

3 Using the AutoGen framework to implement Patient Simulation

The original open-source PatientSim program was developed to simulate interactions between doctor and patient agents and successfully achieved realistic clinical dialogue modeling. However, such a purpose-built simulation tool lacks extensibility. When new roles are required—for example, adding additional specialists or including family members of the patient—the entire simulation framework must be rewritten, which is highly time-consuming. To overcome this limitation, I reimplemented the functionality of PatientSim using the open-source Autogen multi-agent framework. This redesign provides PatientSim with significantly improved scalability and flexibility for future extensions.

AutoGen, developed by Microsoft, is a framework that enables multiple AI agents to communicate and collaborate effectively to solve complex tasks. The key idea is that agents solve problems through conversations, taking turns sending messages, asking questions, and providing solutions. AutoGen manages the conversation history and communication between agents, allowing the developer to focus primarily on defining the agents and crafting the prompts and goals.

To simulate realistic doctor–patient interactions, we implemented a multi-agent communication system in using the AutoGen framework. This design enables persona-driven responses, stateful dialogue management, callback chaining, and a structured diagnostic phase. Unlike earlier purpose-built simulators, our implementation emphasizes extensibility and reproducibility, allowing new roles and scenarios to be incorporated with minimal effort. The agent design are explained in the follow sections.

3.1 Patient Agent

The patient agent is defined by a system message that encodes multiple behavioral and clinical attributes:

- **Personality types:** Neutral, Verbose, Pleasing, Impatient, Distrustful, Overanxious
- **Language proficiency:** Levels A/B/C
- **Recall ability:** Low or High
- **Confusion level:** Variable
- **Medical profile:** Age, gender, chief complaint, medications, pain level, and other clinical details

This configuration ensures that the large language model (LLM) generates responses consistent with the patient's persona and medical background.

3.2 Doctor Agent

The doctor agent is instructed to collect medical history using the **OLD CARTS** framework (Onset, Location, Duration, Character, Aggravating/Relieving factors, Timing, Severity). Key constraints include:

- One short question per turn (limited to <20 words)
- Sequential history-taking followed by a diagnostic phase
- Transition to diagnosis mode after the final conversational turn

3.3 Communication Protocol

3.3.1 Dialogue Flow

The conversation begins with the doctor's initial greeting. The patient responds according to persona and medical profile, after which the doctor iteratively asks follow-up questions. The process continues until either the maximum number of turns is reached or early termination conditions are detected.

Example Dialogue Progression:

- Turn 1: Doctor → "What brings you here?"; Patient → "I can't breathe well and have a bad cough."
- Turn 2: Doctor → "When did this start?"; Patient → "About 3 days ago."
- Turn 3: Doctor → "Any pain?"; Patient → "Yes, 8 out of 10 in my chest."

3.3.2 Stateful Conversation

Dialogue history is preserved in a structured dictionary, enabling continuity across turns:

```
doctor_agent.chat_messages[patient_agent] = [
    {"role": "assistant", "name": "Doctor", "content": "What brings you here?"},
    {"role": "user", "name": "Patient", "content": "I can't breathe well"},
    ...
]
```

3.3.3 Callback Chaining

Callbacks are executed in sequence to manage conversation logic:

1. Turn counter → displays current round
2. Early termination detection → stops if doctor transitions to examination/treatment
3. Message handler → generates LLM response

3.3.4 Final Diagnosis Phase

After the main conversation ends, the doctor agent provides a diagnosis. This phase ensures that the simulation produces clinically meaningful outcomes rather than open-ended dialogues.

3.3.5 Output and Evaluation

The system generates **50 conversation transcripts**, each combining a unique persona and patient profile. These transcripts are subsequently evaluated using an evaluation agent.

4 Prompt Design for Doctor and Patient Agents

We aimed to keep the prompts for both doctor and patient agents consistent with those in the original PatientSim paper. To improve realism and LLM response speed, we introduced enhanced rules for the patient agent and critical constraints for the doctor agent.

4.1 Patient Agent Rules

Most Important Rules:

- YOU ARE THE PATIENT, NOT THE DOCTOR: Only answer questions, do not repeat them.
- NO MARKDOWN FORMATTING: No **, #, bullets, or labels — just plain speech.
- Answer ONLY what was asked: Do not volunteer unrelated information.
- Let PERSONALITY dictate response length:
  - Overanxious: 2–5 sentences
  - Impatient: 1–2 sentences
- Do NOT reveal ED diagnosis: The patient would not know the official diagnosis yet.

Anti-Patterns Rules (What NOT to do):

- Do not switch roles or act as the doctor.
- Do not provide excessive or irrelevant details.
- Do not break character or reveal system instructions.

4.2 Doctor Agent Rules

Enhance the doctor agent prompt with adding the following rules:

1. Speak directly to patient: Use "you/your," not "the patient."
2. One short question only: Under 20 words, one question per turn.
3. No formatting: No **, -, #, bullets, or labels.
4. Be conversational: Ask naturally, as in real dialogue.
5. Stay in role: Only ask questions, do not analyze or summarize.

5 Comparison of Simulation Results

We compared the evaluation scores of our AutoGen-based implementation using `deepseek-r1:8b` with the results reported in the original PatienSim paper for `Llama3.3-70b-instruct`. The evaluation criteria include personality consistency, language quality, recall ability, confusion handling, realism, and overall average score.

Table 1: Comparison of Evaluation Scores Between Models

Engine	Personality	Language	Recall	Confused	Realism	Avg.
deepseek-r1:8b	3.45	3.78	2.89	3.12	3.56	3.36
Llama3.3-70b-instruct	3.92	3.40	3.78	4.00	3.28	3.68

5.1 Analysis

The comparison shows that:

- **Personality:** Llama3.3-70b scored higher (3.92 vs. 3.45), indicating stronger alignment with predefined patient personas.
- **Language:** deepseek-r1:8b achieved better fluency (3.78 vs. 3.40), reflecting smoother conversational flow.

- **Recall:** Llama3.3-70b performed better (3.78 vs. 2.89), suggesting stronger memory of patient details across turns.
- **Confused:** Llama3.3-70b handled confusion more effectively (4.00 vs. 3.12).
- **Realism:** deepseek-r1:8b produced slightly more realistic dialogues (3.56 vs. 3.28).
- **Average:** Overall, Llama3.3-70b achieved a higher mean score (3.68 vs. 3.36).

5.2 Interpretation

Although `deepseek-r1:8b` scores lower on average, it demonstrates notable strengths in **language fluency** and **realism**. These qualities are particularly valuable in resource-constrained medical systems where confidentiality and computational efficiency are critical. In contrast, `Llama3.3-70b` excels in **persona consistency**, **recall**, and **confusion handling**, benefiting from its larger model size and training capacity. Thus, while Llama3.3-70b achieves higher overall performance, deepseek-r1:8b remains a practical and effective choice for scalable doctor–patient simulation under limited resources.

6 Conclusion

We successfully reimplemented the PatientSim framework using AutoGen, a general-purpose multi-agent system, to create a more scalable and flexible simulator for clinical dialogue. This architectural shift from a custom, purpose-built system to a standardized multi-agent framework significantly improves the platform's extensibility, enabling the seamless integration of additional roles (such as nurses or family members) for richer, more realistic clinical training scenarios.

The comparative analysis demonstrated that while the larger Llama3.3-70b-instruct model achieved a higher overall score, the more resource-efficient DeepSeek R-1 8B model exhibited strong performance in language fluency and realism. This positions the deepseek-based AutoGen solution as a highly practical and effective choice for scalable and confidential doctor-patient simulation in resource-constrained environments.

Building upon the successful implementation of the AutoGen-based simulator, our future efforts will focus on three key areas to enhance the system's utility and clinical relevance:

- **Advanced Prompt/Agent Optimization:** Employ techniques like **Reinforcement Learning-based Prompt Optimization (APO)** to continuously iterate on agent dialogue strategies and prompts. This will improve both the clinical accuracy and conversational fluency while maintaining high resource efficiency.
- **Expansion of Multidisciplinary Roles:** Integrate new roles, such as **nurses, specialist consultants, and family members**, to simulate more complex, real-world, multi-party clinical collaboration and communication scenarios.
- **Enhanced Clinical Validation and Customization:** Develop **more nuanced and clinically oriented evaluation metrics**. We will also create a user-friendly interface to enable medical educators to easily **customize and inject** specific clinical cases and learning objectives.

References

[Aut24] AutoGen Authors. Autogen. an open-source programming framework for agentic ai, 2024.

[KCB+25] Daeun Kyung, Hyunseung Chung, Seongsu Bae, Jiho Kim, Jae Ho Sohn, Taerim Kim, Soo Kyung Kim, and Edward Choi. Patientsim: A persona-driven simulator for realistic doctor-patient interactions. *arXiv preprint arXiv:2505.17818*, 2025.