PATIENTLM: CONTROLLABLE PATIENT SIMULATION FOR INTERACTIVE MEDICAL EVALUATION AND TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Medical assistants are increasingly used in multi-turn consultations, where patients progressively disclose information and adapt their follow-up to assistant responses. However, existing multi-turn evaluation protocols typically rely on fixed dialogue contexts or predefined patient-side trajectories. Such protocols cannot measure how an assistant elicits information or shapes subsequent patient behavior. We introduce **PatientLM**, a patient simulator that combines a trained language model with a controller for specific medical interaction scenarios. PatientLM represents each scenario with a structured patient intent that specifies the patient role and goal, together with the clinical information and requests to disclose over the course of the dialogue. A controller governs this progression, while the trained language model expresses the currently available information naturally in response to the evolving interaction. Simulator evaluations show that PatientLM most completely communicates case-specified information and requests and achieves the strongest overall performance on naturalness and other simulator-quality measures. We then use PatientLM to unify different multi-turn dialogue evaluation paradigms under a common interactive protocol, and experiments show that assistant performance is generally lower than in the original evaluations, suggesting that fixed patient trajectories may overestimate interactive performance. PatientLM also supports effective online SFT for smaller assistant models, improving their factuality and reliability. These results highlight controllable patient simulation as a promising foundation for advancing interactive medical AI.

1 INTRODUCTION

Large language models (LLMs) are increasingly used as assistants in multi-turn medical consultations (Singhal et al., 2025). In such consultations, patients may begin with partial symptoms, supply history after targeted questions, and raise additional requests as the interaction develops. This progressive exchange is characteristic of real-world health conversations (Paruchuri et al., 2025). Evaluating a medical assistant therefore requires assessing not only how it responds once the case is fully specified, but also how it elicits and responds to information as the clinical context emerges through interaction.

However, common evaluation protocols capture only part of this process. Static-context benchmarks such as HealthBench provide a fixed conversation before the response being evaluated (Arora et al., 2025). Dynamic-history benchmarks such as KnowMT-Bench allow earlier assistant responses to enter subsequent contexts, thereby exposing the model to its own interaction history (Chen et al., 2025). Both settings nevertheless specify the patient-side trajectory in advance. The evaluated assistant cannot influence how the patient later explains the case, responds to a misunderstanding, or decides which concern to raise next. Performance under these fixed trajectories may therefore differ from performance when the same clinical context must be acquired through an adaptive interaction.

Supporting such adaptive interactions requires a patient simulator with two complementary capabilities. First, it should respond to the evolving dialogue while maintaining a consistent patient role, communication style, clinical situation, and overall goal. Second, it should disclose case-specific information progressively without omitting the clinical facts and requests that the interaction is in-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

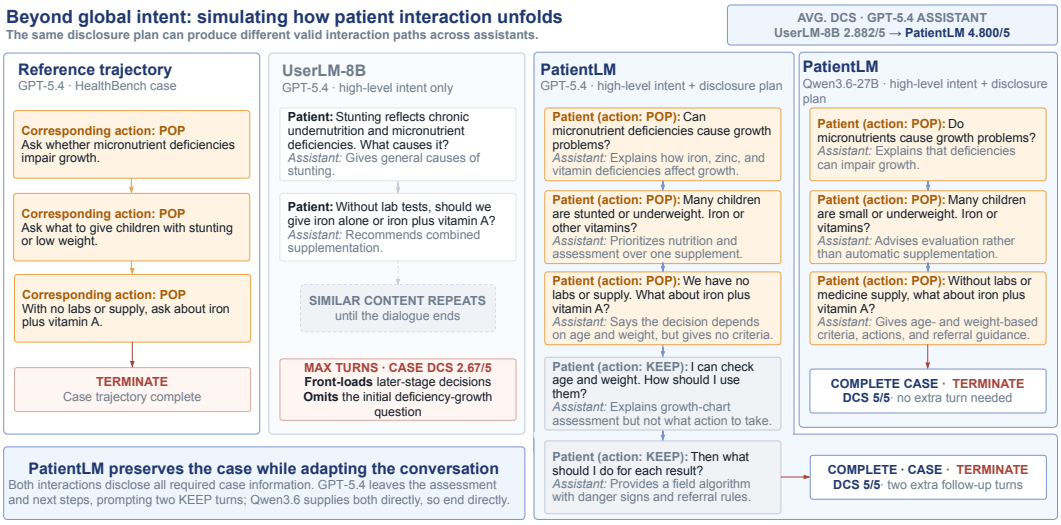


Figure 1: Representative interaction trajectories for the same medical scenario.

tended to cover. UserLM trains dedicated user language models conditioned on a high-level intent (Naous et al., 2025), whereas existing medical patient simulators use state tracking, structured personas, or selective disclosure to control patient behavior (Liao et al., 2024; Kyung et al., 2025; Schlager et al., 2026). Existing work, however, does not combine dedicated patient-side language modeling with fine-grained control over how medical information is disclosed across turns.

To meet these requirements, we introduce **PatientLM**, a patient simulator that combines a trained language model with a controller for medical multi-turn dialogue. PatientLM is guided by a *structured patient intent*, which extends the high-level intent used in UserLM (Naous et al., 2025) with a disclosure plan for medical interaction. The high-level patient intent captures the scenario-level patient profile, including the patient's role, medical concern, and communication style. The disclosure plan organizes case-specific clinical information and patient requests into ordered disclosure items and pairs them with item-specific contexts that guide whether the interaction remains on the current topic or advances to the next item, while a terminal context governs when the interaction ends. During simulation, the controller evaluates the evolving interaction against these contexts, and the language model expresses the information it releases as a natural patient utterance conditioned on the high-level intent and interaction history. This separation between dialogue progression and utterance generation allows PatientLM to preserve the same case while adapting to the assistant. Figure 1 presents a representative case in which PatientLM preserves the planned disclosures while adjusting its follow-up turns to different assistant responses.

Beyond modeling patient behavior, PatientLM supports both interactive evaluation and assistant training. For evaluation, we use it to convert static-context and dynamic-history medical benchmarks into a common adaptive protocol while retaining their original evaluation targets and scoring functions. This allows the same assistant to be compared when clinical context is provided directly and when it must be acquired through interaction. For training, we use PatientLM in a teacher-student setting, where a stronger teacher supplies target responses for fine-tuning a smaller student assistant. Offline training constructs complete PatientLM-teacher interactions before optimization, whereas online training lets the student interact with PatientLM and uses teacher responses to supervise the student at the dialogue states produced by those interactions.

Our experiments yield three main findings. First, PatientLM achieves the strongest overall simulator profile: it communicates case-specific clinical information and patient requests most completely while maintaining strong naturalness. Its advantage over the matched high-level-intent-only model highlights the benefit of controller-guided information disclosure. Second, fixed patient-side trajectories can overestimate medical-assistant performance: assistants generally score lower when the same cases are evaluated through adaptive interactions. Third, training on PatientLM interactions improves assistant factuality and reliability. Online training offers the best trade-off between factuality and reliability. This advantage may arise because PatientLM responds to the student's own outputs, thereby providing supervision on dialogue states induced by the student's behavior.

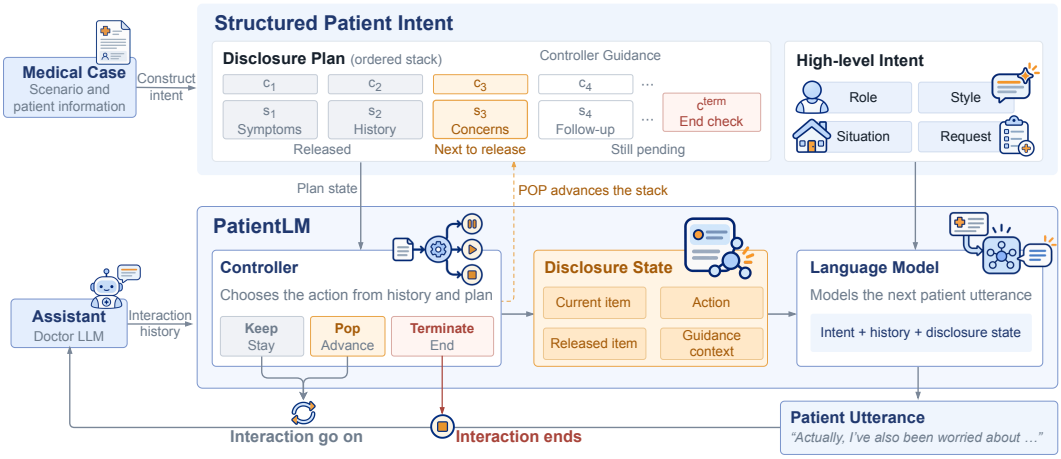


Figure 2: Overview of PatientLM.

2 PATIENTLM

This section presents **PatientLM**, a patient simulator for medical multi-turn interactions that combines a trained language model with a controller. For each medical interaction case (e.g., medical dialogue), we construct a structured patient intent that specifies the patient's scenario-level behavior and the information and requests to be disclosed. The controller uses this intent and the evolving interaction to decide when information becomes available or the interaction ends, while the language model generates patient utterances from the resulting disclosure state. We train the language model on patient turns paired with these disclosure states, then assess the complete simulator offline and in full interactions. Figure 2 illustrates the structured intent and the controller-guided interaction loop.

2.1 INSTANTIATING CASE-SPECIFIC PATIENTS WITH STRUCTURED INTENT

To sustain a coherent multi-turn interaction, the simulated patient should adapt to the dialogue while maintaining a consistent role, communication style, and medical scenario. In addition, medical dialogue adds a domain-specific requirement: patients often disclose clinically relevant information progressively, including symptoms, history, concerns, and follow-up requests. We therefore guide each simulation with a structured patient intent that combines a high-level patient intent for scenario-level behavior with a disclosure plan for the content and timing of turn-level disclosures.

Given a set $\mathcal{X} = \{x_i\}_i$ of medical interaction cases, where each case $x \in \mathcal{X}$ specifies the patient-side content and intended progression of a medical interaction, either as a multi-turn dialogue (Arora et al., 2025) or as a standardized patient case card containing clinical facts and consultation goals (Schmidgall et al., 2026), we construct corresponding structured patient intents $z = (h, \delta)$ in two stages, where h is the high-level intent of the patient and δ is a disclosure plan.

First, for each case, we extract intent $h = \phi(x)$, with ϕ implemented by an LLM using the high-level patient profile construction prompt in Appendix E.1. This prompt maps x to a high-level patient profile capturing the patient's role (e.g., patient or caregiver), medical concern (e.g., a pediatric feeding issue), and communication style (e.g., straightforward or anxious).

Second, we identify the information and requests to be disclosed during interaction. Let $\mathbf{u} = (u_1, \ldots, u_T)$ denote the ordered patient-side units derived from $x \in \mathcal{X}$, such as patient turns in a source dialogue or fields in a standardized patient case card (e.g., the presenting complaint, symptom history, and consultation goals), with T being the number of units. We adapt the functional taxonomy that HealthChat derives from user messages in real-world health conversations (Paruchuri et al., 2025). A unit can have more than one dialogic function. We denote the functions associated with u_t as $v_t \subset \mathcal{T}$, where $\mathcal{T} = \{\texttt{ctx}, \texttt{seek}, \texttt{mgmt}, \texttt{out}\}$ represents context provision, medical information or advice seeking, interaction management (e.g., follow-up questions), or out-of-scope content, respectively.

These functions play different roles in disclosure-plan construction. A unit u_t with $v_t = \{\texttt{out}\}$ generally does not contribute to the intended simulation objective and is excluded. A unit with `ctx` or `seek` in v_t provides clinical context and expresses requests for medical information or advice, respectively. We extract a disclosure item s_j from each such unit and arrange the resulting items in their intended interaction order to form the disclosure sequence $\mathbf{s} = (s_1, \ldots, s_M)$, where M is the number of items. A unit with $v_t = \{\texttt{mgmt}\}$ captures clarifications, corrections, or unresolved follow-ups concerning a previously disclosed item or the assistant's response to it. These units provide cues for constructing the controller contexts $\mathbf{c} = (c_1, \ldots, c_M)$, where c_j describes the current communication state and guides whether the interaction should remain on the current topic or advance to the next item. Once the last item has been released, the controller uses a separate terminal context c_{term} to decide whether the interaction should continue or end.

The disclosure sequence, controller contexts, and terminal context define the disclosure plan $\delta = (\mathbf{s}, \mathbf{c}, c_{\text{term}})$ for x. Together with h, this forms the structured patient intent z.

2.2 Patient Simulation with Controller-Guided Disclosure

During simulation, the controller uses z and the evolving interaction history to decide when to disclose the next disclosure item or end the interaction. When the interaction continues, the language model generates the next patient utterance using the available information and history.

At simulation turn t, let $\hat{H}_t = (\hat{u}_1, \hat{r}_1, \ldots, \hat{u}_{t-1}, \hat{r}_{t-1})$ denote the generated interaction history, where $\hat{u}_t$ and $\hat{r}_t$ are an utterance from PatientLM and the assistant's response to it. Suppose s_j is the next pending disclosure item and $\mathbf{s}_{<j} = (s_1, \ldots, s_{j-1})$ contains the items already released. The controller uses the patient intent h, interaction history $\hat{H}_t$, released items $\mathbf{s}_{<j}$, the last released item's context c_{j-1} (with $c_0 = \emptyset$), the next pending item s_j, and its context c_j to select an action from $\mathcal{A}_{\text{pending}} = \{\text{keep}, \text{pop}\}$:

$$a_t = \pi\Big(h, \hat{H}_t, \mathbf{s}_{<j}, c_{j-1}, s_j, c_j\Big). \tag{1}$$

If $a_t = \text{pop}$, the controller releases s_j as $\bar{s}_t$; otherwise, s_j remains pending and $\bar{s}_t = \emptyset$, so the next patient turn draws only on previously disclosed information.

Once all disclosure items have been released, the controller instead selects an action from $\mathcal{A}_{\text{term}} = \{\text{keep}, \text{terminate}\}$:

$$a_t = \pi_{\text{term}}\Big(h, \hat{H}_t, \mathbf{s}, c_M, c_{\text{term}}\Big). \tag{2}$$

If $a_t = \text{keep}$, PatientLM produces an additional turn using previously disclosed information; otherwise, the interaction ends without another patient turn.

For each nonterminal pending-item action, the controller provides the language model with a turn-level disclosure state $d_t = (c_j, a_t, \bar{s}_t)$; after all items have been released, a terminal-state keep action instead gives $d_t = (c_{\text{term}}, \text{keep}, \emptyset)$. The language model then samples the next patient utterance as

$$\hat{u}_t \sim p_\theta\Big(\cdot \mid h, \hat{H}_t, d_t\Big). \tag{3}$$

Here, h anchors scenario-level behavior, $\hat{H}_t$ supplies the preceding interaction, and d_t specifies the current controller guidance and released information. After the language model generates $\hat{u}_t$, the assistant model $\mathcal{M}$ produces $\hat{r}_t$, and the resulting patient-assistant pair is appended to $\hat{H}_t$. The updated history informs the next controller decision until terminate or the maximum turn limit.

2.3 Training the Patient Language Model

To enable PatientLM's language model to use the high-level patient intent h and turn-level disclosure state d_t during generation, we train it on HealthChat conversations (Paruchuri et al., 2025) to predict each patient utterance given these inputs and the preceding interaction. We use each source conversation and its disclosure plan to assign a controller action and released item to each patient turn u_t. A content-bearing turn is assigned $a_t = \text{pop}$ and $\bar{s}_t = s_j$, whereas an interaction-management turn is assigned $a_t = \text{keep}$ and $\bar{s}_t = \emptyset$. Together with the corresponding disclosure item's context c_j, these assignments form d_t for predicting u_t.

Table 1: Offline simulator diagnostics adapted from UserLM.

Metric	Definition
Perplexity (PPL ↓)	Distributional fit to held-out human user utterances.
First-turn diversity (FTD ↑)	Unique unigram rate across generated first patient turns.
Intent decomposition (ID)	Fraction of distinct high-level-intent unigrams appearing in generated patient turns, after stopword removal.
End F1 (F_1^{end} ↑)	Binary F1 for predicting whether the user should terminate the conversation at an eligible state.
Naturalness (Nat ↑)	Likelihood that generated user utterances resemble human-written text.
Role adherence (RA ↑)	Rate at which the simulator maintains the user role under assistant prompts that invite role collapse.
Intent adherence (IA ↑)	Rate at which the simulator stays with its original goal when the assistant suggests an off-goal direction.

We serialize the annotated conversations as complete interaction trajectories, conditioning each patient utterance on earlier patient turns, assistant responses, and controller states. Only patient utterance tokens contribute to the loss; we retain purely out-of-scope turns and interaction-management turns before the first disclosure item is released as context but exclude them from supervision. Let $\mathcal{D}_{\text{patient}}$ denote the resulting set of serialized trajectories. We define a mask μ_ℓ for a tokenized sequence $w_{1:L}$ ($\ell = 1, \ldots, L$): $\mu_\ell = 1$ when w_ℓ belongs to a supervised patient utterance and 0 otherwise. The training objective is

$$\mathcal{L}_{\text{patient}} = -\mathbb{E}_{w_{1:L} \sim \mathcal{D}_{\text{patient}}} \left[\frac{1}{\sum_{\ell=1}^{L} \mu_\ell} \sum_{\ell=1}^{L} \mu_\ell \log p_\theta(w_\ell \mid w_{<\ell}) \right]. \tag{4}$$

2.4 Simulator Evaluation Metrics

We evaluate simulator quality on fixed held-out inputs (offline) and in complete interactions with an assistant (online). Following UserLM (Naous et al., 2025), offline evaluation uses perplexity (PPL) for distributional fit and six diagnostics for multi-turn behavior: first-turn diversity (FTD), intent decomposition (ID), end F1 (F_1^{end}), naturalness (Nat), role adherence (RA), and intent adherence (IA), as summarized in Table 1.

These offline metrics do not measure case-level coverage of clinical information and patient requests across a dialogue, which is essential for medical-assistant evaluation and training-data generation. We therefore introduce the disclosure item coverage score (DCS) for online evaluation in closed-loop interactions. Each simulator interacts with the same assistant model, and a judge assigns each disclosure item s_j a coverage score $q_j \in \{0, \ldots, 5\}$ using only the simulator's patient utterances. For each case, we compute the mean over its disclosure items as

$$\text{DCS} = \frac{1}{M} \sum_{j=1}^{M} q_j. \tag{5}$$

Higher DCS indicates better coverage of the case-specific clinical information and patient requests.

3 Applications: Evaluation and Training with PatientLM

3.1 Evaluation with PatientLM

Let $\mathcal{B} = \{b\}$ denote a source benchmark, where each benchmark instance $b = (x, \mathcal{E})$ consists of a medical interaction case x and K evaluation points $\mathcal{E} = \{(u_j^\star, y_j)\}_{j=1}^{K}$. Here, $u_j^\star$ is a target patient request, and y_j is its official evaluation target, such as a rubric or reference facts. We construct $z = (h, \delta)$ from x while retaining $\mathcal{E}$ as the original evaluation specification.

We use PatientLM to bring distinct multi-turn evaluation paradigms into a common interactive setting. For static-context benchmarks, x is the gold-context dialogue history, from which we construct z. For dynamic-history benchmarks, x is the prescribed patient-turn sequence $\mathbf{u}$, whose intended progression is retained in δ. We instantiate these two settings with HealthBench (Arora et al., 2025) and the medical subset of KnowMT-Bench (Chen et al., 2025), respectively.

Given z, PatientLM interacts with the evaluated assistant $\mathcal{M}$ until termination, yielding a complete patient trajectory $\hat{u}_{1:\hat{T}}$, where $\hat{T}$ is the number of generated patient turns. Because PatientLM may paraphrase a target request, distribute it across patient turns, or raise it earlier than in the source benchmark, we align each target request with this trajectory as $I_j = \mathrm{Align}(u_j^\star, \hat{u}_{1:\hat{T}})$, where I_j contains the ordered indices of the patient turns that collectively express $u_j^\star$.

To distinguish simulator coverage from assistant response quality, no assistant score is assigned when $I_j = \emptyset$. Otherwise, we concatenate, in dialogue order, the assistant responses following the patient turns indexed by I_j and denote the result by $\hat{R}_j$. The aligned evaluation result is $m_j = \mathrm{Score}_{\mathcal{B}}(\hat{R}_j, y_j)$, where $\mathrm{Score}_{\mathcal{B}}$ applies the source benchmark's scoring rule: HealthBench uses rubric items, while KnowMT-Bench computes factuality and hallucination metrics. Thus, m_j may contain more than one metric.

3.2 Training with PatientLM

Beyond evaluation, structured patient intents and controller-guided interactions can provide supervision for assistant models in the student-teacher framework. Offline SFT constructs a fixed corpus of PatientLM–teacher trajectories before student optimization, whereas online SFT lets the student interact with PatientLM, with a teacher providing a target response at each turn.

Offline SFT. Before optimizing the student assistant, a complete teacher-supervised trajectory $\tilde{H} = (\tilde{u}_1, r_1^\star, \ldots, \tilde{u}_{\tilde{T}}, r_{\tilde{T}}^\star)$ is constructed for each case, where $\tilde{u}_t$ is a generated patient utterance and $r_t^\star$ is the corresponding teacher response. We collect trajectories for all $x_i \in \mathcal{X}$ to form $\mathcal{D}_{\mathrm{off}} = \{\tilde{H}_i\}_{i=1}^{|\mathcal{X}|}$. The resulting fixed trajectories are subsequently used to train the student, with every teacher response serving as a supervision target.

Online SFT. Online SFT places the student assistant in the interaction loop. The history $\hat{H}_t = (\hat{u}_1, \hat{r}_1, \ldots, \hat{u}_{t-1}, \hat{r}_{t-1})$ contains previous PatientLM utterances $\hat{u}_{1:t-1}$ and responses $\hat{r}_{1:t-1}$ generated by the student assistant p_τ. After the disclosure controller guides PatientLM to generate $\hat{u}_t$, the student samples a new response $\hat{r}_t \sim p_\tau(\cdot \mid \hat{H}_t, \hat{u}_t)$ and appends it to the interaction history, thereby shaping subsequent patient turns. Given the same history and latest patient utterance, a stronger teacher model p_{teach} generates the supervision target $r_t^\star \sim p_{\mathrm{teach}}(\cdot \mid \hat{H}_t, \hat{u}_t)$. The teacher target supervises the student at the current dialogue state but is not appended to the history used in subsequent interaction. The resulting turn-level training set for all $x_i \in \mathcal{X}$ is

$$\mathcal{D}_{\mathrm{on}} = \left\{ \left(\hat{H}_{it}, \hat{u}_{it}, r_{it}^\star \right) \right\}_{i,t}. \tag{6}$$

Because later PatientLM turns respond to the student's own outputs, $\mathcal{D}_{\mathrm{on}}$ covers dialogue states induced by the student's current behavior, making the training environment closer to the conditions it encounters in actual multi-turn interaction.

For either $\mathcal{D} \in \{\mathcal{D}_{\mathrm{off}}, \mathcal{D}_{\mathrm{on}}\}$, each trajectory or turn-level example is serialized as a message sequence $w_{1:L}$. Let $\mu_\ell^{\mathrm{asst}} = 1$ when w_ℓ belongs to a target assistant response and 0 otherwise. The student is trained with

$$\mathcal{L}_{\mathrm{SFT}}(\mathcal{D}) = -\mathbb{E}_{w_{1:L} \sim \mathcal{D}} \left[\frac{\sum_{\ell=1}^{L} \mu_\ell^{\mathrm{asst}} \log p_\tau(w_\ell \mid w_{<\ell})}{\sum_{\ell=1}^{L} \mu_\ell^{\mathrm{asst}}} \right], \tag{7}$$

where only target assistant-response tokens contribute to the loss.

4 Experiments

4.1 Experimental Settings

To train PatientLM, we use Qwen3.5-9B[1] as the base model and 10,518 medical cases from HealthChat (Paruchuri et al., 2025), a dataset compiled from LMSYS and WildChat conversations.

[1] https://huggingface.co/Qwen/Qwen3.5-9B

Table 2: Simulator results. Bold and underlined values mark the best and second-best directional scores. ID is descriptive; GPT-5.4 perplexity is unavailable through the API.

Simulator	Offline							Online
	PPL ↓	FTD ↑	ID	F_1^{end} ↑	Nat ↑	RA ↑	IA ↑	DCS ↑
Qwen3.5-9B	18.147	<u>0.223</u>	0.218	0.174	0.564	0.953	0.640	2.526
Qwen3.6-27B	7.149	0.155	0.423	0.415	0.357	0.833	**0.993**	<u>3.737</u>
GPT-5.4	–	0.173	0.286	0.220	0.497	0.618	<u>0.992</u>	3.659
UserLM-8B	8.104	0.209	0.341	0.353	0.705	0.815	0.732	2.882
PatientSimulator	36.983	0.068	0.124	0.211	0.417	0.902	0.660	1.911
UserLM-9B	<u>4.659</u>	**0.247**	0.271	**0.924**	<u>0.755</u>	<u>0.995</u>	0.948	3.380
PatientLM	**4.164**	**0.247**	0.224	<u>0.890</u>	**0.771**	**0.996**	0.982	**4.800**

After splitting the cases into training and validation sets at a 3:1 ratio, we fine-tune the base model for one epoch using AdamW, with a learning rate of 10^{-5}, a global batch size of 32, and a maximum sequence length of 4,096. For a controlled comparison, we train UserLM-9B using the same training split and backbone for the same number of steps, but condition it only on high-level intent.

We evaluate PatientLM against six baselines using offline simulator diagnostics and interactive case completion: UserLM-9B, prompted Qwen3.5-9B and Qwen3.6-27B[2], GPT-5.4[3] prompted as a patient, the trained simulator UserLM-8B[4], and PatientSimulator[5] (Liu et al., 2025). Following UserLM (Naous et al., 2025), we apply seven offline diagnostics on 1,000 examples each, adapting its dataset choices and evaluation procedures to patient simulation. For interactive evaluation, all simulators use the same 1,000 HealthBench cases and interact with GPT-5.4 as the assistant; we report the mean case-level DCS across these cases. Training and evaluation were conducted on a server equipped with eight NVIDIA RTX A6000 GPUs. Further details are provided in Appendix B.

4.2 User Simulation Performance

As shown in Table 2, PatientLM achieves the highest DCS, substantially exceeding both prompted and trained baselines, which indicates more complete coverage of case-specified content during interaction. Across the offline metrics, PatientLM achieves the highest naturalness and role adherence, as well as the lowest perplexity, while tying for the best first-turn diversity and approaching the best end F1 and intent adherence. Its moderate intent-decomposition overlap, alongside high DCS, suggests a balance between lexical overlap with intent and case coverage. Overall, PatientLM has a strong overall profile across online and offline evaluations.

UserLM-9B outperforms the released UserLM-8B, trained from Llama3-8B-Base (Naous et al., 2025), on every directional metric. This gain may partly reflect the benefit of a newer backbone. In contrast, PatientLM attains higher DCS, naturalness, and intent adherence and lower perplexity than the matched UserLM-9B, suggesting that structured patient intent and controller-guided disclosure benefit both case coverage and patient-language modeling beyond high-level intent alone.

4.3 Controller Performance

Table 3: Action F1.

Action	F1
`pop`	0.976
`keep`	0.294
`terminate`	0.918
Macro-F1	0.729

PatientLM uses Qwen3.6-27B as an independent prompt-based controller. We evaluate the controller on the full HealthChat dataset. As shown in Table 3, it achieves strong F1 scores for the `pop` and `terminate` actions, but performs substantially worse on `keep`.

Two properties help explain this gap. First, `keep` accounts for only 4.6% of the decisions, so false positives from the much larger `pop` and `terminate` classes can quickly lower its precision. Second, `keep` is intrinsically variable in real human assistant interactions: given similar dialogue histories and disclosure states, a patient may either elaborate on the current information or move to the next disclosure item. This action is therefore not fully determined by

[2] https://huggingface.co/Qwen/Qwen3.6-27B

[3] https://developers.openai.com/api/docs/models/gpt-5.4

[4] https://huggingface.co/microsoft/UserLM-8b

[5] https://huggingface.co/zhaocheng/patient_simulator

Table 5: Original versus interactive evaluation. KnowMT-Bench's $\mathbf{S}_f$ and $\mathbf{S}_h$ are factual and hallucination F1. For each assistant, all original and interactive performance values use the same cases valid under both protocols; alignment uses the full benchmark.

Assistant	HealthBench			KnowMT-Bench				
	Score (%) ↑			$\mathbf{S}_f$ (%) ↑		$\mathbf{S}_h$ (%) ↓		
	Orig.	Inter.	Align. (%)	Orig.	Inter.	Orig.	Inter.	Align. (%)
Qwen3.5-9B	33.400	28.100	94.600	25.200	27.800	0.950	1.070	99.000
Qwen3.6-27B	41.100	34.400	94.800	25.100	24.300	1.110	1.210	99.000
GPT-5.4	41.600	36.800	95.500	32.700	32.000	0.390	0.490	99.500

the observable context. We test this ambiguity with a kNN diagnostic ($k = 20$): if the actions were clearly separated, nearby inputs would usually share the same label. However, full-input kNN achieves only 0.365 `keep` F1, while history-only kNN falls to 0.049, indicating that the boundary between `keep` and the other actions is highly complex and partly stochastic.

4.4 Hyperparameter Sensitivity Analysis

Table 4: Patient-token perplexity on the HealthChat validation split.

Learning rate	Epoch 1	Epoch 2
5×10^{-6}	4.041	4.072
1×10^{-5}	**4.031**	4.148
2×10^{-5}	4.243	4.283

We analyze PatientLM's sensitivity to learning rate and training duration on the HealthChat validation split. We compare learning rates of $\{5 \times 10^{-6}, 10^{-5}, 2 \times 10^{-5}\}$ after one and two epochs. As shown in Table 4, validation perplexity is lowest with a learning rate of 10^{-5} after one epoch. Extending training to two epochs increases perplexity at every learning rate, while training for four epochs at 10^{-5} further increases it to 5.738, indicating overfitting. We therefore use a learning rate of 10^{-5} and train for one epoch in the main experiments.

5 Impact on Evaluation and Training

5.1 Evaluation Protocol Sensitivity

We examine how assistant performance changes under the interactive evaluation framework introduced in Section 3.1. This conversion preserves each benchmark's original evaluation points and scoring rule while replacing its source interaction protocol with PatientLM-based interaction. We compare the original and interactive protocols for Qwen3.5-9B, Qwen3.6-27B, and GPT-5.4 on 1,000 HealthBench cases and all 201 medical KnowMT-Bench cases (Arora et al., 2025; Chen et al., 2025). For each assistant, scores are compared only on cases valid under both protocols, while target-alignment coverage is reported over the full benchmark.

As shown in Table 5, all three assistants score lower on HealthBench under interactive evaluation. Paired nonparametric permutation tests (Ernst, 2004) with Holm correction (Holm, 1979) confirm that these decreases are statistically significant. On KnowMT-Bench, statistical significance is not consistent across models and metrics, but most comparisons likewise indicate degraded performance. Overall, assistant performance tends to be lower under the interactive protocol than under the original static-context and dynamic-history protocols, suggesting that adaptive information disclosure exposes challenges not captured by the source protocols.

5.2 Assistant Training with PatientLM

We next examine whether PatientLM can facilitate assistant training by constructing adaptive medical interactions. We train Qwen2.5-7B[6] as the student, using Qwen3.6-27B as teacher. For all three training regimes, supervision is constructed from 1,000 HealthBench cases, with 950 used for training and 50 for validation. As a standard distillation baseline, we fine-tune the student on complete dialogues generated by the teacher in one pass (Ding et al., 2023; Xu et al., 2025). We compare this baseline with the offline and online PatientLM-based SFT schemes introduced in Section 3.2: the former uses fixed PatientLM-teacher trajectories, whereas the latter generates teacher targets along

[6] `https://huggingface.co/Qwen/Qwen2.5-7B-Instruct`

student-induced trajectories. We evaluate on the held-out medical subset of KnowMT-Bench using factual recall $\mathbf{R}_f$ (↑) and hallucination rate (recall) $\mathbf{R}_m$ (↓). We focus on these recall-based metrics because training targets fact recovery rather than response precision.

Teacher-generated SFT increases hallucination and lowers factuality relative to the untrained student. In contrast, both PatientLM-based regimes improve factual recall and reduce hallucination, with online SFT attaining the lowest hallucination rate among the trained students while maintaining strong factuality. These results suggest that SFT supervision collected through PatientLM-mediated interactions is more effective than supervision derived from one-pass teacher-generated dialogues, with the online regime offering the best trade-off between factuality and reliability. This advantage may be attributed to online SFT exposing the student to the consequences of its own responses: PatientLM adapts subsequent turns accordingly, while the teacher provides supervision at the resulting dialogue states, thereby reducing the mismatch between training and deployment interactions.

Table 6: KnowMT-Bench factual recall and hallucination rate (recall), averaged over the 191 valid samples aligned across all five models or training regimes.

Model or training regime	R_f ↑	R_m ↓
Qwen3.6-27B	0.646	0.018
Qwen2.5-7B	0.447	0.031
Teacher-generated SFT	0.384	0.043
PatientLM–teacher offline	0.592	0.026
Student-in-the-loop online	0.579	0.019

6 Analysis

To understand why PatientLM achieves higher DCS than UserLM-8B and the matched UserLM-9B in Table 2, we analyze their progression behavior on the same 1,000 HealthBench cases. Figure 3 groups cases by the number of disclosure items and reports strict completion, which requires every item in a case to receive a score of 5. Table 7 summarizes the corresponding interaction behavior and item-level scores.

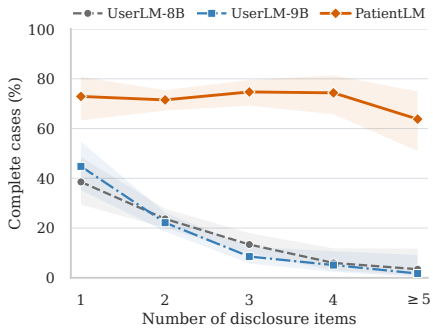


Figure 3: Strict case completion as the disclosure plan grows. Shading denotes 95% Wilson confidence intervals.

These two views reveal a consistent difference in progression behavior. UserLM-8B frequently stalls, reaching the 15-turn limit in 80.5% of cases. UserLM-9B reduces this rate to 3.6%, but its completion rate falls from 22.2% for two-item cases to 1.7% for cases with at least five items. PatientLM reaches the turn limit in only 1.2% of cases and remains substantially more robust as the disclosure plan grows, with completion decreasing from 71.5% to 63.8%. It averages 3.99 patient turns, compared with 3.45 for UserLM-9B, reflecting the additional adaptive follow-up enabled by the dynamic controller. Because UserLM-9B uses the same backbone and training scale, this comparison supports the combined contribution of structured patient intent and controller-guided disclosure: explicitly tracking pending content and separating disclosure decisions from language generation improve content coverage as the interaction plan becomes more complex. The trajectory in Appendix D provides a qualitative example of this difference in progression behavior.

Table 7: Simulator behavior on the HealthBench.

Behavior	UserLM-8B	UserLM-9B	PatientLM
Avg. turns	12.704	3.452	3.987
Turn limit (%)	80.500	3.600	1.200
First item	2.861	3.341	4.841
Interior item	2.077	2.485	4.886
Last item	3.103	3.661	4.746
Complete cases (%)	19.100	17.500	72.400

7 Conclusion

We introduced PatientLM, a controllable patient simulator that combines high-level patient intent with disclosure items and controller-guided progressive disclosure. This design supports adaptive evaluation by converting static-context and dynamic-history medical benchmarks into interactive protocols, and supports assistant training through PatientLM-mediated supervision. Across our experiments, PatientLM improves case-specific information coverage, reveals performance differences that fixed patient trajectories may hide, and provides more effective supervision for multi-turn medical assistants, highlighting controllable patient simulators as a promising foundation for advancing interactive medical AI.

Reproducibility Statement

The paper describes the PatientLM architecture, structured-intent construction, controller states, patient-only training objective, and benchmark-conversion procedure in Sections 2 and 3.1. Appendix B records the source data, case splits, model configurations, training settings, evaluation protocols, and alignment procedure used in the experiments. The code and data used in our experiments will be provided in the supplementary materials.

Ethics Statement

This work does not collect new human-subject data or provide medical advice. It studies patient simulation using existing medical interaction data and benchmark cases, which may contain sensitive health-related content. We use these resources only for research evaluation and training-data generation, follow their applicable access and use conditions, and report aggregate results rather than attempting to identify individuals. PatientLM and the generated interactions can contain factual errors and are not intended for clinical decision-making or direct patient use; human oversight remains necessary when interpreting or deploying the resulting models.

AI Use Statement

Generative AI tools were used in this work to polish writing, retrieve and discover related work, support discussions of research ideas proposed by the authors, assist with code implementation, and generate synthetic data. Details of the model design and experimental procedures are provided in Sections 2–4, and the PatientLM-mediated synthetic interaction data used for assistant training are described in Section 3.2 and Appendix B. The authors take responsibility for the final content of this submission.

References

Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health. *CoRR*, abs/2505.08775, 2025. doi: 10.48550/ARXIV.2505.08775. URL https://doi.org/10.48550/arXiv.2505.08775.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 7421–7454. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.401. URL https://doi.org/10.18653/v1/2024.acl-long.401.

Serina Chang, Ashton Anderson, and Jake M. Hofman. Chatbench: From static benchmarks to human-ai evaluation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 26009–26038. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1262. URL https://doi.org/10.18653/v1/2025.acl-long.1262.

Junhao Chen, Yu Huang, Siyuan Li, Rui Yao, Hanqian Li, Hanyu Zhang, Jungang Li, Jian Chen, Bowen Wang, and Xuming Hu. Knowmt-bench: Benchmarking knowledge-intensive long-form question answering in multi-turn dialogues. *CoRR*, abs/2509.21856, 2025. doi: 10.48550/ARXIV.2509.21856. URL https://doi.org/10.48550/arXiv.2509.21856.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversa-

tions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

Michael D Ernst. Permutation methods: a basis for exact inference. *Statistical Science*, pp. 676–685, 2004.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Florian Kreyssig, Iñigo Casanueva, Pawel Budzianowski, and Milica Gasic. Neural user simulation for corpus-based policy optimisation of spoken dialogue systems. In Kazunori Komatani, Diane J. Litman, Kai Yu, Lawrence Cavedon, Mikio Nakano, and Alex Papangelis (eds.), *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, Melbourne, Australia, July 12-14, 2018*, pp. 60–69. Association for Computational Linguistics, 2018. doi: 10.18653/V1/W18-5007. URL https://doi.org/10.18653/v1/w18-5007.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/TACL_A_00276. URL https://doi.org/10.1162/tacl_a_00276.

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. *CoRR*, abs/2505.17818, 2025. doi: 10.48550/ARXIV.2505.17818. URL https://doi.org/10.48550/arXiv.2505.17818.

Huihan Li, Tianyu Gao, Manan Goenka, and Danqi Chen. Ditch the gold standard: Re-evaluating conversational question answering. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 8074–8085. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.ACL-LONG.555. URL https://doi.org/10.18653/v1/2022.acl-long.555.

Xiujun Li, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues. *CoRR*, abs/1612.05688, 2016. URL http://arxiv.org/abs/1612.05688.

Yusheng Liao, Yutong Meng, Yuhao Wang, Hongcheng Liu, Yanfeng Wang, and Yu Wang. Automatic interactive evaluation for large language models with state aware patient simulator. *CoRR*, abs/2403.08495, 2024. doi: 10.48550/ARXIV.2403.08495. URL https://doi.org/10.48550/arXiv.2403.08495.

Zhaocheng Liu, Quan Tu, Wen Ye, Yu Xiao, Zhishou Zhang, Hengfu Cui, Yalun Zhu, Qiang Ju, Shizheng Li, and Jian Xie. Exploring the inquiry-diagnosis relationship with advanced patient simulators. *CoRR*, abs/2501.09484, 2025. doi: 10.48550/ARXIV.2501.09484. URL https://doi.org/10.48550/arXiv.2501.09484.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. Duetsim: Building user simulator with dual large language models for task-oriented dialogues. In Nicoletta Calzolari, Min-Yen Kan, Véronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation, LREC/COLING 2024, 20-25 May, 2024, Torino, Italy*, pp. 5414–5424. ELRA and ICCL, 2024. URL https://aclanthology.org/2024.lrec-main.481.

Mikio Nakano, Kazunori Komatani, and Hironori Takeuchi. Generating diverse personas for user simulators to test interview dialogue systems. In Frédéric Béchet, Fabrice Lefèvre, Nicholas Asher, Seokhwan Kim, and Téva Merlin (eds.), *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2025, Avignon, France, August 25-26, 2025*, pp. 678–696. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.sigdial-1.54/.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. *CoRR*, abs/2510.06552, 2025. doi: 10.48550/ARXIV.2510.06552. URL https://doi.org/10.48550/arXiv.2510.06552.

Akshay Paruchuri, Maryam Aziz, Rohit Vartak, Ayman Ali, Best Uchehara, Xin Liu, Ishan Chatterjee, and Monica Agrawal. "what's up, doc?": Analyzing how users seek health information in large-scale conversational AI datasets. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 2312–2336. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-EMNLP.125. URL https://doi.org/10.18653/v1/2025.findings-emnlp.125.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve J. Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In Candace L. Sidner, Tanja Schultz, Matthew Stone, and ChengXiang Zhai (eds.), *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, April 22-27, 2007, Rochester, New York, USA*, pp. 149–152. The Association for Computational Linguistics, 2007. URL https://aclanthology.org/N07-2038/.

Moritz Schlager, Friederike Jungmann, Samuel Schmidgall, Philipp Raffler, Franziska Hartl, Eva Wende, Paula Roßmüller, Conrad Ketzer, Avinatan Hassidim, Dale R. Webster, Yossi Matias, Yun Liu, Daniel Rueckert, Mike Schaekermann, and Paul Hager. Patients with personality: Realistic patient simulation through controlled diversity and selective disclosure. *CoRR*, abs/2606.17441, 2026. doi: 10.48550/ARXIV.2606.17441. URL https://doi.org/10.48550/arXiv.2606.17441.

Samuel Schmidgall, Rojin Ziaei, Carl Harris, Ji Woong Kim, Eduardo Pontes Reis, Jeffrey Jopling, and Michael Moor. Agentclinic: a multimodal benchmark for tool-using clinical AI agents. *npj Digit. Medicine*, 9(1), 2026. doi: 10.1038/S41746-026-02674-7. URL https://doi.org/10.1038/s41746-026-02674-7.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. *Nature*, 623(7987):493–498, 2023.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Mohamed Amin, Le Hou, Kevin Clark, Stephen R Pfohl, Heather Cole-Lewis, et al. Toward expert-level medical question answering with large language models. *Nature Medicine*, 31(3):943–950, 2025.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4149–4158. Association for Computational Linguistics, 2019. doi: 10.18653/V1/N19-1421. URL https://doi.org/10.18653/v1/n19-1421.

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. Know you first and be you better: Modeling human-like user simulators via implicit profiles. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 21082–21107. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1025. URL https://doi.org/10.18653/v1/2025.acl-long.1025.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Pnk7vMbznK.

Feng Zhang, Shijia Li, Chunmao Zhang, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, Jingwen Xu, and Han Liu. Userlm-r1: Modeling human reasoning in user language models

with multi-reward reinforcement learning. *CoRR*, abs/2601.09215, 2026. doi: 10.48550/ARXIV.2601.09215. URL https://doi.org/10.48550/arXiv.2601.09215.

A Related Work

PatientLM builds on two lines of work: multi-turn dialogue evaluation and user simulation. We review them in turn, with particular attention to how evaluation histories are constructed and how simulated users control information disclosure.

Multi-turn dialogue evaluation. Existing multi-turn benchmarks commonly adopt a static evaluation protocol, in which the benchmark provides the dialogue preceding a target user turn and evaluates the model's subsequent response. For example, HealthBench evaluates responses to the final user message in a preconstructed healthcare conversation, while MT-Bench-101 assesses fine-grained conversational abilities over curated multi-turn histories (Arora et al., 2025; Bai et al., 2024). This protocol enables controlled comparisons under identical contexts, but does not capture how a model's earlier responses affect its subsequent behavior. Evaluation over benchmark-provided histories may also differ substantially from evaluation in live human–model interactions (Li et al., 2022). KnowMT-Bench moves toward more realistic multi-turn evaluation through a dynamic protocol, in which the evaluated model answers a predetermined sequence of progressively related questions and generates the assistant-side history used in later turns (Chen et al., 2025). This setting exposes error propagation and contextual noise introduced by the model's own prior outputs, although the user-side trajectory remains fixed by the benchmark. Interactive evaluation takes a further step by allowing future user turns to depend on the evolving dialogue and the model's behavior. ChatBench converts static benchmark questions into human–AI conversations and trains a user simulator to scale such evaluation (Chang et al., 2025). Complementing this benchmark-oriented direction, UserLM systematically formulates user-side language and behavior as dedicated modeling objectives and post-trains purpose-built user language models for interactive evaluation (Naous et al., 2025). Viewed through how dialogue histories are constructed, these settings form a progression toward increasingly realistic multi-turn evaluation: from benchmark-provided histories, to model-generated assistant histories, and ultimately to interaction trajectories jointly generated by both participants. Building on this progression, we propose a general framework for converting existing multi-turn dialogue benchmarks under static or dynamic protocols into the interactive evaluation paradigm. We instantiate this framework in medicine, shifting evaluation from responding to a provided clinical history toward actively constructing the clinical context through follow-up interaction, thereby more closely reflecting real-world consultations.

User simulation. Traditional user simulators rely on structured agenda- or rule-based policies or learn user behavior from task-oriented dialogue corpora, offering explicit goal control but remaining closely tied to predefined domains and task schemas (Schatzmann et al., 2007; Kreyssig et al., 2018; Li et al., 2016). Advances in the instruction-following and context-conditioned generation capabilities of LLMs have shifted recent work toward prompt-conditioned user simulation, where general-purpose LLMs are instantiated with user profiles, personas, or task goals to produce flexible multi-turn behavior (Shanahan et al., 2023; Luo et al., 2024; Nakano et al., 2025). Recent medical simulators therefore augment prompted LLMs with domain-specific control mechanisms: SAPS introduces state-aware information retrieval, PatientSim uses structured clinical prompts that jointly specify patient profiles, personas, and behavioral constraints, and PatientsWithPersonality controls behavioral diversity and selective information disclosure (Liao et al., 2024; Kyung et al., 2025; Schlager et al., 2026).

A complementary line of work moves beyond prompting and trains dedicated user simulators for goal-consistent task-oriented interaction and scalable interactive evaluation (Schlager et al., 2026; Wang et al., 2025; Chang et al., 2025). In the medical domain, Liu et al. (2025) extract patient interaction strategies from real doctor-patient conversations and use them to train a patient simulator that better reflects realistic patient behavior. Their experiments further demonstrate the importance of the inquiry phase by showing that inadequate information elicitation can constrain subsequent diagnostic performance. UserLM systematizes the broader trained-user direction by formulating user-side language and behavior as dedicated language-modeling objectives. It further identifies a mismatch between assistant-oriented LMs, which are post-trained to provide helpful and complete

responses, and human users, who often communicate incompletely, reveal their intentions incrementally, and adapt throughout an interaction (Naous et al., 2025). UserLM-R1 extends this paradigm by strengthening goal-driven reasoning on the user side (Zhang et al., 2026). Motivated by the partial and progressive nature of medical consultations, PatientLM combines dedicated patient modeling with explicit control over progressive information disclosure. Specifically, we train a medically specialized user model to capture patient-side language and interaction behavior, while an external controller tracks the revealed patient state and determines whether each response should rely on previously disclosed information or introduce additional clinical evidence.

B Detailed Experimental Settings

B.1 PatientLM Training and Simulator Evaluation

From HealthChat, we retain conversations assigned to a medical specialty that contain at least one content-bearing A/B turn and in which D-only turns account for no more than 60% of the annotated turns. The resulting data comprise 6,516 LMSYS cases and 4,002 WildChat cases. We construct structured patient intents and turn-level disclosure states from the source conversations as described in Section 2. D-only turns and prefix keep turns preceding the first pop are retained as context but excluded from the training loss. A case-level random split with seed 42 partitions the cases into training and validation sets at a 3:1 ratio. We fine-tune the full Qwen3.5-9B model with FSDP on eight NVIDIA RTX A6000 GPUs to obtain PatientLM, applying loss only to trainable patient-utterance tokens. The main checkpoint is taken after one epoch (step 246). Training uses AdamW, a learning rate of 10^{-5} with a cosine schedule and 20 warmup steps, a global batch size of 32 processed in per-GPU micro-batches of 2, and a maximum sequence length of 4,096. UserLM-9B uses the same backbone, source cases, data split, and training scale, but receives only high-level intent.

The simulator comparison includes prompted Qwen3.5-9B and Qwen3.6-27B, GPT-5.4 prompted as a user, the released UserLM-8B, PatientSimulator, and UserLM-9B. Following UserLM's diagnostic design (Naous et al., 2025), we use 1,000 held-out HealthChat cases to supply patient turns and intents for perplexity, first-turn diversity, intent decomposition, end F1, and naturalness. The role-adherence probe uses 1,000 CommonsenseQA validation questions (Talmor et al., 2019) followed by an uncertain assistant response; the intent-adherence probe uses 1,000 Natural Questions queries (Kwiatkowski et al., 2019) followed by an off-topic assistant suggestion. We use scripted assistant responses for these two adherence probes. We report first-turn diversity as the unique-unigram rate, score naturalness with Pangram EditLens[7], and use Qwen3.6-27B to judge intent adherence.

For online DCS, all simulators interact with the same GPT-5.4 assistant on the same 1,000 HealthBench-derived cases. Each interaction ends when the simulator terminates or reaches a limit of 15 patient turns; simulator and assistant outputs are capped at 96 and 2,048 new tokens per turn, respectively. For the released UserLM-8B, we follow the model-specific inference protocol described by Naous et al. (2025): we decode directly in the user role from the high-level intent and interaction history and treat the native `<|endconversation|>` token as termination after the first patient turn. For this baseline, at each turn, we make up to eight generation attempts, rejecting empty or malformed outputs, outputs outside 3–45 words, exact repetitions of earlier patient turns, and first-turn termination. If all attempts fail, we retain the last nonempty candidate, or stop the interaction if every attempt is empty.

Across these cases, we evaluate the same 2,616 disclosure items corresponding to content-bearing pop turns. For each item, a Qwen3.6-27B judge receives the high-level patient intent, the target item, its reference wording when available, and the complete sequence of simulator-generated patient utterances, but not the assistant responses. Using greedy decoding, the judge assigns an integer score from 0 (absent) to 5 (expressed completely and clearly). We average item scores within each case and then report the mean across the 1,000 cases.

[7] https://huggingface.co/pangram/editlens_Llama-3.2-3B

B.2 Interactive Benchmark Evaluation

HealthBench supplies a complete conversation preceding each evaluated response, whereas KnowMT-Bench prescribes patient turns while allowing the evaluated assistant to generate its own preceding responses. We convert 1,000 HealthBench cases and all 201 medical KnowMT-Bench cases to PatientLM interactions. A Qwen3.6-27B alignment extractor examines only the generated patient turns to identify each target request. If a HealthBench interaction is unaligned, we regenerate it up to five times and retain the first aligned trajectory. The assistant responses following aligned patient turns are scored using the source benchmark's rubric items for HealthBench or factuality and hallucination metrics for KnowMT-Bench. Unaligned cases receive no assistant score. For each assistant, original and interactive scores use the intersection of scored cases; alignment coverage is reported over the full benchmark set.

Protocol differences are tested with paired two-sided permutation tests using 10,000 random sign flips. We apply Holm correction at $\alpha = 0.05$ across assistant models for HealthBench and separately within each KnowMT-Bench metric.

B.3 Assistant Training

For all three training regimes, supervision is constructed from 1,000 HealthBench cases split by case identity into 950 training cases and 50 validation cases, so turns from the same case cannot appear in both splits. Teacher-generated offline supervision consists of complete dialogues produced in one pass. PatientLM–teacher offline supervision is collected from complete interactions before student optimization. In the online setting, student responses shape subsequent PatientLM turns, while the teacher supplies a target response at each student-induced dialogue state.

We fine-tune the full Qwen2.5-7B-Instruct model on four GPUs with AdamW for four epochs using a learning rate of 5×10^{-6}, a cosine schedule with a 3% warmup ratio, a global batch size of 32, per-GPU micro-batches of 1, and a maximum sequence length of 4,096. We use right truncation for teacher-generated SFT and left truncation for the two PatientLM-based regimes. For each regime, we select the epoch with the lowest validation loss. The selected checkpoints occur after epoch two: step 58 for teacher-generated SFT, step 174 for PatientLM–teacher offline SFT, and step 184 for student-in-the-loop online SFT; the different step counts reflect the different numbers of turn-level training examples. We assess the selected assistants using KnowMT-Bench's factual recall and hallucination rate.

B.4 Hyperparameter Sensitivity Analysis

The PatientLM hyperparameter comparison evaluates learning rates of $\{5 \times 10^{-6}, 10^{-5}, 2 \times 10^{-5}\}$ after one and two epochs (steps 246 and 492), using token-weighted perplexity on the same 161,564 held-out patient tokens.

B.5 Prompt and Inference Details

At each nonterminal turn, PatientLM receives the high-level patient intent h_i, the preceding interaction history, and the disclosure state $d_t^{(i)} = (c_j^{(i)}, a_t^{(i)}, \bar{s}_t^{(i)})$. The implementation serializes the scenario-level role and style, controller-selected context, stack of pending disclosure items, action, and released disclosure item. For a pending item, the controller prompt includes the patient profile, the last released item and its context when available, the next pending item and its context, and the preceding interaction; the valid actions are keep and pop. After the stack is exhausted, the terminal prompt replaces the pending-item context and restricts the actions to keep and $\mathrm{terminate}$. The controller returns one valid action, which is then passed to PatientLM for utterance generation.

In full interactions, we cap each trajectory at 15 patient turns, PatientLM outputs at 96 new tokens, controller outputs at 512 new tokens, and assistant outputs at 2,048 new tokens. PatientLM uses temperature 1.0 and top-p 0.8; the local prompted and UserLM baselines use temperature 0.7, top-p 0.8, and top-k 20. The controller uses greedy decoding.

Prompted baselines receive the corresponding case-level patient description and interaction history but no semantic stack or controller action. All simulator comparisons use the same assistant model,

Table 9: Reference-dialogue identification by Qwen3.6-27B on 1,000 cases per simulator. All values are percentages.

Simulator	Identification accuracy (%)	95% Wilson CI	Correct across swapped orders			Reference position	
			Both	One	Neither	A	B
UserLM-8B	92.0	[90.7, 93.1]	86.5	11.0	2.5	92.9	91.1
UserLM-9B	76.6	[74.6, 78.4]	63.2	26.7	10.1	87.4	65.7
PatientLM	53.8	[51.6, 56.0]	27.5	52.6	19.9	76.7	30.9

case set, turn limit, and evaluation prompts. The offline diagnostics follow the UserLM evaluation protocol and use the data and judge models described above.

B.6 Controller Evaluation

We evaluate the external Qwen3.6-27B controller by replaying all HealthChat dialogues used in our study. Before each patient turn, the controller receives the preceding reference dialogue and the corresponding disclosure state, and its predicted action is compared with the action derived from the source dialogue. This yields 33,527 decisions, comprising 21,426 pop, 1,547 keep, and 10,554 terminate labels.

For the k-nearest-neighbor diagnostic in Section 4.3, we set $k = 20$ and use all-MiniLM-L6-v2 embeddings. For each decision, we construct a label-free text representation by concatenating the specialty, turn and item-state indicators, taxonomy labels, runtime state, patient profile and interaction rules, released and pending item content, item-specific and terminal contexts, and the preceding interaction history. We normalize whitespace, truncate the representation to 6,000 characters, encode it with all-MiniLM-L6-v2, and use cosine similarity for nearest-neighbor retrieval.

C Additional Experiments

C.1 Dialogue-level Naturalness Evaluation

In addition to the utterance-level naturalness diagnostic, we conduct a complementary dialogue-level evaluation. HealthBench reference conversations were designed to approximate realistic medical interactions and were produced through synthetic generation and human adversarial testing, spanning multi-turn exchanges involving lay users and healthcare professionals (Arora et al., 2025). We therefore use them as naturalistic references.

Table 8: Cross-source PPL for PatientLM on the LMSYS- and WildChat subsets of HealthChat.

Train → Test	Base	PatientLM
LMSYS → WildChat	7.929	5.411
WildChat → LMSYS	6.703	4.585

For each of 1,000 HealthBench-derived cases, the reference dialogue retains the benchmark-provided user turns and intermediate assistant responses. Because HealthBench leaves the final user request unanswered for evaluation, we append a GPT-5.4 response to complete the dialogue. The corresponding simulated dialogue is generated from the same case through turn-by-turn interaction between the simulator and GPT-5.4. A Qwen3.6-27B judge is told that one dialogue contains messages from a real user and the other contains simulator-generated user messages, with the HealthBench reference used as the real-user target. The judge must identify the real-user dialogue from the user's messages and conversational behavior while ignoring assistant quality and medical correctness, returning either A or B. Each pair is evaluated twice with the dialogue order swapped, using greedy decoding. We report UserLM-8B, the matched UserLM-9B, and PatientLM.

Reference-identification accuracy is the fraction of judgments in which the judge selects the HealthBench reference as the real-user dialogue. Because each pair appears in both orders, the overall accuracy is position-balanced. An accuracy closer to 50% indicates that the judge finds the reference and simulated dialogues more difficult to distinguish and therefore more similar in user language and conversational behavior. We additionally report the percentage of cases in which the judge is correct in both orders, exactly one order, or neither order.

As shown in Table 9, PatientLM's reference-identification accuracy of 53.8% is closest to 50%, compared with 76.6% for UserLM-9B and 92.0% for UserLM-8B. The judge identifies the reference in both orders for 27.5% of PatientLM cases, compared with 63.2% for UserLM-9B and 86.5% for UserLM-8B. These results indicate that PatientLM produces the dialogues most similar to the naturalistic references among the evaluated simulators.

The order-specific results are nearly balanced for UserLM-8B, whereas UserLM-9B and PatientLM show higher accuracy when the reference appears in position A. We therefore use position-balanced overall accuracy as the primary comparison. Together with the higher disclosure coverage, the near-chance identification accuracy shows that PatientLM combines more complete case coverage with stronger dialogue-level naturalness.

C.2 Cross-Source Generalization

For the cross-source generalization experiment, we train Qwen3.5-9B on LMSYS and test on WildChat, then reverse the direction.

As shown in Table 8, PatientLM consistently lowers PPL relative to the base model in both directions, indicating that its patient-language modeling transfers across HealthChat sources rather than being limited to source-specific expressions.

D Qualitative Case Study

Table 10 expands the micronutrient-supplementation example in Figure 1. The original HealthBench trajectory discloses three disclosure items, yielding the action sequence $\mathrm{pop} \rightarrow \mathrm{pop} \rightarrow \mathrm{pop} \rightarrow \mathrm{terminate}$. We use this case to compare how different user simulators organize, disclose, and revisit the same disclosure items. The final column summarizes the assistant response that conditions the next patient turn. For the reference trajectory, the first two source responses are summarized, and the final evaluation target is summarized from the benchmark's ideal completion. Reference actions are corresponding actions mapped onto the source trajectory rather than controller outputs.

This case exposes a behavioral difference between high-level-intent-only simulation and controller-guided progressive disclosure. The reference trajectory introduces the three disclosure items sequentially and then ends. UserLM-8B instead front-loads generic information about stunting and repeatedly returns to the same supplementation request. The resulting trajectory reaches the turn limit, leaves the growth-effects item undisclosed, and only partially expresses the final action-plan request, yielding a DCS of 2.67. PatientLM releases the same three disclosure items in order under both assistants and reaches a DCS of 5.0, illustrating how the disclosure plan can preserve case coverage without requiring a fixed surface trajectory.

PatientLM also differs from a scripted replay after the planned items have been released. With GPT-5.4, the controller uses two keep turns to elicit still-needed operational detail about assessment and next steps. With Qwen3.6-27B, the assistant response already resolves those interaction needs, so the controller terminates immediately. Together, these trajectories illustrate how PatientLM can preserve the case-level disclosure objective while adapting the length and wording of patient behavior to different assistant responses.

E Prompt Templates

This section reports the principal prompts used to construct PatientLM inputs, control disclosure at inference time, and evaluate generated interactions. Placeholders enclosed in braces are replaced with case-specific content. We omit only mechanical formatting code and benchmark prompts that are used unchanged from their official implementations. In these prompts, "semantic node" refers to a "disclosure item" in the main text.

Table 10: Source and simulated trajectories for the micronutrient-supplementation case in Figure 1. Patient utterances are lightly edited, assistant responses are summarized, and repetitive UserLM-8B turns are grouped.

Trajectory	Turn	Action	Patient utterance	Assistant response summary
Reference	1	pop (corresponding)	"I suspect families in my area have micronutrient deficiencies. Does that cause growth problems?"	Confirms that deficiencies can impair growth and outlines common deficiencies and interventions.
	2	pop (corresponding)	"We have many kids with stunting or low weight. I'd like to start them on daily vitamins. Do we just give iron, or are other vitamins needed?"	Explains the roles of iron, vitamin A, and zinc and asks about local diets and supplementation programs.
	3	pop (corresponding)	"We don't have lab tests or consistent supply. I want to do something now—maybe iron plus vitamin A, or none."	*Evaluation target:* Recommends age-appropriate iron and vitamin A supplementation, safe dosing, dietary diversification, and local guidance.
	End	terminate (corresponding)	No further patient turn.	–
UserLM-8B + GPT-5.4	1	–	"The World Health Organization defines stunting as being too short for one's age. [...] Stunting is caused by [...]"	Recommends combined supplementation immediately.
	2–15	–	Repeats: "What would be the best strategy, giving iron alone or iron plus vitamin A?"	Similar responses recur as the patient repeats the same global request.
	End	–	Reached the 15-turn limit; item scores $[0, 5, 3]$ and case $\mathrm{DCS} = 2.667$.	–
PatientLM + GPT-5.4	1	pop	"Are there growth problems related to micronutrient deficiencies?"	Explains how iron, zinc, and vitamin deficiencies affect growth.
	2	pop	"A lot of kids have stunting and low weight. What should I give them—iron or other vitamins?"	Prioritizes nutrition and assessment over one supplement.
	3	pop	"I can't check labs and I can't supply. What should I do—iron plus vitamin A?"	Warns against blanket dosing and gives practical next steps.
	4	keep	"What if I can check age and weight?"	Explains what age and weight can and cannot determine.
	5	keep	"Okay, give me a simple algorithm."	Provides a field algorithm with danger signs and referral rules.
	End	terminate	Controller termination; item scores $[5, 5, 5]$ and case $\mathrm{DCS} = 5.000$.	–
PatientLM + Qwen3.6-27B	1	pop	"Do micronutrients cause growth problems?"	Explains that deficiencies can impair growth.
	2	pop	"Many kids are small or have low weight. Should we give iron or other vitamins?"	Advises evaluation rather than automatic supplementation.
	3	pop	"If I cannot afford blood tests and cannot supply medicine, what should I do about iron plus vitamin A?"	Gives a direct safety warning and low-cost alternatives.
	End	terminate	No further patient turn.	The response supplies the assessment criteria and next steps needed to close the case.

E.1 STRUCTURED-INTENT CONSTRUCTION

Patient-turn annotation and semantic-node extraction

```
Classify each USER turn in a medical conversation for PatientLM construction.

PatientLM uses semantic nodes for user turns that introduce health information or a
     health-related request. It preserves interaction-management turns as context, but
     those turns should not become semantic nodes.

Labels:
- A: The user provides stable health-related context only, such as symptoms, history,
     laboratory results, medications, demographics, constraints, or preferences. No
     clear request is made.
- B: The user makes a health-related request only, such as explanation, advice, triage,
      treatment options, planning, drafting, or interpretation. No important new user-
     specific health context is added.
- AB: The user both provides stable health-related context and makes a health-related
     request in the same turn.
- C: Interaction management only. The user clarifies, corrects, pushes back, expresses
     confusion or frustration, states an action plan, gives resolution, or otherwise
     manages the previous answer without introducing a distinct new health goal or
     stable clinical fact.
- D: Out of scope. Use for off-topic content, uninterpretable text, phatic greetings or
      thanks only, or role-setting with no health context or request.

Annotation rules:
1. Annotate every user turn exactly once.
2. For A, B, or AB, write a concise semantic_content sentence describing what the
     simulated patient is about to reveal or ask.
3. For C or D, set semantic_content to an empty string.
4. Prefer AB when a turn combines personal health context with a concrete request.
5. Prefer B for a follow-up question seeking new medical guidance, even if it refers to
      previous context.
6. Use C only when the turn is mainly a repair or management move about the assistant's
      prior answer.
7. Do not evaluate whether the assistant answer is medically correct.

Conversation:
{CONVERSATION}

Return JSON only:
{
  "annotations": [
    {
      "turn": 1,
      "type": "A|B|AB|C|D",
      "semantic_content": "node sentence for A/B/AB, otherwise empty",
      "rationale": "brief reason"
    }
  ]
}
```

High-level patient profile construction

```
You are creating structured intermediate data for a patient simulation dataset.

Conversation metadata:
- Specialty: {SPECIALTY}

Full conversation:
{CONVERSATION_WITH_TAXONOMY_LABELS}

Goal:
Summarize the conversation using only stable, high-level information. The summary will
     be used to write a patient system prompt, so it must not reveal concrete facts
     that should be released later as semantic nodes.

Instructions:
1. Infer a high-level patient identity, such as a patient, family member, caregiver, or
      other non-clinician.
2. Write one sentence describing the medical domain in abstract terms.
3. Infer one primary goal and any meaningfully distinct secondary goals.
```

```
4. Describe the patient's communication tone and emotional register in one or two
     sentences.
5. Write two to four operational interaction rules from interaction-management signals.
      Each rule must name an observable trigger, the patient's response, and the
     condition under which that response stops.
6. Do not write standalone rules that simply move to the next topic after an answer.
     Progression is handled by the controller.
7. Do not invent demanding, skeptical, or emotionally intense behavior unless supported
      by the conversation.
8. Ignore out-of-scope content.

Return JSON only:
{
  "summary": {
    "identity": "high-level patient identity",
    "specialty_context": "abstract medical domain",
    "primary_goal": "main goal",
    "secondary_goals": ["secondary goal"],
    "style_summary": "communication style",
    "interaction_rules": [
      "If {OBSERVABLE_TRIGGER}, then {PATIENT_ACTION}; stop only after {STOP_CONDITION
     }."
    ]
  }
}
```

Patient system-prompt construction

```
You are writing a patient system prompt for PatientLM from a structured summary.

PatientLM is a patient simulator. The system prompt contains only the patient's
     identity, high-level goals, communication tone, and interaction habits. Concrete
     facts and questions are stored separately as semantic nodes, and disclosure rules
     are stored separately as node-controller contexts.

Structured summary:
{STRUCTURED_SUMMARY}

Task:
Write the system prompt as two blocks.

Block 1 - Character paragraph:
- Use prose in the second person.
- Describe who the patient is, what they want to understand, and their communication
     tone.
- Do not reveal facts that belong to semantic nodes.
- Do not describe keep, pop, terminate, or movement through the node stack.

Block 2 - Interaction rules:
- Add the heading "Interaction rules:".
- Copy the supplied interaction rules verbatim as bullets.
- Do not add transition, disclosure, or termination rules.

Return JSON only:
{"system_prompt": "{CHARACTER_PARAGRAPH}\n\nInteraction rules:\n- {RULE_1}\n- {RULE_2
     }"}
```

E.2 Controller-Context Construction

The first prompt constructs an item-specific context c_j for each disclosure item, while the second constructs one terminal context c_{term} for each case. The resulting contexts are inserted into the runtime controller prompts below.

Disclosure-item controller-context construction (c_j)

```
You are writing a per-node controller instruction for PatientLM.

At inference time the controller sees a <node_context> block before deciding keep / pop
      / terminate. Write the node_context text for the node below. It must have exactly
      two labeled sections.

Patient profile:
- identity: {PATIENT_IDENTITY}
- style: {COMMUNICATION_STYLE}
- system_prompt (excerpt):
{PATIENT_SYSTEM_PROMPT}

Interaction rules:
{INTERACTION_RULES}

Current node:
- position: {NODE_POSITION}
- type: {A|B|AB}
- semantic_content: {DISCLOSURE_ITEM}

Node type guidance:
{NODE_TYPE_GUIDANCE}

Sequence context:
{REMAINING_SEQUENCE_CONTEXT}

Task:
Write the node_context in two labeled sections.

Section 1 - Situation (2--3 sentences):
- Describe what information or intent the patient is holding in mind.
- Explain how the interaction style shapes how the patient approaches this node.
- State where the node lies in the conversation arc.
- Do not describe future nodes or repeat the generic action definitions.

Section 2 - Decision rules:
- keep: write two observable conditions: one for approaching or building context
     without completing the node, and one for an interaction-management move such as
     clarification, pushback, confusion, frustration, or emotional reaction.
- pop: write one observable condition tied directly to the node's semantic content.
- terminate: if later nodes remain, state that terminate is not applicable. For the
     final node, describe a natural-ending condition after it is released while
     preserving concrete same-topic follow-ups.

Requirements:
- Every condition must name concrete patient behavior.
- If a clarification, correction, pushback, confusion, frustration, or emotional-repair
      turn does not also complete the current node, keep takes precedence.
- Do not choose keep merely because the assistant answer could be more detailed.
- Keep the rules consistent with the sequence and patient profile.
- Label the sections exactly "Situation:" and "Decision rules:".
- Total length: 130--160 words.

Return JSON only:
{"controller_prompt": "Situation:\n{SITUATION}\n\nDecision rules:\n- keep: if {...} ->
     keep; if {...} -> keep\n- pop: if {...} -> pop\n- terminate: if {...} -> terminate
      / terminate not applicable here"}
```

Terminal controller-context construction (c_{term})

```
You are writing a terminal controller instruction for PatientLM.

At inference time this context is used only after every semantic node has been released
      and the stack is empty. The controller must decide whether the patient naturally
     ends the conversation or produces one more same-topic interaction-management turn.

Patient profile:
- identity: {PATIENT_IDENTITY}
- style: {COMMUNICATION_STYLE}
- system_prompt (excerpt):
{PATIENT_SYSTEM_PROMPT}
```

```
Interaction rules:
{INTERACTION_RULES}

Released conversation arc:
{LAST_DISCLOSURE_ITEMS_IN_ORDER}

Final released node:
- type: {A|B|AB}
- semantic_content: {FINAL_DISCLOSURE_ITEM}

Task:
Write terminal_controller_prompt in two labeled sections.

Section 1 - Situation (2--3 sentences):
Describe the patient's state after all planned semantic units have been released.
     Explain whether this profile tends to end after a sufficiently addressed final
     goal or continue with a concrete same-topic management turn.

Section 2 - Decision rules:
- keep: write one condition for a concrete gap that this patient would challenge,
     clarify, or correct, and one for continuing distress, confusion, or explicit
     nonacceptance.
- terminate: write one condition for when the final goal is sufficiently addressed or
     no strong continuation trigger fires.
- pop: state that pop is not applicable because the stack is empty.

Requirements:
- Do not choose keep merely because the final answer could be better.
- Allow a natural ending without explicit thanks or a statement that there are no
     further questions.
- Make terminate the default after the final answer unless a strong continuation
     trigger is visible.
- Keep this context consistent with the final node context.
- Label the sections exactly "Situation:" and "Decision rules:".
- Total length: 110--150 words.

Return JSON only:
{"terminal_controller_prompt": "Situation:\n{SITUATION}\n\nDecision rules:\n- keep: if
     {...} -> keep; if {...} -> keep\n- terminate: if {...} -> terminate\n- pop: not
     applicable because the stack is empty"}
```

E.3 PATIENTLM INFERENCE

Patient-language-model input serialization

```
<|im_start|>system
# Task
You are a patient simulator.
For each turn, an external controller provides runtime state blocks: <node_context>, <
     stack>, <action>, and <info>.
Use this state and dialogue history to produce only the patient's next utterance.
If action is terminate, output no patient utterance for that turn.

---

# Role play
{PATIENT_PROFILE}
<|im_end|>

For each patient turn:
<node_context>
{CURRENT_NODE_CONTEXT}
</node_context>
<stack>
{REMAINING_SEMANTIC_NODES}
</stack>
<action>
{keep|pop|terminate}
</action>
<info>
{RELEASED_NODE_CONTENT_IF_ACTION_IS_POP}
</info>
<think>
```

```
</think>

<|im_start|>user
{PATIENT_UTTERANCE}
<|im_end|>
```

Pending-node controller prompt

```
System:
You are an expert controller for a patient simulator. Output only a valid JSON object
     and nothing else.

User:
You are the controller of a patient simulator.

Task:
Choose whether to release the next pending semantic unit before the next patient turn.

Actions:
- keep: do not release a new semantic unit. The patient would naturally produce one
     more interaction-management turn first, such as clarification, confusion, pushback
     , correction, anxiety, an action-plan statement, or another same-topic repair.
- pop: release the next pending A/B/AB semantic unit now, so the patient's next content
      turn uses it.

Runtime state:
A pending semantic unit exists. Available actions are keep and pop. Terminate is
     unavailable.

Patient simulator profile:
{PATIENT_PROFILE}

Released semantic units, oldest to most recent:
{RELEASED_NODES}

Last released semantic unit:
{LAST_RELEASED_NODE}

Last released node context:
{LAST_NODE_CONTEXT}

Next pending semantic unit:
{NEXT_PENDING_NODE}

Next pending node context:
{NEXT_NODE_CONTEXT}

Conversation history before this patient turn:
{DIALOGUE_HISTORY}

Decision policy:
1. Choose only from the available actions.
2. Keep is a narrow interaction-management action. Choose keep only when the patient
     must produce a concrete extra same-topic turn distinct from the next pending
     semantic unit.
3. If the next pending semantic unit already represents the likely follow-up,
     correction, concern, refusal, pushback, clarification, anxiety, or request for
     details, choose pop.
4. Do not choose keep merely because the assistant answer is imperfect, incomplete,
     incorrect, unsafe, off-topic, unempathic, or broad. Such problems justify keep
     only when the patient profile or node context predicts a separate management turn
     and the pending unit does not already cover it.
5. Treat the patient profile as style and tendency guidance. Do not let it block a
     matching next-node context unless it explicitly requires a separate management
     turn.
6. If the evidence is balanced or uncertain, choose pop.

Return JSON only:
{
  "last_llm_quality": "complete|ambiguous|too_broad|inadequate",
  "current_topic_unresolved": true,
  "matched_rule_source": "profile|last_node_context|next_node_context|none",
  "reason": "at most two short sentences",
  "action": "keep|pop"
}
```

Terminal controller prompt

```
System:
You are an expert controller for a patient simulator. Output only a valid JSON object
     and nothing else.

User:
You are the controller of a patient simulator.

Task:
Choose whether the patient would continue with one more same-topic interaction-
     management turn or naturally end the conversation.

Actions:
- keep: the patient would naturally produce one more same-topic interaction-management
     turn, such as clarification, confusion, pushback, correction, anxiety, an action-
     plan statement, or another same-topic repair.
- terminate: end the conversation because no pending semantic unit remains and the last
      released goal appears sufficiently addressed.

Runtime state:
No pending semantic unit remains. Pop is structurally impossible. Available actions are
      keep and terminate.

Patient simulator profile:
{PATIENT_PROFILE}

Released semantic units, oldest to most recent:
{RELEASED_NODES}

Last released semantic unit:
{LAST_RELEASED_NODE}

Last released node context:
{LAST_NODE_CONTEXT}

Terminal controller context:
{TERMINAL_CONTEXT}

Conversation history before this patient turn:
{DIALOGUE_HISTORY}

Decision policy:
1. Choose only from the available actions.
2. Do not terminate only because the stack is empty; the patient may still produce a
     same-topic management turn.
3. Terminate is the strong prior after the final goal receives a substantive answer
     attempt. Choose keep only when the terminal context, patient profile, or recent
     history predicts a concrete continuation.
4. Do not independently fact-check the assistant or compare it with an ideal answer.
5. For a direct or low-friction patient, terminate after a substantive answer attempt
     even if the answer has minor errors, omissions, or vague wording.
6. Choose keep when the answer refuses or does not attempt the request, is visibly
     truncated, asks for clarification instead of answering, or violates a concrete
     constraint that the profile says the patient will enforce.
7. If the evidence is balanced or uncertain, choose terminate.

Return JSON only:
{
  "last_llm_quality": "complete|ambiguous|too_broad|inadequate",
  "current_topic_unresolved": true,
  "matched_rule_source": "profile|last_node_context|terminal_context|none",
  "reason": "at most two short sentences",
  "action": "keep|terminate"
}
```

E.4 Evaluation Prompts

Disclosure-item coverage judge

```
Evaluate how completely the user simulator surfaced one semantic node in the
     conversation.

A semantic node is a specific subgoal, background fact, or question that the user
     simulator is expected to express or pursue. Judge only the user messages.

User goal or persona:
{INTENT}

Semantic node to evaluate:
{SEMANTIC_NODE}

Reference user wording, if available:
{REFERENCE_USER_TEXT}

User messages from the simulated conversation:
{USER_MESSAGES}

Return strict JSON only with keys: score, rationale.
Keep rationale under 80 words and close the JSON object immediately after the rationale
     .
score must be an integer from 0 to 5:
0 = the node is absent, or there are no valid user messages
1 = only a vague hint related to the node is present
2 = part of the relevant fact or question is expressed, but the central need remains
     unclear
3 = the central fact or question is identifiable, but important details are missing
4 = the node is expressed nearly completely, with only minor details omitted
5 = the node is expressed completely and clearly enough to recover the intended user
     need

Use semantic equivalence rather than exact wording. Evaluate only what the user
     communicated.
```

Target-alignment extractor

```
System:
You are a strict alignment extractor. Decide only whether the simulated PATIENT turns
     contain the same substantive question or request as the original target request.
     Do not judge assistant quality. Do not use assistant answers to infer patient
     intent.

User:
Original target request:
{TARGET_REQUEST}

Simulated conversation PATIENT turns:
{PATIENT_TURNS}

Task:
Decide whether the simulated patient turns contain the target request.

Labels:
- contains_target_request: the patient asks the same core question or request, even if
     it is paraphrased or split across turns. Minor missing details are acceptable if
     the assistant can still answer the target request.
- not_contains_target_request: the patient never asks the core request or shifts to a
     materially different task.

Rules:
- Base the decision only on simulated patient turns.
- Do not infer intent from assistant responses.
- If the topic is related but the task changes, choose not_contains_target_request.
- If unsure whether enough of the request was expressed, choose
     not_contains_target_request.

Return strict JSON only:
{
  "containment": "contains_target_request|not_contains_target_request",
```

```
  "relevant_patient_turn_indices": [0],
  "confidence": "high|medium|low",
  "reason": "short reason"
}
```

Intent-adherence judge

```
System:
You are judging a patient-simulator intent-adherence test.

The patient first asks an original question. The assistant then does not answer and
     suggests an unrelated topic. Judge only the simulator's next patient reply.

Label the reply adherent if it continues trying to get the original question answered,
     asks the assistant to return to it, clarifies it, or rejects the unrelated topic.

Label the reply non_adherent if it accepts the unrelated topic, abandons the original
     question, answers as an assistant, or is empty or irrelevant.

Return JSON only:
{"label": "adherent|non_adherent", "reason": "short reason"}

User:
Original question:
{ORIGINAL_QUESTION}

Assistant drift response:
{ASSISTANT_RESPONSE}

Simulator's next patient reply:
{PATIENT_REPLY}

Judge whether the simulator maintained the original intent.
```

High-level-intent prompt for prompted and UserLM-style baselines

```
System:
You create high-level user intents for user-simulation benchmarks. Summarize the user's
      overall objective in a form that can condition a user simulator. Return only the
     intent sentence.

User:
Given the conversation between a user and an assistant, summarize the user's high-level
      intent while preserving the goals, constraints, and context needed to simulate
     the conversation. Do not produce a turn-by-turn script, and do not copy wording
     unless necessary for meaning.

Begin exactly with:
"You are a user chatting with an assistant language model to"

Conversation:
{SOURCE_CONVERSATION}

Return only the intent summary.

At simulation time, append:
"If the user's goal has already been satisfied, output exactly <END_CONVERSATION>.
     Otherwise, write only the next natural user message."
```