SAGE: A Top-Down Bottom-Up Knowledge-Grounded User Simulator for Multi-turn AGent Evaluation

Anonymous ACL submission

Abstract

Evaluating multi-turn interactive agents is challenging due to the need for human assessment. Evaluation with simulated users has been introduced as an alternative, however existing approaches typically model generic users and overlook the domain-specific principles required to capture realistic behavior. We propose *SAGE*, a novel user **S**imulation framework for multi-turn **AG**ent **E**valuation that integrates knowledge from business contexts. *SAGE* incorporates top-down knowledge rooted in business logic, such as ideal customer profiles, grounding user behavior in realistic customer personas. We further integrate bottom-up knowledge taken from business agent infrastructure (e.g., product catalogs, FAQs, and knowledge bases), allowing the simulator to generate interactions that reflect users' information needs and expectations in a company's target market. Through empirical evaluation, we find that this approach produces interactions that are more realistic and diverse, while also identifying up to 33% more agent errors, highlighting its effectiveness as an evaluation tool to support bug-finding and iterative agent improvement.

1 Introduction

Interactive agents are increasingly deployed in high-stakes domains such as customer support, sales, and enterprise productivity (Muthusamy et al., 2023). As these systems become more sophisticated, evaluation has become a major bottleneck in their development. Traditional approaches rely heavily on human users, which is both costly and difficult to scale across iterative development cycles (Davidson et al., 2023). Automatic evaluation methods promise greater efficiency, but existing approaches are limited. In particular, most rely on static benchmarks or narrow task-specific datasets, that fail to capture the open-ended and dynamic nature of real-world interactions (Yao et al., 2024).

Automated evaluation using user simulators has therefore emerged as a promising direction, enabling researchers and practitioners to benchmark multi-turn agents in diverse scenarios with reduced human effort. Yet most existing simulators depend heavily on human interaction data, which is often scarce, proprietary, or expensive to obtain (Dhole, 2024; Shi et al., 2019). More recent work has explored prompting-based approaches to mitigate this dependence, but these methods typically portray users as domain-general interlocutors, with goals and behaviors only loosely tied to the target application (Barres et al., 2025; Davidson et al., 2023). In practice, however, many real-world agents are deployed in specialized business settings where user behavior is deeply shaped by who the user is (e.g., their role, demographics, or purchasing potential) and what information they seek (e.g., company-specific product knowledge). These dimensions are easily overlooked by simulators that fail to integrate business design principles, resulting in interactions that are less realistic and with limited utility for rigorous evaluation.

To address these limitations, we introduce *SAGE*, a knowledge-grounded user simulation framework for multi-turn agent evaluation. *SAGE* draws on top-down principles grounded in business logic to construct realistic user personas, incorporating attributes from ideal customer profiles (ICPs) alongside other factors that shape customer behavior. In addition, it integrates bottom-up knowledge taken from business agent infrastructure. This includes structured and unstructured knowledge such as product catalogs, FAQs, and knowledge bases, allowing the simulator to generate interactions that accurately reflect the knowledge and needs of relevant users of the company. By combining top-down and bottom-up knowledge, *SAGE* produces more realistic and contextually relevant interactions, enabling scalable evaluation without requiring costly human efforts.

Through empirical evaluation, we demonstrate that *SAGE* generates interactions that are more human-like and more diverse than existing simulation methods. Moreover, it uncovers a greater number of agent errors, underscoring its effectiveness as a tool for bug discovery and iterative agent improvement. Our contributions are as follows:

- We introduce *SAGE*, a novel user simulation framework for multi-turn agent evaluation. *SAGE* integrates top-down principles grounded in business logic and bottom-up knowledge from business agent infrastructure to generate realistic user interactions.
- We conduct empirical evaluations illustrating that *SAGE* produces interactions that are more human-like, diverse, and contextually grounded compared to existing methods.
- We demonstrate *SAGE*'s utility for identifying agent errors, finding that it identifies 25-33% more bugs than existing methods.

2 Related Work

Prior work on multi-turn agent evaluation has largely focused on creating static or task-specific benchmarks. For example, MTRAG (Katsis et al., 2025) introduced a human-written multi-turn RAG benchmark that evaluates end-to-end performance across four domains. τ-bench (Yao et al., 2024) evaluates API-augmented agents in the retail and airline domains using dynamic task-oriented dialogues between a simulated user and the target agent. CRMArena (Huang et al., 2025) starts with human-written seed query templates for initial user requests and then rolls out interactions using prompt-based rules. While these methods are able to effectively assess agents in certain scenarios, their evaluations are limited to a fixed set of tasks. *SAGE* provides a flexible simulation framework that can adapt to specialized business settings, allowing it to better capture the evaluation needs of real-world agent deployment.

Previous work on user simulation methods often make heavy use of human data to generate realistic interactions. Sun et al. (2025) conducts a human study to collect demographic information, shopping preferences, and personality traits to build digital mapping twins. PlatoLM (Kong et al., 2024), Parrot (Sun et al., 2024) and USP (Wang et al., 2025) learn human discourse patterns directly from human-LLM interactions via supervised learning. DAUS (Sekulić et al., 2024) fine-tunes an LLM on domain-specific conversation data to reduce hallucinations and maintain coherent dialogues for task-oriented dialogue. SimUSER (Bougie and Watanabe, 2025) extracts self-consistent personas from historical user data for use in in-context learning. Although these methods can produce effective simulations, their heavily reliance on human data often limits their usability for real-world use cases where this kind of data is difficult to obtain.

More recent work has explored prompt-based user simulators due to their cost efficiency and flexibility in domain adaptation. RecUserSim (Chen et al., 2025) introduces an LLM-based user simulator for evaluating conversational recommender systems. DuetSim (Luo et al., 2024) develops a simulator with two LLMs in tandem to ensure that the dialogue aligns with complex constraints. However, these approaches construct prompts using only generic user information without integrating the specialized knowledge needed to generate realistic interactions. As a result, they produce coherent conversations but often model generic users that are unable to capture the nuances of real-world agent interactions. Our work addresses this limitation by integrating both top-down and bottom-up knowledge, enabling simulations that are more realistic and useful for agent evaluation.

3 The *SAGE* framework

We introduce *SAGE* as a knowledge-grounded simulation framework designed for scalable multi-turn interactive agent evaluation. The process begins by configuring top-down and bottom-up knowledge including user attributes, relevant agent capabilities, and business agent resources (Section 3.1). Next, we construct task scenarios by selecting and combining these knowledge elements (Section 3.2). These scenarios serve as inputs to an LLM, which generates realistic multi-turn interactions with the agent being evaluated (Section 3.3). Finally, the generated interactions are analyzed to surface agent errors and failure cases (Section 3.4). An overview of the full process is provided in Figure 1.

3.1 Configure Top-down Bottom-up Knowledge

The first step in *SAGE* involves configuring the knowledge base that underpins the simulated users. *SAGE* integrates three complementary knowledge components. The first component, **user attributes**,

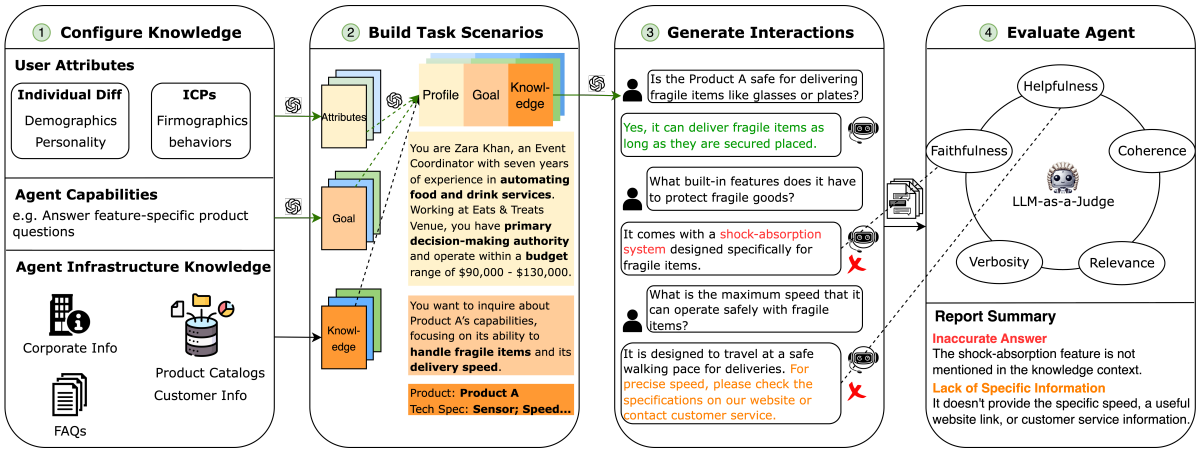

Figure 1: An overview of *SAGE*, which consists of four processes: **configure knowledge**, **build task scenarios**, **generate interactions**, and **evaluate agent**. The knowledge configuration step defines the top-down (user attributes) and bottom-up (agent infrastructure knowledge) knowledge. This knowledge is leveraged to construct task scenarios which guide the user simulator in its interactions with the agent being evaluated. The resulting interactions are analyzed to identify errors in the agent's response. The complete user profile is shown in Appendix D and the interaction examples are shown in Table 4.

defines the behavioral and personality traits that characterize each simulated user. We provide a predefined set of attributes that can be customized to align with a company's specific requirements. The second component, **agent capabilities**, represents the set of functionalities the business aims to evaluate, supplied directly by the company. The third component, **agent infrastructure knowledge**, is provided by the company and given to the simulator to align its informational context with the agent being evaluated (Figure 1).

User Attributes The user attributes we provide are derived from top-down principles grounded in business logic. These attributes fall into two complementary categories: individual differences (Williamson, 2018), which include psychological and demographic factors; and customer-specific attributes, captured by ideal customer profiles. These attributes are entirely configurable and can be changed or modified to fit the company's needs.

Individual Differences include the Big Five Personality framework (John and Srivastava, 1999) for modeling personality traits, and demographic categories following Toubia et al., 2025. These have been shown to be effective in predicting customer's behavior and have been used to construct simulated consumers in prior work (Khodabandehlou et al., 2020; Weinstein, 1994).

Ideal Customer Profiles characterize the type of customer most valuable to a business. They include attributes such as firmographics, decision-making authority, and budget. Prior marketing studies (Kivistö, 2024; Borgh et al., 2020) show that these components are more predictive of customer behaviors. Marketing teams often use ICPs to formulate sales strategies and prepare for customer interactions. Incorporating ICPs into our framework enables the simulator to reflect business-relevant user characteristics and generate more realistic interactions. See Appendix B for more details.

Agent Capabilities The agent capabilities are provided by the company and represent the set of functionalities that the business seeks to evaluate (Figure 1). To enable the simulator to assess these capabilities, *SAGE* translates each capability into a corresponding user goal using an LLM. User goals define the objective of an interaction, for instance, a user may aim to purchase a cleaning robot or inquire about the cleaning efficiency of a particular model. Creating the goals in this way anchors the objective of the user simulator while providing a systematic way to identify errors across the agent's full range of supported tasks.

Agent Infrastructure Knowledge We also provide the simulator with bottom-up knowledge derived from agent infrastructure. The information spans both unstructured resources (e.g., webpages, FAQs) and structured data (e.g., product catalogs, inventory tables, customer records). This knowledge is provided to align the information needs and expectations of the simulated users with actual business customers. For example, a real customer will

likely have read some of the company's webpages before interacting with the agent. Giving the user simulator access to the same knowledge helps align its informational context with actual users, allowing for more realistic and useful agent evaluations. This knowledge is also useful for later evaluation stages, allowing *SAGE* to directly compare the information in agent responses to the ground truth infrastructure knowledge.

3.2 Build Task Scenarios

Each task scenario is a natural language description composed of three components: a user goal, relevant agent infrastructure knowledge, and a user profile. We begin by choosing a user goal, then select the infrastructure knowledge most aligned with the goal. Then, we create a user profile that logically aligns with the chosen infrastructure knowledge. The final task scenario is formed by combining the components into a coherent description.

Goal Selection To choose a user goal, we loop through the full set of goals derived in Section 3.1. Since the user goals are generated based on agent capabilities, looping through them provides full coverage of the tasks we want to test the agent on.

Agent Infrastructure Knowledge Selection Given the user goal, we adopt a hierarchical selection process to ensure that the task scenarios include diverse yet relevant infrastructure knowledge. The agent infrastructure knowledge is organized into categories (e.g., FAQs, product catalogs), each containing multiple knowledge pieces such as structured product catalogs in SQL databases or unstructured text stored in vector databases. Given a user goal (e.g., inquire about discounts or promotional offers), we first identify relevant knowledge categories by prompting an LLM. Within each selected category (e.g., FAQ), we use a clustering algorithm to find the most relevant cluster of knowledge pieces (e.g., those related to reward programs or promo codes). To encourage diversity, we sample a knowledge piece from the identified cluster that has not been previously used; if all have been used, one is randomly selected from the cluster.

Profile Construction The user profile is generated based on the agent infrastructure knowledge and user attributes. Each user attribute consists of a key and a tuple of predefined values (e.g., purchase preference: discount oriented, economic, quality oriented). Some keys don't have predefined values since they may heavily depend on the specific use case or have too many values to enumerate (e.g., firmographics, budget). Therefore, the values are generated based on the selected infrastructure knowledge. We choose a final set of values jointly for all keys to preserve coherence among interdependent fields (e.g., job information depends on industry and company name). The final key-value pairs are used to construct the user profile, which is a natural language description of the user generated by an LLM (Figure 1). The related prompts are in Appendix G.1.

Task Scenario Construction The final task scenario is created using a structured template that combines the user goal, agent infrastructure knowledge, and user profile. The template also incorporates additional rules designed to help the model adhere more closely to the scenario and produce more human-like behavior. The constructed scenario is used as a system prompt to guide the user simulator during its interactions with the agent (see Appendix G.1 for details).

3.3 Generate Interactions

The user simulator and the agent both operate within a shared environment to ensure coherent and contextually aligned interactions. This environment integrates the interaction history, agent infrastructure knowledge, and other general world knowledge that depends on the specific use case. During interaction, both entities reference and update this environment dynamically. The user simulator is explicitly grounded on the agent infrastructure knowledge through its system prompt, whereas the agent accesses it through its internal framework, treated as a black box. The interaction history is continuously appended to both sides' context as the interaction progresses. The interaction terminates when the user emits a sentinel token (###STOP###) or the turn budget is reached.

3.4 Evaluate Agent and Identify Bugs

We conduct evaluations by identifying the errors that the agent makes during the generated interactions. The process begins by highlighting problematic turns using an LLM-as-a-Judge approach. Following prior work on agent evaluation (Zhang et al., 2025; Wang et al., 2024), each reply is judged along five dimensions: **helpfulness**, **coherence**, **verbosity**, **relevance**, and **faithfulness**. Each dimension is given a score from 0-4 accompanied

by a justification of the chosen score. *SAGE*'s bottom-up knowledge integration helps enhance this scoring by offering a concrete reference point for the LLM judge to assess agent responses, rather than relying on the model's internal knowledge.

In order to isolate agent errors, we only keep turns with a score below two. We validated this threshold through human expert verification, sampling 50 low-scoring turns (turns with scores below two) and finding that 94% contained bugs. In contrast, among 50 high-scoring turns, only four exhibited errors.

To give developers more concrete agent errors and actionable insights for improvement, we use an LLM to summarize the justifications from low-scoring turns into a list of unique bugs. This is done using a two-stage process. In the first stage, we prompt an LLM to generate high-level failure categories (e.g. unspecific response, inaccurate response, etc.) using the problematic turns and their score justifications. In the second stage, the output from the first stage along with the justifications are given to an LLM to generate fine-grained, unique bug statements (e.g., "did not provide weight capacity for Product A when explicitly requested"). Finally, we generate a report consisting of these bug statements along with the turn linked with each bug (see Figure 1 for an example).

We validated the quality of the LLM-identified bugs by comparing them to bugs identified by a human expert. Across the same set of 50 low-scoring turns, the list of bugs generated by the LLM had a 74% overlap with the list of bugs annotated by a human. Additional implementation details are provided in Appendix A.1, and prompt details are given in Appendix G.3 and G.4.

4 Experiments

We conduct several experiments to evaluate the effectiveness of *SAGE*'s top-down bottom-up knowledge incorporation. These experiments are designed to test the following two hypotheses:

H1: Incorporating top-down knowledge such as ICPs improves the user simulator's performance.

H2: Providing bottom-up knowledge from agent infrastructure improves simulator performance.

4.1 Evaluation Metrics

We evaluate the performance of our user simulator using the following metrics: lexical diversity, simulator fidelity, and number of agent errors identified.

Lexical Diversity To assess diversity, we utilize reference-free lexical metrics following prior work (Dhole, 2024). These include Measure of Textual Lexical Diversity (MTLD), logTTR, rootTTR (Dhole, 2024; McCarthy and Jarvis, 2010), Hypergeometric Distribution D (HD-D) (McCarthy and Jarvis, 2010), vocab size (Shi et al., 2019), and distinct n-gram counts (for n=1,2,3) (Li et al., 2016).

Simulator Fidelity We assess fidelity through human evaluation. We engaged two trained annotators who were tasked with rating the simulated user turns in interactions on a 3-point Likert scale across four dimensions:

- **Human-likeness**: How closely the simulated user's language resembles that of a human.
- **Coherence**: How well the simulated user's utterances follow logically from the dialogue history.
- **Specificity**: Whether the simulated user provides enough relevant and precise information for the system to proceed with the task.
- **Consistency**: Whether the simulated user remains consistent in facts, preferences, constraints, and style across turns.

These dimensions follow prior work on assessing user simulator fidelity (Luo et al., 2024). Additional details are shown in Appendix E.

Error Identification We also assess the simulators' ability to identify agent errors. This is done by following the agent evaluation process in Section 3.4 and counting the number of unique bugs that were identified. A higher count of unique bugs indicates broader ability to evaluate the the agent's capabilities and failure cases.

4.2 Experimental Setup

Settings We conduct our experiments by comparing *SAGE* against two different conditions:

w/o ICP: In this condition, the user simulator is given all knowledge except for top-down knowledge with respect to ideal customer profiles. Instead it has to rely exclusively on individual differences (Section 3.1) and bottom-up knowledge to generate realistic interactions. This approach follows prior methods of constructing prompt based user simulators such as the ones in τ-bench (Yao et al., 2024) and CRMArena (Huang et al., 2025).

Method	MTLD ↑	HDD ↑	logTTR ↑	rootTTR ↑	Vocab Size ↑	Distinct@1/2/3 ↑
RAG-based Sales Agent						
w/o ICP	81.43	**0.86**	0.77	11.58	1011	0.21/0.60/0.81
w/o Agent-Infra	80.76	0.85	0.76	10.65	973	0.18/0.55/0.76
SAGE	**86.05**	**0.86**	**0.78**	**11.80**	**1034**	**0.21/0.61/0.82**
Tool-Augmented Shopping Agent						
w/o ICP	76.52	**0.86**	0.75	9.88	907	0.18/0.57/0.78
w/o Agent-Infra	**81.59**	**0.86**	0.75	9.82	953	0.17/0.56/0.78
SAGE	79.97	**0.86**	**0.76**	**10.37**	**961**	**0.19/0.59/0.80**

Table 1: Lexical diversity results of *w/o ICP*, *w/o Agent-Infra* and *SAGE* for the RAG-based Sales Agent and Tool-augmented Shopping Agent cases using `gpt-4.1`. The best score for each metric is shown in **bold**.

Method	Human-likeness ↑	Coherence ↑	Specificity ↑	Consistency ↑
w/o ICP	2.38 (0.70)	2.75(0.49)	**2.80 (0.41)**†	**2.88 (0.33)**
w/o Agent-Infra	2.20 (0.72)	2.50 (0.68)	2.53 (0.55)	**2.88 (0.33)**
SAGE	**2.58 (0.59)***	**2.85 (0.36)***	2.75 (0.44)†	**2.88 (0.33)**

Table 2: Human evaluation results of *w/o ICP*, *w/o Agent-Infra*, and *SAGE* for the RAG-based Sales Agent use case. Each condition is rated on a 3-point Likert scale across four dimensions. The best score for each metric is shown in **bold**, standard deviations are in parenthesis. Statistically significant improvements (independent two-sample t-test, $p < 0.05$) over both baselines are marked with *, improvements over one baseline are marked with †.

	Number of unique bugs	
Method	**RAG-based**	**Tool-Augmented**
w/o ICP	29.0 (1.78)	14.4 (2.10)
w/o Agent-Infra	13.6 (1.02)	13.0 (1.02)
SAGE	**38.6 (1.24)**	**18.0 (1.62)**

Table 3: The average counts of unique bugs found over five runs with *w/o ICP*, *w/o Agent-Infra*, and *SAGE*. The highest counts are shown in **bold**, standard deviations are in parenthesis.

w/o Agent-Infra: In this setting, the user simulator is given all knowledge except for bottom-up knowledge from the company's agent infrastructure. It is only provided with high-level corporate overviews and top-down knowledge. This approach mirrors existing methods of creating persona-based user simulators such as DuetSim (Ahmad et al., 2025; Luo et al., 2024).

Use Cases We conduct our experiments on two customer-facing agents: (i) a production-deployed RAG-based agent, and (ii) a tool-augmented shopping agent deployed on the Shopify platform.

RAG-based Sales Agent We simulate interactions with a production-deployed RAG agent used by a robotics company. The agent is designed to provide answers to customer inquiries about the company's products and services using technical product catalogs, FAQs, and corporate information.

Tool-augmented Shopping Agent We also generate interactions with a shopping agent deployed on the Shopify platform. The agent integrates 13 APIs, such as `get_recommendation`, `get_user_info`, `get_product`, `return_item`, and `cancel_order`, alongside FAQs and store policies.

These settings are designed to test the two hypotheses outlined previously. We test each setting using three different LLMs: `gpt-4.1` (OpenAI, 2025a), `gpt-5` (OpenAI, 2025b), and `gemini-2.5-flash` (Comanici et al., 2025) to show *SAGE*'s ability to generalize across different models. The results with `gpt-4.1` are in Section 5, evaluations with `gpt-5` and `gemini-2.5-flash` are in Appendix F.1.

5 Results

The results of our experiments are in the following sections. On average, *SAGE* performs the best across all evaluations, supporting both of the hypotheses outlined in Section 4. This showcases the importance of including both top-down ICP knowl-

w/o ICP	*SAGE*	w/o Agent-Infra
User: I'm interested in the Product_A delivery robot. How does it compare to other delivery solutions? **Agent**: The Product_A delivery robot is . . . For more details, you can explore our website. Which model(s) are you interested in? **User**: I'm mainly interested in delivery solutions and comparing Product_A. Can you tell me more about its benefits over other robots? **Agent**: The Product_A delivery robot excels in indoor environments like restaurants and cafes, . . . **User**: Can it handle different types of deliveries efficiently? **Agent**: Yes, it can efficiently handle various types of deliveries. . . . To speed up your process, would you like to provide your business name? **User**: I'm currently exploring options. Thanks for the info!	**user**: hey, how does Product_A compare with other delivery robots? need it for agriculture operations. **Agent**: Product_A is designed for efficient delivery operations. . . . Which model(s) are you interested in? **User**: i'm trying to evaluate if Product_A is suitable for outdoor environments in farming. what's its durability like? **Agent**: Product_A is designed for robust outdoor use, making it suitable for agricultural operations. . . . Do you want to rent or buy our robots? **User**: how long does it take to charge? i'm considering all options. **Agent**: It takes approximately 3 hours to fully charge from 0 to 100%. . . . To expedite your rental process, could you please provide your business name? **User**: it's AgriFuture Technologies. not renting yet, just exploring options.	**User**: How do the Product_B and Product_C compare for farm work? **Agent**: The Product_B is versatile and can be used for multiple purposes. . . . **User**: What about Product_A robot? **Agent**: The Product_A robot is a delivery robot designed for transporting items. . . . Which model(s) are you interested in? **User**: Delivery bot like Product_A I think. Can it handle outdoor conditions? **Agent**: Currently, our products are primarily designed for indoor business environments, but they may also be suitable for outdoor conditions. . . . **User**: Got it. Thanks for the info.

Table 4: Interaction samples from our human evaluation under three conditions: *SAGE* (middle), *w/o ICP* (left), and *w/o Agent-Infra* (right) with the same user's goal: compare products. Text highlighted in blue reflects generated content tied to infrastructure knowledge, while text in green reflects content consistent with ICP attributes. Agent responses marked in orange indicate faithfulness errors. On the left, user queries in magenta indicate repetitive and generic responses. On the right, the user queries in magenta are ones which are irrelevant to product attributes. Product_A, Product_B and Product_C are placeholders used to mask the actual product names. The complete interaction examples are provided in Table 10.

edge and bottom-up agent infrastructure knowledge to generate high quality simulated interactions.

5.1 Automatic Evaluation

We evaluated both lexical diversity and error identification using simulated interactions between a user and an agent. For each condition and use case, we generated 150 interactions. Lexical diversity was computed directly from these interactions. To assess error identification, we identified the number of unique agent bugs in the interactions using the process in Section 3.4. This process was repeated five times to control for any variation in the bug-finding process. The final results report the average number of bugs discovered.

Our lexical diversity evaluation is in Table 1. *SAGE* performs the best across all diversity metrics with the exception of MTLD score in the Tool-augmented Shopping Agent case. The *w/o ICP* condition does slightly better than the *w/o Agent-Infra* condition in the RAG-based Sales Agent case, although both perform similarly in the Tool-Augmented Shopping Agent case.

Table 3 shows the outcome of our error identification evaluation. *SAGE* identifies the most errors followed by the *w/o ICP* condition. Excluding bottom-up knowledge leads to the largest decrease in errors found, especially in the case with the RAG-based agent. This is likely due to the fact that bottom-up knowledge allows the simulator to ask more specific questions, which helps directly target the agent's knowledge retrieval capabilities.

5.2 Human Evaluation

Our human evaluation of simulator fidelity is based on 120 annotated interactions (40 per condition) between the simulators and the RAG-based Sales Agent. We choose this agent as it is deployed in an actual production environment and therefore represents a more realistic use case for our simulator.

The results of the evaluation are in Table 2. The interactions generated by *SAGE* are significantly better than the other conditions except for specificity and consistency. The simulation method without ICPs performs slightly better in terms of specificity, suggesting that only bottom-up knowledge is essential to achieve high performance. All conditions perform similarly with respect to consistency, indicating that knowledge integration may not be necessary to perform well in this dimen-

sion. The *w/o ICP* condition tends to outperform the *w/o Agent-Infra* condition, implying that agent-infrastructure knowledge may be more important than ICP knowledge for generating high-fidelity simulated interactions in the case of RAG agents.

5.3 Case Study

To further illustrate the differences among simulation methods, we conducted a case study using three interaction samples drawn from our human evaluation. The samples were generated by *SAGE*, *w/o ICP*, and *w/o Agent-Infra* (see Table 4). Each method was applied to the same goal, comparing company products, enabling a direct comparison of simulator fidelity and evaluation effectiveness.

SAGE was the only method that identified agent errors in the interaction. Both of these were faithfulness errors where the agent failed to accurately retrieve information from its knowledge base. Specifically, the agent incorrectly stated that Product_A was "designed for robust outdoor use" and required only 3 hours to charge, when in fact it is intended primarily for indoor use and requires 5.5 hours. The first error was triggered after the simulator asked if Product_A was suitable for outdoor use in farming, a query informed by its top-down ICP knowledge. The second arose when the simulator inquired about charging time, which is based on its bottom-up agent infrastructure knowledge. Together, these cases illustrate how both top-down and bottom-up knowledge integration contribute to identifying different agent errors.

We observe that the simulator *w/o ICP* often produces generic or repetitive responses (see Table 4, first column), resulting in less diverse conversational trajectories and reducing its effectiveness as an evaluation tool. This limitation appears to stem from the use of a more generic persona that lacks the fine-grained attributes provided by ICPs. Such attributes are essential for modeling customer behavior and enabling realistic, contextually useful interactions in business settings (Kivistö, 2024). As a result, the *w/o ICP* simulator tends to underestimate the complexity of real-world user behavior, leading to less useful interactions.

The simulator *w/o Agent-Infra* frequently produces interactions that are misaligned with the company's informational context. For example, as shown in the third column of Table 4, it asks the agent to compare Products B and C for farm work, despite Product_B being designed for serving food and drinks in a restaurant setting and thus irrelevant to farming. Without access to agent infrastructure, the simulator generates less realistic conversations and tends to default to generic queries, which limits its ability to rigorously evaluate agent responses or test their information retrieval capabilities.

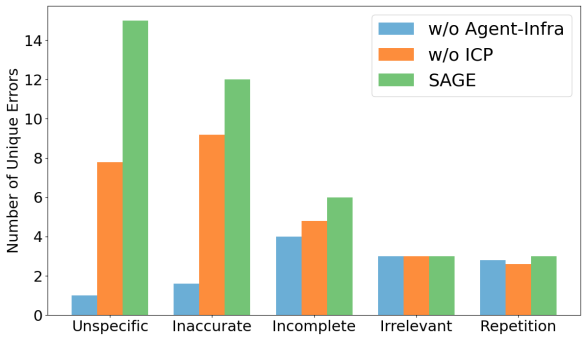

Figure 2: Number of unique errors across high-level error categories for the RAG-based Sales Agent under *w/o ICP*, *w/o Agent-Infra*, and *SAGE* settings.

5.4 Error Analysis

Figure 2 shows the distribution of high-level error categories identified by *SAGE*, *w/o ICP*, and *w/o Agent-Infra* during our evaluation of the RAG-based Sales Agent. The most frequent issues found by *SAGE* fall under the categories of "Unspecific Response" and "Inaccurate Answer," which aligns with the relevant functionalities of a RAG-based system. This demonstrates that *SAGE* effectively captures the agent's most relevant failure modes. The *w/o Agent-Infra* struggles to find errors in these categories, highlighting the importance of incorporating agent infrastructure knowledge to induce and identify errors related to information retrieval. We also observe that the vast majority of errors emerge after the first turn, underscoring the need for multi-turn evaluation when assessing agent behavior.

6 Conclusion

User simulators been proposed as a scalable alternative for evaluating multi-turn interactive agents. However, existing approaches often rely on static benchmarks or model generic users. To address this, we proposed *SAGE*, a knowledge-grounded user simulation framework for multi-turn agent evaluation. *SAGE* combines top-down principles derived from business logic with bottom-up knowledge from agent infrastructure to improve simulation quality. We evaluated our framework using automatic metrics and human assessments, finding that *SAGE* produces interactions that are more diverse, realistic, and useful for agent evaluation.

7 Limitations

One limitation of the *SAGE* framework is that its predefined user attributes are derived from business logic, which mostly relates to customer-facing agents. While these attributes can be changed as needed to adapt to different use cases, developers creating internal-facing agents for education, invoicing, or other back-office tasks may need to customize the attributes for their specific needs.

Our current task scenarios only contain a single user goal and one piece of knowledge selected from agent infrastructure knowledge. Although real customer-agent sessions are often short, users may still switch to new topics without starting a new session. Future work should incorporate multiple goals and multiple knowledge sources within a single session to capture more complex interaction dynamics and better reflect real-world conversation patterns.

Finally, our user simulator evaluation focuses on a cold-start scenario without a direct comparison to real user interactions. The automatic diversity evaluation is limited to reference-free lexical measures, and the fidelity assessment relies on human judgments. Although this setup reflects realistic development conditions where a new agent lacks historical interaction data, future work could benefit from using real-world logs as evaluation references once production-level interaction data becomes available.

References

2025. lexical_diversity.

Adnan Ahmad, Stefan Hillmann, and Sebastian Möller. 2025. Simulating User Diversity in Task-Oriented Dialogue Systems using Large Language Models. *arXiv preprint*. ArXiv:2502.12813 [cs] version: 1.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982.

Michel Borgh, Juan Xu, and Marin Sikkenk. 2020. Identifying, analyzing, and finding solutions to the sales lead black hole: A design science approach. *Industrial Marketing Management*, 88. BANT-criteria (i.e., budget, authority, need, timing) are used to determine prospects.

Nicolas Bougie and Narimasa Watanabe. 2025. SimUSER: Simulating User Behavior with Large Language Models for Recommender System Evaluation. *arXiv preprint*. ArXiv:2504.12722 [cs] version: 1.

Luyu Chen, Quanyu Dai, Zeyu Zhang, Xueyang Feng, Mingyu Zhang, Pengcheng Tang, Xu Chen, Yue Zhu, and Zhenhua Dong. 2025. RecUserSim: A Realistic and Diverse User Simulator for Evaluating Conversational Recommender Systems. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 133–142. ArXiv:2507.22897 [cs].

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3290 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *Preprint*, arXiv:2507.06261.

Sam Davidson, Salvatore Romeo, Raphael Shu, James Gung, Arshit Gupta, Saab Mansour, and Yi Zhang. 2023. User simulation with large language models for evaluating task-oriented dialogue. *Preprint*, arXiv:2309.13233.

Kaustubh Dhole. 2024. KAUCUS - knowledgeable user simulators for training large language models. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)*, pages 53–65, St. Julians, Malta. Association for Computational Linguistics.

Kung-Hsiang Huang, Akshara Prabhakar, Onkar Thorat, Divyansh Agarwal, Prafulla Kumar Choubey, Yixin Mao, Silvio Savarese, Caiming Xiong, and Chien-Sheng Wu. 2025. CRMArena-Pro: Holistic Assessment of LLM Agents Across Diverse Business Scenarios and Interactions. *arXiv preprint*. ArXiv:2505.18878 [cs].

Oliver P. John and Sanjay Srivastava. 1999. The big five trait taxonomy: History, measurement, and theoretical perspectives.

Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. 2025. Mtrag: A multi-turn conversational benchmark for evaluating retrieval-augmented generation systems. *Preprint*, arXiv:2501.03468.

Samira Khodabandehlou, S. Alireza Hashemi Golpayegani, and Mahmoud Zivari Rahman. 2020. An effective recommender system based on personality traits, demographics and behavior of customers in time context. *Data Technologies and Applications*, 55(1):149–174.

Viljami Kivistö. 2024. Customer profiling and purchase decision influencers: an empirical case study from the management consulting industry.

Chuyi Kong, Yaxin Fan, Xiang Wan, Feng Jiang, and Benyou Wang. 2024. PlatoLM: Teaching LLMs in

multi-round dialogue via a user simulator. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7841–7863, Bangkok, Thailand. Association for Computational Linguistics.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *Preprint*, arXiv:2505.06120.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. 2024. DuetSim: Building User Simulator with Dual Large Language Models for Task-Oriented Dialogues. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5414–5424, Torino, Italia. ELRA and ICCL.

Alistair Mavin, Philip Wilkinson, Adrian Harwood, and Mark Novak. 2009. Easy approach to requirements syntax (ears). In *2009 17th IEEE International Requirements Engineering Conference*, pages 317–322.

Philip M. McCarthy and Scott Jarvis. 2010. Mtld, vocd-d, and hd-d: a validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42(2):381–392.

Vinod Muthusamy, Yara Rizk, Kiran Kate, Praveen Venkateswaran, Vatche Isahagian, Ashu Gulati, and Parijat Dube. 2023. Towards large language model-based personal agents in the enterprise: Current trends and open problems. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6909–6921, Singapore. Association for Computational Linguistics.

OpenAI. 2025a. OpenAI: Introducing GPT-4.1 in the api.

OpenAI. 2025b. OpenAI: Introducing GPT-5.

Diksha Panwar, Swati Anand, Farman Ali, and Kanika Singhal. 2019. Consumer decision making process models and their applications to market strategy.

Scott I. Rick, Cynthia E. Cryder, and George Loewenstein. 2008. Tightwads and spendthrifts. *Journal of consumer research*, 34(6):767–782.

Susana Santos and Helena Martins Gonçalves. 2021. The consumer decision journey: A literature review of the foundational models and theories and a future perspective. *Technological Forecasting and Social Change*, 173:121117.

Ivan Sekulić, Silvia Terragni, Victor Guimarães, Nghia Khau, Bruna Guedes, Modestas Filipavicius, André Ferreira Manso, and Roland Mathis. 2024. Reliable LLM-based User Simulator for Task-Oriented Dialogue Systems. *arXiv preprint*. ArXiv:2402.13374 [cs].

Weiyan Shi, Kun Qian, Xuewei Wang, and Zhou Yu. 2019. How to build user simulators to train RL-based dialog systems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1990–2000, Hong Kong, China. Association for Computational Linguistics.

Lu Sun, Shihan Fu, Bingsheng Yao, Yuxuan Lu, Wenbo Li, Hansu Gu, Jiri Gesi, Jing Huang, Chen Luo, and Dakuo Wang. 2025. Llm agent meets agentic ai: Can llm agents simulate customers to evaluate agentic-ai-based shopping assistants? *Preprint*, arXiv:2509.21501.

Yuchong Sun, Che Liu, Kun Zhou, Jinwen Huang, Ruihua Song, Xin Zhao, Fuzheng Zhang, Di Zhang, and Kun Gai. 2024. Parrot: Enhancing multi-turn instruction following for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9729–9750, Bangkok, Thailand. Association for Computational Linguistics.

Olivier Toubia, George Z. Gui, Tianyi Peng, Daniel J. Merlau, Ang Li, and Haozhe Chen. 2025. Twin-2K-500: A dataset for building digital twins of over 2,000 people based on their answers to over 500 questions. *arXiv preprint*. ArXiv:2505.17479 [cs].

Kuang Wang, Xianfei Li, Shenghao Yang, Li Zhou, Feng Jiang, and Haizhou Li. 2025. Know you first and be you better: Modeling human-like user simulators via implicit profiles. *Preprint*, arXiv:2502.18968.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024. HelpSteer2: Open-source dataset for training top-performing reward models. *arXiv preprint*. ArXiv:2406.08673 [cs].

{Art T.} Weinstein. 1994. *Market Segmentation: Using Demographics, Psychographics and Other Niche Marketing Techniques to Predict Customer Behavior*.

Jeanine M. Williamson. 2018. Chapter 1 - Individual Differences. In Jeanine M. Williamson, editor, *Teaching to Individual Differences in Science and Engineering Librarianship*, pages 1–10. Chandos Publishing.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. $$-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *arXiv preprint*. ArXiv:2406.12045.

Qiyuan Zhang, Yufei Wang, Yuxin Jiang, Liangyou Li, Chuhan Wu, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, Fuyuan Lyu, and Chen Ma. 2025. Crowd Comparative Reasoning: Unlocking Comprehensive Evaluations for LLM-as-a-Judge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5059–5074, Vienna, Austria. Association for Computational Linguistics.

A Implementation Details

A.1 Identity Errors

We evaluate agent performance through systematic error identification. The process begins by scoring each agent response at the turn level using an LLM-as-a-Judge approach. Specifically, we employ the `gpt-4.1` model (`gpt-4.1-2025-04-14`) with `temperature=0.1` as the evaluation model. Turns that receive a score below a predefined threshold (e.g., 2 on a 5-point Likert scale) are flagged for further inspection. The threshold is determined through human alignment: annotators review the LLM-provided scores and justifications to establish which values should be considered indicative of an error. This alignment step is critical because different evaluation models exhibit different scoring biases. For example, we find `gpt-5` tends to assign lower helpfulness scores on average than `gpt-4.1`. After human verification, we choose to use 2 as the threshold for result evaluated by `gpt-4.1` model to retain the problematic turns.

Table 5 reports the LLM-as-a-Judge evaluation results on interactions generated by `gpt-4.1` with `temperature=0.1`. Each entry indicates the proportion of turns, across 150 conversations, that received a score below 2. A higher percentage reflects a greater share of turns exhibiting potential issues. The results suggest that *w/o Agent-Infra* produces more problematic turns. However, many of these problematic turns contain duplicate errors, meaning that the *w/o Agent-Infra* condition doesn't actually provide a wider coverage of agent failure modes. We identify the number of unique bugs in order to properly assess the simulators' ability to identify diverse agent errors. Additionally, the bug statements we provide pinpoint concrete failure cases and give actionable insights which are not captured by the LLM judge's justifications. The detailed discussion for interactions generated by other models (`gpt-5` and `gemini-2.5-flash`) are shown in Appendix F.1.

B ICP

We distinguish ICP attributes by Business-to-Business (B2B) and Business-to-Consumer (B2C) contexts, since they differ fundamentally in decision-making processes and relevant information. B2B interactions are typically collective and process-driven, where attributes including firmographics (e.g., industry, company size) or authority are central. While B2C interactions are shaped by individual decision-making styles, such as purchasing preferences, spending behavior (Rick et al., 2008) and loyalty levels.

- Business-to-Business (B2B) Attributes : Following prior work (Kivistö, 2024; Borgh et al., 2020), we identify the most common B2B attributes (with predefined values listed in parentheses): *customer type* (new prospect; returning customer), *discovery channel* (search engine; third party website; friend recommendation; social media), *budget*, *decision-making authority* (primary; secondary), *deal stage* (new leads; demo schedule; after demo; decision make; contract sent; contract signed; paid), *industry*, *company name*, *job information*, and *company size*. Some categories (e.g., authority, customer type) admit small, stable value sets; others (e.g., company name, industry) are generated dynamically.

- Business-to-Consumer (B2C) Attributes: For B2C we reuse the transferable fields from the B2B attributes (customer type, discovery channel, budget) and add other attributes that specific for B2C scenarios. *decision-making style* (rational; intuitive), *purchasing preference* (discount-oriented; value-oriented; quality-oriented), *spending behavior* (trouble spending money; some trouble spending mony; no trouble spending money; some trouble limiting spending; over spending) (Rick et al., 2008), and *loyalty level* (loyal; neutral; disloyal) (Sun et al., 2025).

C Agent Capabilities

Agent capabilities capture the concrete functionalities an interactive agent is designed to perform. In the e-commerce, these functions broadly serve the goal of streamlining consumer decision-making and enabling efficient customer journeys. Following established consumer decision-making models (Panwar et al., 2019; Santos and Gonçalves, 2021), we categorize the capabilities into five categories: *Need Recognition*, *Information Search*, *Evaluation of Alternatives*, *Purchase Decision*, and *Post-Purchase Behavior*. Inspired by real-world systems such as Amazon Rufus and Walmart Sparky, we further refine these capabilities into concrete, testable instructions using a Verb–Object–Prepositional Phrase (constraint) structure (Mavin et al., 2009). The full heuristics are summarized in Table 6.

Method	Helpfulness(%)	Coherence(%)	Verbosity(%)	Relevance(%)	Faithfulness(%)
RAG-based Sales Agent					
w/o ICP	5.18	0.00	7.14	2.14	7.71
w/o Agent-Infra	**8.48**	**3.78**	**13.03**	**6.67**	6.06
SAGE	6.37	0.00	7.29	3.64	**8.56**
Tool-Augmented Shopping Agent					
w/o ICP	7.35	4.68	15.05	8.69	8.19
w/o Agent-Infra	**11.67**	4.66	**22.43**	**12.71**	8.43
SAGE	11.13	**4.73**	15.85	10.82	**9.76**

Table 5: The LLM-as-a-Judge evaluation results for the *w/o ICP*, *w/o Agent-Infra*, and *SAGE* using `gpt-4.1` with temperature `0.1` as user simulator model. Each value denotes the proportion of turns (across 150 conversations) rated below 2 on a 5-point Likert scale. The highest values are shown in **bold**, and the second highest are underlined.

Stages	Agent Capabilities
Need Recognition	Assist purchase planning based on scenario;
Information Search	Recommend products within defined categories and constraints; Answer feature-specific product questions;
Evaluation of Alternatives	Provide comparisons of products;
Purchase Decision	Answer queries about delivery; Answer queries about discount and promotions offers;
Post-Purchase Behavior	Handle queries about order tracking and status updates; Handle reports about product issues or defects; Handle requests for order refund or returns;

Table 6: Heuristics of agent capabilities for the shopping agent.

You are **Zara Khan** [name] who is **conscientious, extroverted, antagonistic, neurotic, and closed to experience** [personality]. As a **32-year-old** [age] **male** [sex], you are a **college graduate with some postgraduate education** [education] and call **Karachi, Pakistan, your home** [citizenship, location]. You bring your skills to bear as **an Event Coordinator with seven years of experience** [job information] in **automating food and drink services** [business type] at large-scale events. Working at **Eats Treats Venue** [company name], a hospitality company with **11-50 employees** [company size], you have **primary decision-making authority** [decision-making authority] and operate within a budget range of $90,000-$**130,000** [budget]. Your personal life is marked by being **a divorced father** [marital status] in **a household of more than four** [household], and you navigate your social and professional spheres through **a democratic lens** [political affiliation]. Despite being rooted in your traditions and having a firm worldview, you maintain a professional curiosity for innovative approaches, particularly those that can elevate the events you coordinate. Having discovered <Company> through **a friend's recommendation** [discovery type], you're interested in exploring their service robotics solutions, recognizing the potential benefits they can offer in seamlessly integrating into your venue's operations. **As a new prospect** [customer type] **at the new leads stage** [deal stage], you have a keen eye on how technologies can streamline your services, and you're poised to enhance your venue's offerings in the competitive **hospitality industry** [business type].

Figure 3: A user profile example

D Profile Sample

The complete user profile in Figure 1 is shown in Figure 3. The corresponding attribute categories are in brackets. The sensitive information about the company is masked through human review.

E Human Evaluation Details

For human evaluation, we recruited two graduate students as annotators. Both had prior experience with developing interactive agents, which qualified them to perform the task. The annotators conducted the evaluation collaboratively, discussing each case until they reached a consensus score. Detailed annotation guidelines are provided below. Our data collection protocols are IRB approved.

1. Human-Likeness
Definition: How closely the simulated user's language resembles that of a human.
Scale (1–3):
1 = Clearly artificial, awkward, or robotic phrasing.
2 = Somewhat human-like with occasional awkwardness or unnatural phrasing.
3 = Highly natural; indistinguishable from a real user.

2. Coherence
Definition: How well the simulated user's utterances follow logically from the dialogue history.
Scale (1–3):
1 = Often off-topic or logically inconsistent.
2 = Mostly coherent with occasional minor lapses.
3 = Fully coherent throughout.

3. Specificity
Definition: Whether the simulated user provides enough relevant and precise information for the system to proceed with the task.
Scale (1–3):
1 = Missing essential details or providing irrelevant content.
2 = Provides partial or vague information.
3 = Provides all necessary, relevant, and concise information.

4. Consistency
Definition: Whether the simulated user remains consistent in facts, preferences, constraints, and style across turns.
Scale (1–3):
1 = Contradicts previous statements or changes facts/preferences without reason.
2 = Mostly consistent with minor contradictions or shifts.
3 = Fully consistent across the dialogue.

F Results Analysis

F.1 Automatic Evaluation

We use the `lexical_diversity` package (lex, 2025) to calculate the diversity score. Table 7 reports the results for interactions using `gpt-5` with the default hyperparameter settings, Table 8 presents results for `gemini-2.5-flash` with `temperature=0.1`. All models are called through APIs following their terms of use. Across both models and use cases, *SAGE* consistently achieves the best or comparable performance on most metrics.

Table 9 summarizes the average number of unique bugs identified over five runs for interactions generated by `gpt-5` and `gemini-2.5-flash`. Consistent with our previous results, *SAGE* uncovers the highest number of unique errors, followed by the *w/o ICP* setting. Removing bottom-up knowledge leads to the sharpest drop in error identification. These findings demonstrate that *SAGE* is robust across different models.

F.2 The Agent Errors Analysis

Figure 4 shows the distribution of high-level error categories identified by *SAGE*, along with when these errors occurred within the interaction. We observe that most errors occur after the first turn, particularly "Incomplete Response", "Irrelevant Response" and "Repetition Defect", which mostly emerge in non-first turns. This highlights the importance of multi-turn evaluation for reliably assessing agent behavior. This finding is consistent with prior work showing that LLMs struggle to stay on track in extended dialogues and have difficulty recovering once errors occur (Laban et al., 2025). Table 11 shows more interaction examples for each high-level error category.

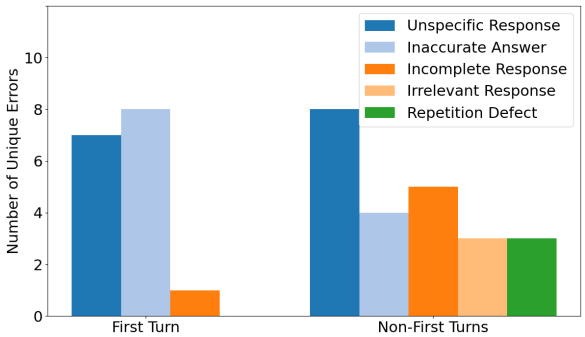

Figure 4: The number of unique errors in each category, grouped by their occurrence in the first turn versus subsequent turns.

Method	MTLD↑	HDD↑	logTTR↑	rootTTR↑	Vocab Size↑	Distinct@1/2/3↑
RAG-based Sales Agent						
w/o ICP	84.17	**0.86**	0.77	11.87	1076	**0.20/0.60/0.81**
w/o Agent-Infra	**94.01**	**0.86**	0.76	10.56	1053	0.16/0.54/0.76
SAGE	88.01	**0.86**	**0.78**	**12.26**	**1150**	**0.20/0.60/0.81**
Tool-Augmented Shopping Agent						
w/o ICP	66.97	0.85	0.75	10.07	947	**0.18/0.58/0.79**
w/o Agent-Infra	**77.89**	**0.86**	0.75	9.81	932	0.17/0.57/0.78
SAGE	72.57	**0.86**	**0.76**	**10.45**	**951**	**0.18/0.58/0.79**

Table 7: Lexical diversity results of *w/o ICP*, *w/o Agent-Infra* and *SAGE* for the RAG-based Sales Agent and Tool-augmented Shopping Agent cases using `gpt-5`. The best score for each metric is shown in **bold**.

Method	MTLD↑	HDD↑	logTTR↑	rootTTR↑	Vocab Size↑	Distinct@1/2/3↑
RAG-based Sales Agent						
w/o ICP	91.14	0.87	0.77	15.41	2241	0.15/0.53/0.77
w/o Agent-Infra	101.41	0.88	0.76	15.97	2708	0.13/0.50/0.76
SAGE	**105.08**	**0.89**	**0.78**	**17.55**	**2765**	**0.16/0.55/0.79**
Tool-Augmented Shopping Agent						
w/o ICP	74.94	**0.87**	**0.75**	**12.88**	1755	0.14/0.48/0.73
w/o Agent-Infra	76.58	0.86	**0.75**	12.58	1743	0.13/0.48/0.73
SAGE	**77.52**	**0.87**	**0.75**	**12.88**	**1851**	**0.14/0.48/0.74**

Table 8: Lexical diversity results of *w/o ICP*, *w/o Agent-Infra* and *SAGE* for the RAG-based Sales Agent and Tool-augmented Shopping Agent cases using `gemini-2.5-flash`. The best score for each metric is shown in **bold**.

	GPT-5		Gemini-2.5-Flash	
Method	**RAG-based**	**Tool-Augmented**	**RAG-based**	**Tool-Augmented**
w/o ICP	27.4 (1.02)	19.2 (1.31)	22.6 (0.33)	20.8 (0.75)
w/o Agent-Infra	14.6 (0.73)	17.8 (0.92)	16.0 (1.03)	18.6 (2.01)
SAGE	**28.8 (1.12)**	**24.0 (0.61)**	**27.8 (1.70)**	**24.2 (1.47)**

Table 9: The average counts of unique bugs found over five runs with *w/o ICP*, *w/o Agent-Infra*, and *SAGE* using `gpt-5` and `gemini-2.5-flash` as the user simulator models. The highest counts are shown in **bold**, standard deviations are in parenthesis.

w/o ICP	*SAGE*	w/o Agent-Infra
User: I'm interested in the Product_A delivery robot. How does it compare to other delivery solutions? **Agent**: The Product_A delivery robot is designed for efficient indoor deliveries, thriving in dynamic environments like restaurants and cafes. It uses smart obstacle avoidance and can interact with elevators for secure deliveries. Compared to other solutions, Product_A offers cost-effectiveness and reliability, reducing labor costs and enhancing service speed. For more details, you can explore our website. Which model(s) are you interested in? **User**: I'm mainly interested in delivery solutions and comparing Product_A. Can you tell me more about its benefits over other robots? **Agent**: The Product_A delivery robot excels in indoor environments like restaurants and cafes, offering smart obstacle avoidance and elevator interaction for secure deliveries. Compared to other robots, Product_A is cost-effective, reliable, and enhances service speed, reducing labor costs. **User**: Can it handle different types of deliveries efficiently? **Agent**: Yes, it can efficiently handle various types of deliveries. It excels in indoor environments, using smart navigation and obstacle avoidance to ensure timely deliveries. This enhances service speed and reduces labor costs, making it a versatile solution for businesses. To speed up your process, would you like to provide your business name? **User**: I'm currently exploring options. Thanks for the info!	**user**: hey, how does Product_A compare with other delivery robots? need it for agriculture operations. **Agent**: Product_A is designed for efficient delivery operations. It enhances operational efficiency and optimizes labor resources. Which model(s) are you interested in? **User**: i'm trying to evaluate if Product_A is suitable for outdoor environments in farming. what's its durability like? **Agent**: Product_A is designed for robust outdoor use, making it suitable for agricultural operations. It is built to withstand various environmental conditions, ensuring durability and reliability in farming tasks. It's battery life varies based on usage, but it typically runs for several hours before needing a recharge. For more specific details, please visit our website. Do you want to rent or buy our robots? **User**: how long does it take to charge? i'm considering all options. **Agent**: It takes approximately 3 hours to fully charge from 0 to 100%. This efficient charging time ensures minimal downtime for your agricultural operations. For more details, please visit the our website. To expedite your rental process, could you please provide your business name? **User**: it's AgriFuture Technologies. not renting yet, just exploring options.	**User**: How do the Product_B and Product_C compare for farm work? **Agent**: The Product_B is versatile and can be used for multiple purposes, including making tea, coffee, and cocktails. However, it is not specifically designed for farm work. The Product_C is a delivery robot, primarily intended for transporting items, which might not be suitable for farm tasks. If you have specific requirements for farm work, please let me know, and I can provide more tailored information! **User**: What about Product_A robot? **Agent**: The Product_A robot is a delivery robot designed for transporting items, primarily in business settings. It enhances operational efficiency and optimizes labor resources. For more details, visit our website. We have amazing worker, delivery, cleaning, and multipurpose robots. Which model(s) are you interested in? **User**: Delivery bot like Product_A I think. Can it handle outdoor conditions? **Agent**: Currently, our products are primarily designed for indoor business environments, but they may also be suitable for outdoor conditions, such as agricultural settings. If you have specific requirements for outdoor use, please contact our customer service team for more details. **User**: Got it. Thanks for the info.

Table 10: Full interaction samples from our human evaluation under three conditions: *SAGE* (middle), *w/o ICP* (left), and *w/o Agent-Infra* (right) with the same user's goal: compare products. Text highlighted in blue reflects generated content tied to infrastructure knowledge, while text in green reflects content consistent with ICP attributes. Agent responses marked in orange indicate faithfulness errors. On the left, user queries in magenta indicate repetitive and generic responses. On the right, the user queries in magenta are ones which are irrelevant to product attributes. Product_A, Product_B and Product_C are placeholders used to mask the actual product names.

Unspecific Response
Bug Description: Did not provide the specific weight capacity for Product_A when requested
Example: *[Earlier parts omitted for brevity]* **User**: hmm. what about weight? Can it carry heavy items or no? **Agent**: The Product_A is designed to handle a variety of delivery tasks, but it has weight limitations. For specific weight capacity details, please refer to the product specifications on our website or contact our customer service for more information. **User**: ugh, don't have time for that. can't you just give me a number here? **Agent**: The Product_A is capable of handling various delivery tasks, but it does have specific weight limitations. For detailed information on its weight capacity, please check the product specifications on our website or contact our customer service.
Inaccurate Answer
Bug Description: Incorrectly claimed Company_name ships internationally
Example: **User**: Hey, do you guys offer international shipping? **Agent**: Yes, we offer international shipping for our robots. Delivery times may vary depending on the destination. If you have any specific questions or need further assistance, feel free to ask! **User**: Wait, are you sure? I thought it was different. **Agent**: I apologize for the confusion earlier. Currently, Company_name only operates within the US and does not offer international shipping. If you have any other questions or need further assistance, feel free to ask! **User**: That's disappointing. Can you be more consistent next time?
Incomplete Response
Bug Description: Did not provide requested product comparison (e.g., Product_C vs similar robots)
Example: **User**: Can you tell me how Product_C compares with similar delivery robots? **Agent**: The Product_C delivery robot is designed for efficient and reliable delivery services in business environments. It offers advanced navigation and obstacle avoidance, making it suitable for various settings. While I can't provide a direct comparison with other specific delivery robots, Product_C is known for its robust performance and adaptability. If you have more specific questions about its features or capabilities, feel free to ask! **User**: I asked about a comparison, not just more features... Can you please be more useful?
Irrelevant Response
Bug Description: Requested personal contact information and/or created support tickets without user consent or clear necessity
Example: **User**: What's the cleaning efficiency of the Product_D? And how does it navigate? **Agent**: The Product_D is designed for commercial cleaning with high efficiency, utilizing advanced sensors and mapping technology for precise navigation. If you have more questions or need further assistance, feel free to ask! **User**: This isn't what I asked for. Can you provide more details on these sensors and its exact cleaning efficiency? **Agent**: The Product_D cleans at a rate of 21,958 sqft/hr. It uses a combination of sensors, including a monocular panoramic camera, binocular camera, LiDAR, 3D ToF camera, ultrasonic wave, IMU, and electronic bumper for navigation. For more details, you can visit the Company_name website. I'm sorry to hear that! We are creating a support ticket now. Could you provide me with your email address and phone number?
Repetition Defect
Bug Description: Repeated the same non-answer across turns despite user frustration
Example: **User**: hey, can u tell me the shipment status with the tracking number i got? **Agent**: I'm sorry, but I can't access shipment details directly. Please use the tracking number you received via email to check your order status online. If you need further assistance, feel free to ask! **User**: Uhh, yeah I got the tracking number in the email, just need some details about the current status! **Agent**: I'm sorry, but I can't access shipment details directly. Please use the tracking number you received via email to check your order status online. If you need further assistance, feel free to ask! *[Later interaction repeat the same answer multiple times]*

Table 11: The synthesized interaction examples generated by the *SAGE* framework. The sentences highlighted in orange represent the agent errors for the corresponding error category.

G Prompts

G.1 Profile Construction

Augment attributes with the agent infrastructure knowledge

Given the agent infrastructure knowledge, your task is to generate {num_personas} set of personal attributes for the new categories.

Examples:
#######################
Generate 2 sets of attributes.
agent infrastructure knowledge:
{knowledge}

Here are new attribute descriptions that you need to generate values for:
business_type: The type of business or industry the individual is associated with, such as technology, finance, healthcare, retail, or manufacturing.
company_name: The name of the company or organization the individual is associated with.
job_information: Detailed information about the individual's professional roles, including their official job title, primary responsibilities, and total years of relevant work experience within the industry or field. Such as marketing manager with 7 years of experience in the industry, taking care of exploration and expansion of the market.
company_size: The size of the company or organization the individual is associated with, better with the number of employees.

Return:
{{"attributes": [{{"business_type": "commercial cleaning services", "company_name": "House Cleaner", "job_information": "Facilities Manager with 8 years of experience in commercial cleaning services, managing cleaning staff and implementing automation solutions", "company_size": "100-200 employees"}}, {{"business_type": "hospitality (hotels, bars, restaurants)", "company_name": "Lily Motel", "job_information": "Food Beverage Director with 10 years of experience in hospitality, overseeing bar and restaurant operations and technology adoption", "company_size": "11-50 employees"}}]}}

Rules:
- You should return a JSON list of {num_personas} sets of attributes values.
- The result should be diverse and not repetitive.
- The attributes values should be aligned with the corporate overview.

#######################
Generate {num_personas} sets of attributes.

agent infrastructure knowledge:
{knowledge}

Here are the categories that you need to generate values for, one value per category:
{category_description}

Return:

Validate attributes

Given a set of user attributes, your task is to check whether each pair of attributes is consistent. You should first generate the thought and then return the response "Yes" or "No" in JSON FORMAT.

Example:
#######################
User attributes:
job_information: operations manager with 10 years of experience in hospitality, overseeing automation and service efficiency
business_type: commercial cleaning services
company_size: 11-50 employees
company_name: Caesars Entertainment

Return:
{{
"thought": "a commercial cleaning services company will not name as Entertainment company",
"response": "No"
}}
#######################
User attributes:
customer_type: returning customer
deal_stage: new leads

Return:
{{
"thought": "a returning customer will not be a new leads",
"response": "No"
}}

#######################
User attributes:
{user_attributes}

Return:

Convert user attributes to a user profile

Given a list of user attributes, your task is to convert them into a natural language description of a user profile.

Rules:
- Write the response in the second person.
- Begin the profile with: "You are <name> who is <TRAIT 1, ..., TRAIT 5>.", where [TRAIT 1, ..., TRAIT 5] represents the assigned Big Five personality traits.
- Do not add any prefixes such as "Profile:" or "Customer Profile:".

User attributes:
{user_attr}

Profile:

G.2 User simulator prompt

User simulator's system prompt

You are a customer interacting with an agent. The agent is supplied by the following company:
{summary}
{profile}
You have the following goal when interacting with this agent: {goal}
Here is the content that you might be interested in and might have questions about:
{knowledge}

Rules:
- Just generate one line at a time to simulate the user's message. Keep the response brief and concise.
- Do not give away all the instruction at once. Only provide the information that is necessary for the current step.
- Do not hallucinate information that is not provided in the instruction. For example, if the agent asks for the order id but it is not mentioned in the instruction, do not make up an order id, just say you do not remember or have it.
- If the instruction goal is satisified, generate '###STOP###' as a standalone message without anything else to end the conversation.
- Do not repeat the exact instruction in the conversation. Instead, use your own words to convey the same information.
- Try to personalize your question to the your background, such as job information, household, event or activities you did or going to do.
- Adjust your tone/style based on your personalities.
- As a customer, you tend to write short questions with occasional typos or incomplete sentences. Replicate the chitchat typing behavior of a human customer and begin the conversation with a question to achieve your goal.

G.3 Agent Evaluation Metrics

All the agent evaluation prompts start with the **TASK_INSTRUCTION** followed by the specific rubrics for each metric.

TASK_INSTRUCTION

Given a conversation between a user and an AI assistant, your task is to evaluate the last message from the AI assistant according to the criteria below, assigning a score from 0 to 4. Follow the detailed descriptions to guide your assessment. Return the score and reasoning in JSON FORMAT.

Format:
{{
"score": <integer 0-4>,
"reason": "<reasoning>"
}}

Helpfulness

[TASK_INSTRUCTION]
Helpfulness:
Description: Determine the effectiveness and value of the AI assistant's responses in the last message in addressing the user's needs.
Evaluation Criteria:
• 4 - The response is extremely helpful and completely aligned with the spirit of what the prompt was asking for.
• 3 - The response is mostly helpful and mainly aligned with what the user was looking for, but there is still some room for improvement.
• 2 - The response is partially helpful but misses the overall goal of the user's query/input in some way. The response did not fully satisfy what the user was looking for.
• 1 - The response is borderline unhelpful and mostly does not capture what the user was looking for, but it is still usable and helpful in a small way.
• 0 - The response is not useful or helpful at all. The response completely missed the essence of what the user wanted.

Here is the conversation between user and AI assistant:
{conversation}

Evaluation Form:

Coherence

[TASK_INSTRUCTION]
Coherence:
Description: Determine the logical flow and consistency of the AI assistant's responses in the last message.
Evaluation Criteria:
• 4 (Perfectly Coherent and Clear) - The response is perfectly clear and self-consistent throughout. There are no contradictory assertions or statements, the writing flows logically and following the train of thought/story is not challenging.
• 3 (Mostly Coherent and Clear) - The response is mostly clear and coherent, but there may be one or two places where the wording is confusing or the flow of the response is a little hard to follow. Over all, the response can mostly be followed with a little room for improvement.
• 2 (A Little Unclear and/or Incoherent) - The response is a little unclear. There are some inconsistencies or contradictions, run on sentences, confusing statements, or hard to follow sections of the response.
• 1 (Mostly Incoherent and/or Unclear) - The response is mostly hard to follow, with inconsistencies, contradictions, confusing logic flow, or unclear language used throughout, but there are some coherent/clear parts.
• 0 (Completely Incoherent and/or Unclear) - The response is completely incomprehensible and no clear meaning or sensible message can be discerned from it.

Here is the conversation between user and AI assistant:
{conversation}

Evaluation Form:

Verbosity

[TASK_INSTRUCTION]
Verbosity:
Description: How concise is the last message from the AI assistant. Does it use more words than needed?
Evaluation Criteria:
• 4 (Verbose) - The response is particularly lengthy, wordy, and/or extensive with extra details given what the prompt requested from the assistant model. The response can be verbose regardless of if the length is due to repetition and incoherency or if it is due to rich and insightful detail.
• 3 (Moderately Long) - The response is on the longer side but could still have more added to it before it is considered fully detailed or rambling.
• 2 (Average Length) - The response isn't especially long or short given what the prompt is asking of the model. The length is adequate for conveying a full response but isn't particularly wordy nor particularly concise.
• 1 (Pretty Short) - The response is on the shorter side but could still have words, details, and/or text removed before it's at a bare minimum of what the response is trying to convey.
• 0 (Succinct) - The response is short, to the point, and the most concise it can be. No additional information is provided outside of what is requested by the prompt (regardless of if the information or response itself is incorrect, hallucinated, or misleading. A response that gives an incorrect answer can still be succinct.).

Here is the conversation between user and AI assistant:
{conversation}

Evaluation Form:

Relevance

[TASK_INSTRUCTION]
Relevance:
Description: How much the last message from the AI assistant is addressing the question/input from the customer and to what degree?
Evaluation Criteria:
• 4 (Very Relevant) - The response is addressing the input question in its entirety, and just the question.
• 3 (Additional information) - The response is fully addressing the question, but also providing information not directly related to the question alongside.
• 2 (Partially Relevant) - The response is partially addressing the question i.e., only a part of the question is addressed.
• 1 (Pretty Short) - The response only address small portion of the question.
• 0 (Not Relevant) - The response is not relevant to the user's query/input at all.

Here is the conversation between user and AI assistant:
{conversation}

Evaluation Form:

Faithfulness

[TASK_INSTRUCTION]
Faithfulness:
Description: How accurate/factual the answer seems to be, based on: provided knowledge content and majorly agreed common knowledge.
Evaluation Criteria:
• 4 - The response is completely correct and faithfulness to the knowledge with no necessary details missing and without false, misleading, or hallucinated information. If the prompt asks the assistant to do a task, the task is completely done and addressed in the response.
• 3 - The response is mostly correct and faithfulness to the knowledge with a small amount of missing information. It contains no misleading information or hallucinations. If the prompt asks the assistant to perform a task, the task is mostly successfully attempted.
• 2 - The response contains a mix of correct and incorrect information. The response may miss some details, contain misleading information, or minor hallucinations, but is more or less aligned with what the prompt asks for. If the prompt asks the assistant to perform a task, the task is attempted with moderate success but still has clear room for improvement.
• 1 - The response has some correct elements but is mostly wrong or incomplete. The response may contain multiple instances of hallucinations, false information, misleading information, or irrelevant information. If the prompt asks the assistant to do a task, the task was attempted with a small amount of success.
• 0 - The response is completely incorrect. All information provided is wrong, false or hallucinated. If the prompt asks the assistant to do a task, the task is not at all attempted, or the wrong task was attempted in the response. The response is completely irrelevant to the prompt.

Here is the knowledge that assistant's response should be grounded:
{knowledge}

Here is the conversation between user and AI assistant:
{conversation}

Evaluation Form:

G.4 Identify Errors

In order to identify more specific agent errors, we adopt two-stage process. In the first stage, we prompt an LLM to generate high level failure categories using the problematic turns and their score justifications. In the second stage, the output from the first stage along with the justifications are given to an LLM to generate fine-grained, unique bug statement.

identify high-level error categories

Given the failure cases from user-agent interactions, your task is to categorize them into high-level failure categories.

Rules:
- For each case, first provide a short reasoning statement (“thought”) explaining why the case belongs to a particular category.
- Assign a concise and meaningful name to the high-level failure category.
- Provide a clear description that characterizes the nature of this category.
- Return the final output strictly in JSON format.

Format:
{{"errors": [{{"thought": "<brief reasoning for the categorization>", "high_level_error_category": "<name of the category>", "description": "<definition of the category>"}}]}}

Failure cases:
{items}

Return:

identify unique errors

Given the failure cases from user-agent interactions, and the high-level failure categories, your task is to deduplicate these failure cases and identity fine-grained unique errors under each high-level failure category.

Rules:
- Do not follow the error types from "helpfulness", "coherence", "verbosity", "faithfulness", "relevance" anymore. Instead, concentrate specifically on the agent’s behavior. For example, identify which specific pieces of information are missing. Each missing piece of information should be treated as a separate unique error.
- Return the error types in the JSON format, where high_level_error_category is one of the high-level error categories from the choice list, unique_error_description is the description of the unique error, example is item_id corresponding to that unique error.

Format:
{{"unique_errors": [{{"high_level_error_category": "<high-level error category name>", "unique_error_description": "<unique_error_description>", "example": ["<item_id1>, <item_id2>, ..."]}}]}}

High-level failure categories:
{high_level_failure_categories}

Failure cases:
{items}

Return: