ShoppingBench: A Real-World Intent-Grounded Shopping Benchmark for LLM-based Agents

Jiangyuan Wang [1*], **Kejun Xiao**[1*], **Qi Sun**[1†],
Huaipeng Zhao[1], **Tao Luo**[1], **Jian Dong Zhang**[1], **Xiaoyi Zeng**[1]

[1]Alibaba International Digital Commercial Group
{wangjiangyuan.wjy, xiaokejunkejun.xia, qiran.sq}@alibaba-inc.com

Abstract

Existing benchmarks in e-commerce primarily focus on basic user intents, such as finding or purchasing products. However, real-world users often pursue more complex goals, such as applying vouchers, managing budgets, and finding multi-products seller. To bridge this gap, we propose ShoppingBench, a novel end-to-end shopping benchmark designed to encompass increasingly challenging levels of grounded intent. Specifically, we propose a scalable framework to simulate user instructions based on various intents derived from sampled real-world products. To facilitate consistent and reliable evaluations, we provide a large-scale shopping sandbox that serves as an interactive simulated environment, incorporating over 2.5 million real-world products. Experimental results demonstrate that even state-of-the-art language agents (such as GPT-4.1) achieve absolute success rates under 50% on our benchmark tasks, highlighting the significant challenges posed by our ShoppingBench. In addition, we propose a trajectory distillation strategy and leverage supervised fine-tuning, along with reinforcement learning on synthetic trajectories, to distill the capabilities of a large language agent into a smaller one. As a result, our trained agent achieves competitive performance compared to GPT-4.1.

Code — https://github.com/yjwjy/ShoppingBench

Introduction

Large language models (LLMs) have empowered agents with impressive abilities in task automation and decision-making, leading to growing interest from both academia and industry (Yao et al. 2024). In recent years, a variety of agent benchmarks have been introduced to systematically assess language agent performance across different scenarios. These benchmarks typically focus on evaluating end-to-end capabilities such as task planning, tool using, and reasoning. As a highly practical field with broad application prospects, e-commerce has naturally become a key focus for evaluating agent capabilities.

However, existing benchmarks for evaluating language agents in e-commerce primarily focus on straightforward user intents such as locating and purchasing products (Zhou et al. 2023; Yao et al. 2022a). In practice, e-commerce users often pursue multifaceted goals that extend beyond mere product acquisition. For example, as shown in Figure 1, language agent is expected to optimize discounts, combine multiple orders to qualify for free shipping, or verify total expenditures against budget constraints. Such grounded user intents require language agents to perform multi-step reasoning, effectively utilize domain-specific knowledge, and leverage external tools to fulfill complex user instructions. Despite increasing interest in language agents as autonomous decision-makers (Mialon et al. 2023), current agent benchmarks in e-commerce rarely incorporate these realistic and nuanced user intents.

Beyond above agent benchmarks, previous e-commerce datasets primarily address isolated or narrowly scoped downstream tasks (Jin et al. 2023; Reddy et al. 2022; Yangning et al. 2023; Liu et al. 2023; Jia et al. 2022). While large-scale benchmarks such as Shopping MMLU (Jin et al. 2024) and ChineseEcomQA (Chen et al. 2025) have been proposed based on large e-commerce corpora, they mainly focus on question answering(Wang et al. 2025) and skill-based evaluation rather than end-to-end agent performance. This limits their effectiveness in assessing a language agent's ability to fulfill complex user intents in real-world shopping scenarios.

To bridge the above gaps, we propose ShoppingBench, a large-scale end-to-end shopping benchmark comprising 3,310 user instructions, designed to encompass progressively challenging levels of grounded intent in shopping scenarios. Specifically, we propose a scalable framework to simulate user instructions based on various intents derived from sampled real-world products. To facilitate consistent and reliable evaluations, we provide a large-scale e-commerce shopping sandbox that serves as an interactive simulated environment, incorporating over 2.5 million real-world products. To automatically evaluate the quality of language agents, we propose a series of new metrics based on different intent constraints.

In addition, we also propose a trajectory distillation strategy, where tool-use trajectories are generated by the GPT-4.1, and using rejection sampling to filter low-quality trajectories. Then, we use these synthetic trajectories to train Qwen3-4B with Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL), which can significantly improve the performance. As a result, our trained language agent achieves com-

[*]These authors contributed equally.
[†]Corresponding author.

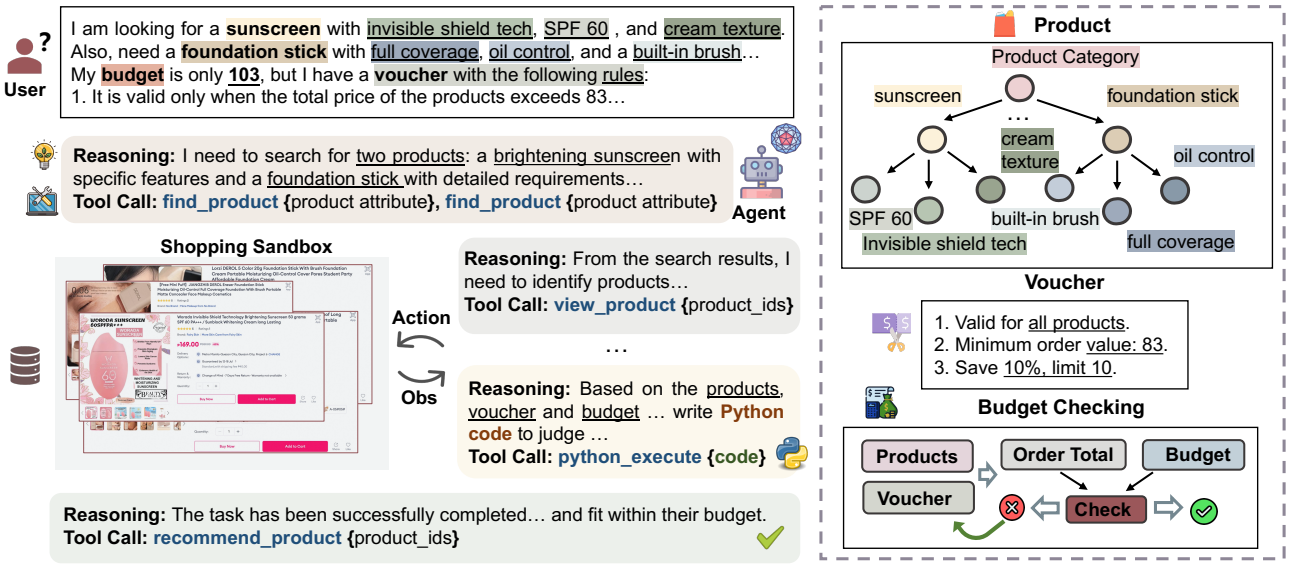


Figure 1: An illustration to depict a real-world user instruction with complex intent. Unlike previous agent benchmarks that solely focus on basic product purchases, ours incorporates coupon usage and optimal product combination within a budget.

petitive performance compared to GPT-4.1 agent.

Our experiments illustrate that even the best-performing language agent (GPT-4.1-based) achieves a success rate below 50%, underscoring the challenge of our benchmark. Quantitative and qualitative analysis of failure cases reveals existing agents' limitations in understanding user instruction with complex intent and choosing appropriate tools. These findings underscore the need for advances in agent architecture, tool usage, problem decomposition, and web information integration.

Our contribution can be summarized as follows:

- We propose a scalable framework to simulate diverse user instructions and provide a sandbox environment with over 2.5 million products for consistent and interactive evaluation.
- We propose new automatic evaluation metrics, grounded in intent constraints, to rigorously assess language agents in e-commerce shopping tasks.
- We propose a trajectory distillation strategy to synthesize training data, filter out low-quality trajectories using our proposed automatic evaluation metric, and then use SFT and RL to efficiently distill GPT-4.1's abilities into a smaller model, which achieves comparable performance.
- We evaluate 17 existing language agents, along with our fine-tuned Qwen3-4B agent. Even the best-performing model, GPT-4.1, achieves a success rate below 50%, highlighting the challenge of our benchmark.

Related Work

Existing benchmarks related to e-commerce shopping can generally be categorized into two types: agent benchmarks and task-oriented dialogue benchmarks.

Agent Benchmarks

Recent advances in language agents have generated significant interest regarding their potential to drive unprecedented automation across diverse industries (Chen et al. 2021; Yao et al. 2022b). Evaluating the capabilities of language models as agents requires examining their ability to aggregate information for multi-step reasoning and autonomous decision-making, as well as their proficiency in effective tool utilization (Huang et al. 2023; Schick et al. 2023; Yao et al. 2024). Recent research focus on developing domain-specific agents (Mialon et al. 2023) to address these challenges. E-commerce represents an especially realistic and pressing application scenario for language agents. Thus, more recent studies adopted web shopping (Zhou et al. 2023; Yao et al. 2022a) as a key benchmark domain to evaluate the capabilities of language agents in fulfilling user purchase requests. Most existing benchmarks for agents in e-commerce scenarios primarily focus on evaluating the user's basic intent, namely, the successful purchase of a product. These benchmarks typically adopt a web shopping setting and define task success based on whether an order can be successfully placed.

However, in real-world e-commerce scenarios, user intents often extend beyond the basic goal of finding products. More complex and realistic intents, such as combining orders for free shipping or optimizing for coupon discounts, require language models to reason with specific e-commerce knowledge that are not adequate in existing agent benchmarks. To bridge this gap, we propose ShoppingBench, an end-to-end shopping benchmark for language agent, which grounds a wide range of realistic user intents.

E-commerce Datasets

Previous e-commerce benchmarks mainly comprise isolated or narrowly related downstream tasks (Jin et al. 2023; Reddy et al. 2022; Yangning et al. 2023; Liu et al. 2023; Jia et al.

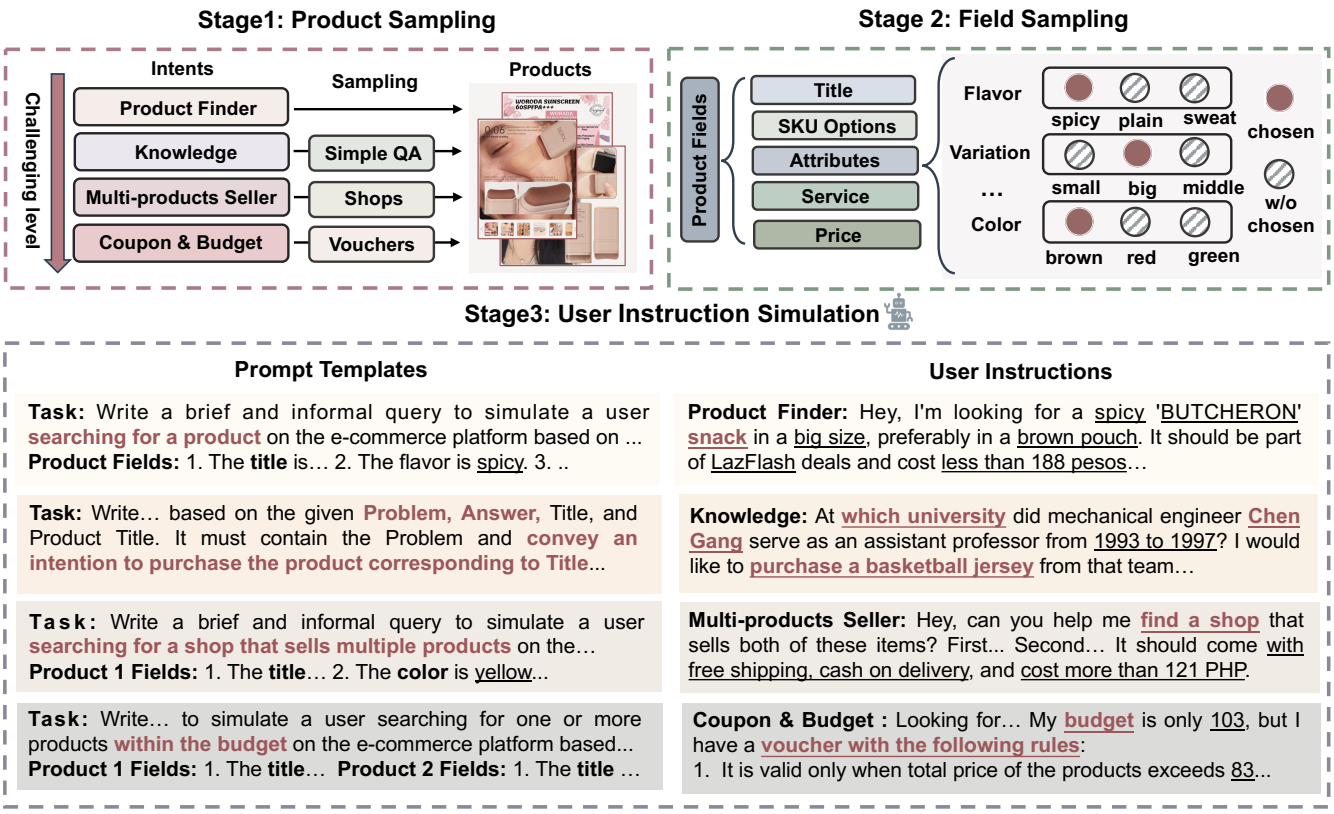


Figure 2: Construction of our ShoppingBench. Our benchmark encompasses four types of real-world user purchase intents: Products Finder, Knowledge, Multi-products seller, and Coupon & Budget , with complexity increasing progressively.

2022). Recently, multidimensional benchmarks such as Shopping MMLU (Jin et al. 2024) and ChineseEcomQA (Chen et al. 2025) have been constructed based on comprehensive e-commerce corpora. EcomScriptBench (Wang et al. 2025) is proposed to evaluate the ability of language models to generate plans with scripts and recommend products. However, these benchmarks primarily focus on conceptual and skill-based question answering in e-commerce, which poses challenges for the end-to-end evaluation of e-commerce agents.

Problem Formulation

Each trajectory can be represented as a partially observable Markov decision process (POMDP) $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R})$. It consists of the following components: a natural language instruction space $\mathcal{U}$, a state space $\mathcal{S}$, an action space $\mathcal{A}$, an observation space $\mathcal{O}$, a transition function $\mathcal{T}$, and a reward function $\mathcal{R}$. When a shopping agent performs an action, it interacts with the environment by invoking tools, which generates observations and updates the state. This process can be mathematically represented as:

$$(s_{t+1}, o_{t+1}) = \mathcal{T}(s_t, a_t)$$

where s_t represents the state at time step t, a_t denotes the action executed by the agent, $\mathcal{T}$ is the transition function, s_{t+1} is the updated state, and o_{t+1} is the observation.

At the terminal state, we evaluate whether the predicted products in the terminal state, s_T, satisfy all the requirements specified in the user's instructions, $\mathcal{U}$. The task is deemed successfully completed if:

$$\text{success} = \begin{cases} 1, & \text{if all conditions in } \mathcal{U} \text{ are met in } s_T, \\ 0, & \text{otherwise.} \end{cases}$$

ShoppingBench Construction

In this section, we introduce the construction of our ShoppingBench, which consists of three key components: a simulated interactive environment, intent-grounded user instructions, and a predefined tool set.

Grounded Shopping Intention

As shown in Figure 2, our ShoppingBench includes the following four real-world user purchase intents, with the challenges progressively increasing for each intent.

Products Finder. The language agent needs to find the corresponding product based on the user's description of the product attributes.

Knowledge. The language agent needs to infer the knowledge in the user's question and identify the related product.

Multi-products seller. The language model needs to find the store that sells all the products described by the user.

Coupon & Budget . The language agent needs to understand the voucher rules and find optimal product combinations within a budget.

Shopping Sandbox Implements

To ensure more consistent and reliable evaluations, we offer a large-scale shopping sandbox, serving as a simulated interactive environment that incorporates over 2.5 million real-world products sourced from Lazada.com. In this environment, an AI shopping agent is tasked with recommending suitable products tailored to the user's real-world intents by leveraging a variety of tools. To support the API tool, we build two search engines: one for the product database and another for web-based knowledge.

Search engine. We use Pyserini(Lin et al. 2021) to build a product search engine, utilizing the BM25 sparse retrieval model to construct the index offline.

Web Search engine. We encapsulated a web search tool using Serper[1], enabling access to online searches.

Intent-Grounded Instruction Generation

In this subsection, we present the framework for generating intent-grounded user instructions, which encompasses three key stages.

Stage I: Sampling Real-World Products We begin by sampling a diverse set of real-world products from our shopping sandbox, ensuring coverage across a wide range of constraints such as variations in categories, brands, attributes, and service. The sampling distribution of the product can be seen in Appendix A. Notably, for the Knowledge intent, SimpleQA (Wei et al. 2024) is used to link products, ensuring the verifiability of the responses. For the Coupon & Budget intent, we also synthesize multiple voucher rules and sample products that meet these rule requirements.

Stage II: Extracting Product Fields In the second stage, specific fields are extracted from the sampled products. These fields include detailed information such as product titles, attributes, associated services, and other relevant meta data. This structured information forms the basis for developing realistic user scenarios.

Stage III: Simulating User Queries Using the extracted product fields, we employ GPT-4.1 to simulate diverse and realistic user queries. These queries are carefully tailored to align with each purchase intent, which can be seen in Figure 2. Each simulated user instruction is intent-specific and grounded in real-world scenarios, aiming to evaluate the model's capability to understand and navigate the constraints inherent in e-commerce tasks.

[1]https://serper.dev/

Interaction Tools

We provide a set of API tools to interact with our shopping sandbox. As shown in the Figure 3, we design six API tools: retrieving product lists, viewing product details, calculating discounts and budgets, retrieving web knowledge, recommending products, and terminating states. Each invocation of a tool is formalized as an action a_t, specifically represented as tool_name$(para)$, where tool_name denotes the name of the tool being called, and $para$ represents the parameters passed to the tool. Upon invocation, the tool returns an observation o_{t+1}, which is the result or output of tool execution.

Evaluation

Given a user instruction, the model outputs its reasoning process as well as an action in the form of a tool call in each step(Yao et al. 2022b). Based on the predicted tool calling, we parse the list of invoked tools and execute the corresponding functions, returning the observation. Then the agent generates the next round of reasoning and action predictions based on the current observation and the user instruction. This process is repeated until a terminal tool is called to end the current trajectory. In the terminal state, we compare the predicted products with the target products to automatically determine whether the task has been successfully completed.

Constrain Scores

We design the Absolute Success Rate (ASR) and Cumulative Average of the product Relevance (CAR) as metrics, which calculated from the following constrain scores.

Product Relevance Score. To measure the relevance between the predicted product and the target product, we consider three dimensions: title similarity, price similarity, and product feature similarity. The formulation can be seen as follows:

$$r_{\text{pro}} = \frac{\mathbb{I}_{sim \geq 0.5} + \mathbb{I}_{min \leq p \leq max} + |F_{\text{t}} \cap F_{\text{p}}|}{2 + |F_{\text{t}}|}, \quad (1)$$

where $\mathbb{I}_{sim \geq 0.5}$ indicates that when the title similarity between the predicted product and the target product exceeds the threshold (set to 0.5), the value is 1. $\mathbb{I}_{min \leq p \leq max}$indicates that when the price p is within the target product price range $[min, max]$, the value is 1. $|F_{\text{t}} \cap F_{\text{p}}|$ is the number of overlapping features, and $|F_{\text{t}}|$ is the total number of features in the target product.

Knowledge Constrain Score. For the Knowledge intent, we evaluate whether the predicted products have the correct knowledge attribute, as follows:

$$r_{\text{kw}} = \begin{cases} 1, & \text{if knowledge\_attribute in title,} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Shop Constrain Score. For Multi-products seller intents, to assess whether the predicted products satisfy the user's request that all products come from the same shop, we define the shop relevance score as follows:

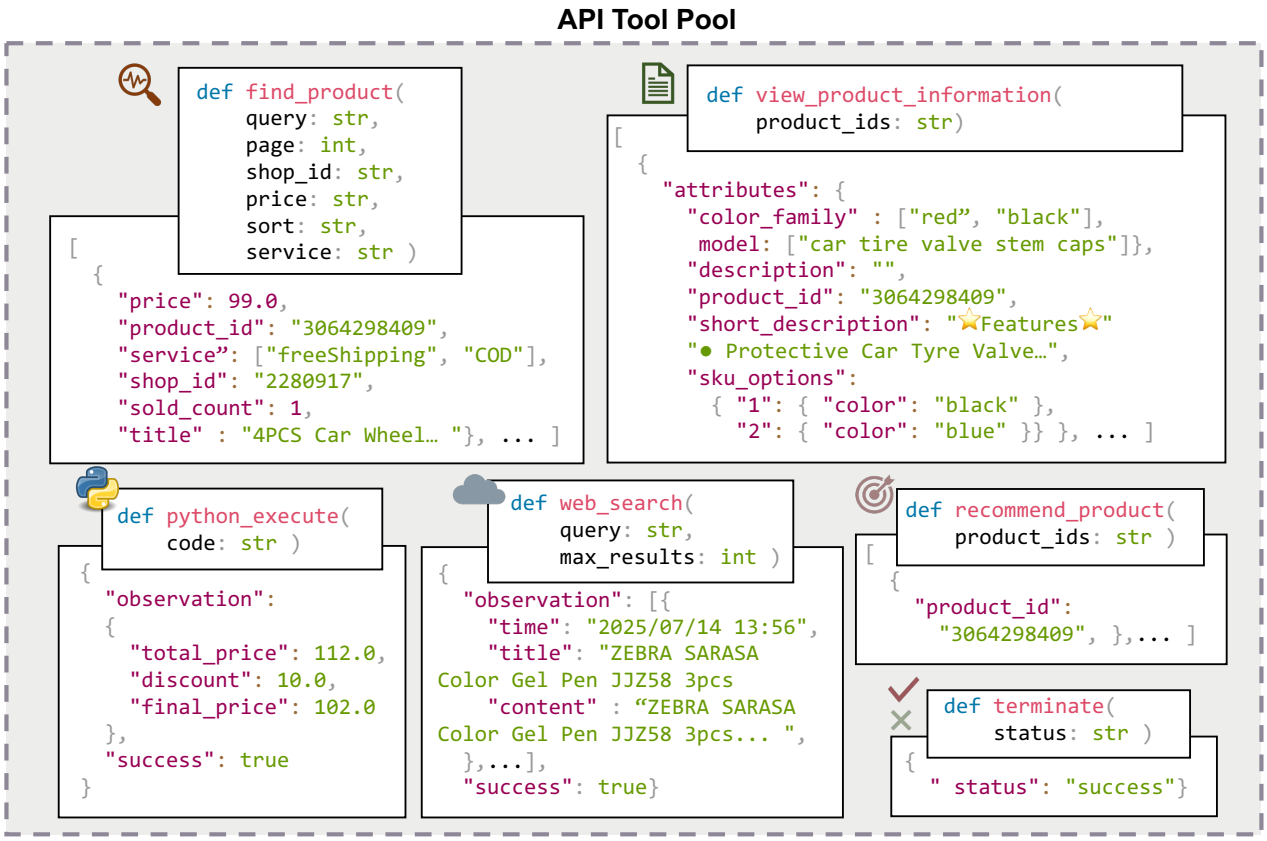


Figure 3: ShoppingBench provides six API tools designed to facilitate agent interaction with our shopping sandbox environment.

$$r_{\text{shop}} = \begin{cases} 1, & n_{\text{t}} = n_{\text{p}} \text{ and } |S_{\text{rec}}| = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The score is 1 when the number of predicted products n_p is equal to the number of target products n_t and all predicted products come from the same store ($|S_{\text{rec}}| = 1$).

Budget Constrain Score. For Coupon & Budget intent, to evaluate whether the predicted products meet the user's budget , we define the budget score as follows:

$$r_{\text{budget}} = \begin{cases} 1, & \text{if total\_price} \leq \text{budget}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Overall Metrics

In this subsection, we detailed introduce our metrics: the Absolute Success Rate (ASR) and the Cumulative Average of the product Relevance (CAR).

Cumulative Average of the Product Relevance For each intention, we compute the Cumulative Average of the product Relevance (CAR) between the predicted and target products, defined as:

$$A_{\text{pro}} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} r_{\text{pro}}^{(j)}, \tag{5}$$

where n indicates the number of samples. $\frac{1}{n_i} \sum_{j=1}^{n_i} r_{\text{pro}}^{(j)}$ represents the average product relevance within the i-th sample. n_i indicates the number of products in i-th sample.

Absolute Success Rate We also further design the following metrics to measure the absolute success rate (ASR) for each intent.

Products Finder: For intents where the user wants to locate a particular product according to its attributes, we use the product relevance score r_{pro} (Equation 1) to determine task success.

$$S_{\text{pro}} = \frac{1}{n} \sum_{i=1}^{n} \delta(r_{\text{pro}}^{(i)} = 1), \tag{6}$$

where, the indicator function $\delta(\cdot)$ is defined as:

$$\delta(r_{\text{pro}}^{(i)} = 1) = \begin{cases} 1, & \text{if } r_{\text{pro}}^{(i)} = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Knowledge: For intents where user instructions require knowledge reasoning, we include the knowledge constraint score r_{kw} (Equation 2).

$$S_{\text{kw}} = \frac{1}{n} \sum_{i=1}^{n} \delta(r_{\text{pro}}^{(i)} = 1, r_{\text{kw}}^{(i)} = 1). \tag{8}$$

Models	Products Finder		Knowledge		Multi-products seller		Coupon & Budget		Avg.
	ASR(%)	CAR(%)	ASR(%)	CAR(%)	ASR(%)	CAR(%)	ASR(%)	CAR(%)	
Closed-Source Large Language Models									
GPT-4.1	59.6	83.6	**62.0**	**67.3**	46.4	79.2	30.4	72.8	48.2
o3-mini	42.0	62.6	51.3	57.3	36.8	50.3	31.6	61.4	39.2
GPT-4o	52.4	71.5	50.0	58.7	24.0	52.4	25.2	65.6	36.6
GPT-4o-mini	33.2	46.9	28.0	31.3	10.4	52.7	11.6	54.6	20.0
Gemini-2.5-Flash	49.2	71.3	39.3	46.7	32.0	40.9	22.8	55.0	35.4
Claude-4-Sonnet	48.0	73.1	51.3	62.7	37.6	59.5	24.0	72.2	39.0
Qwen2.5-max	58.4	81.0	42.7	50.7	22.8	67.3	22.4	65.8	35.9
Open-Source Large Language Models									
DeepSeek-R1	53.2	75.8	44.0	53.3	37.2	51.5	24.4	43.9	39.2
DeepSeek-V3	54.8	75.8	48.0	54.7	22.8	46.9	21.2	55.3	35.4
Qwen3-235B-A22B	49.2	77.2	40.0	46.7	28.8	56.3	14.4	55.3	32.3
Qwen3-32B	51.6	77.6	45.3	54.0	25.6	54.2	18.0	63.8	34.0
Qwen3-14B	46.0	70.4	30.0	37.3	19.2	53.1	12.4	58.1	26.6
Qwen3-8B	40.0	65.8	22.7	27.3	13.6	31.7	11.2	53.2	21.8
Qwen3-4B	36.4	66.4	18.7	26.0	8.8	29.8	8.4	45.5	18.0
Gemma-3-27B	32.0	48.5	46.7	57.3	18.0	65.4	17.2	62.5	26.5
Gemma-3-12B	27.2	42.5	32.0	36.7	9.6	51.7	13.6	55.7	19.3
Gemma-3-4B	24.4	40.1	16.7	20.7	0	31.1	4.8	30.7	10.9
Ours									
SFT-Qwen3-4B	55.6	81.1	52.7	59.3	39.2	77.9	30.4	76.0	43.6
SFT+RL-Qwen3-4B	**60.8**	**86.1**	46.7	51.3	**53.2**	**85.5**	**33.2**	**79.0**	**48.7**

Table 1: Main results of different language agents on our ShoppingBench, including absolute success rate (ASR) and cumulative average of the product relevance (CAR). The average reported is domain-weighted ASR, rather than task-weighted.

Multi-products seller: For the intent where users want to find multiple products sold by the same shop, we introduce the shop constraint score r_{shop} (Equation 3).

$$S_{\text{shop}} = \frac{1}{n}\sum_{i=1}^{n}\delta(\frac{1}{n_i}\sum_{j=1}^{n_i} r_{\text{pro}}^{(j)} = 1, r_{\text{shop}}^{(i)} = 1). \quad (9)$$

Coupon & Budget : For the intent with budget requirements, we introduce the budget constraint score r_{budget} (Equation 4). The formulation is defined as follows:

$$S_{\text{budget}} = \frac{1}{n}\sum_{i=1}^{n}\delta(\frac{1}{n_i}\sum_{j=1}^{n_i} r_{\text{pro}}^{(j)} = 1, r_{\text{budget}}^{(i)} = 1). \quad (10)$$

Shopping Agent Training

We utilize synthetic trajectories to train Qwen3-4B backbone using Supervised Fine-Tuning (SFT) and tool-calling based Reinforced Learning (RL).

Trajectory Distillation. We leverage GPT-4.1 to generate tool-calling trajectories from 2,410 user instructions. To ensure high data quality, we apply rejection sampling based on our evaluation metrics, retaining only trajectories that achieve absolute success. Specifically, any trajectory with a final success score strictly less than 1 is filtered out. While this stringent threshold discards approximately 50% of the generated data, it effectively preserves a high-quality subset of fully successful examples.

Cold Start with SFT. We sample multiple steps from each trajectory. The final training dataset includes 5,552 steps. The model input includes the user instruction as well as the observation (e.g., a retrieved product list). The output consists of a reasoning trace (the process) and the next action (tool-calling). Then, we perform SFT on Qwen3-4B to enhance the model's ability to understand complex instructions, process multi-round observations, and predict actions.

Reinforced Tool Calling. To further enhance the model's tool-calling capabilities, we apply GRPO (Shao et al. 2024) with the tool reward(Qian et al. 2025) to continue training the SFT-Qwen3-4B model. The overall reward function combines a format reward and a tool-matching reward.

The tool-matching reward is defined as:

$$R_{\text{mat}} = r_n + r_k + r_v \in [0, S_{\max}], \quad (11)$$

where r_n denotes the tool name match rate, r_k is the parameter name match rate, and r_v is the parameter value match rate. Here, $S_{\max}$ represents the max score for R_{mat}. This raw score is then normalized to the interval $[-3, 3]$:

$$R_{\text{tool}} = 6 \cdot \frac{R_{\text{mat}}}{S_{\max}} - 3. \quad (12)$$

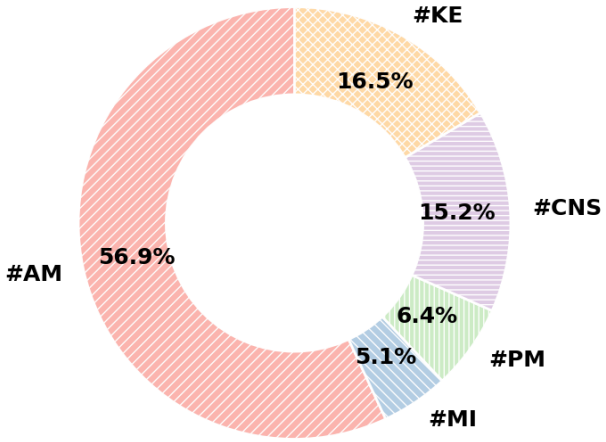


Figure 4: Breakdown of failed GPT-4.1 agent trajectories in ShoppingBench, categorized as attribute mismatch (#AM), metric issue (#MI), product missing (#PM), constraint not satisfied (#CNS), and knowledge error (#KE).

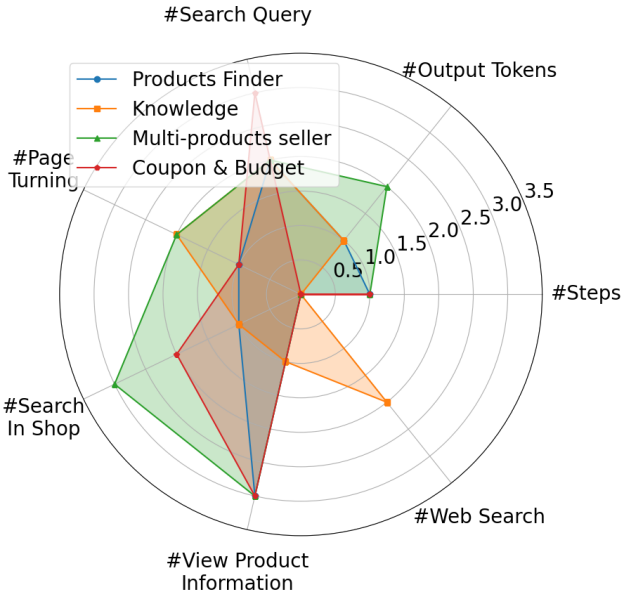


Figure 5: Correlation analysis between various factors and absolute success rates across different intents. Detailed can be seen in Appendix F.

The format reward R_{format} evaluates structural correctness. It equals 1 if the output strictly adheres to the required JSON schema (validated via regex), and 0 otherwise.

The final reward combines both components:

$$R_{\text{final}} = R_{\text{tool}} + R_{\text{format}} \in [-3, 4]. \quad (13)$$

Experiments

Baselines. We evaluate 17 language agents on our ShoppingBench, including leading closed-source models (e.g., GPT-4.1, Claude-4-Sonnet, Qwen2.5-max) and open-source models (e.g., DeepSeek-R1, Qwen3-32B, Gemma-3-27B). Additionally, we further train Qwen3-4B with synthetic trajectories using supervised fine-tuning and reinforcement learning. The training details can be seen in Appendix B.

Dataset. The ShoppingBench dataset consists of 3,310 user instructions in total, with 2,410 used for training and 900 for testing. The test set includes 150 samples for the Knowledge intent and 250 samples for each of the other intents. Detailed data statistics can be seen in Appendix C.

Main Results

As shown in Table 1, we evaluated various language agents. The experimental results lead to the following conclusions:

- **Overall Performance:** On untrained language agents, GPT-4.1 achieves the highest overall performance, with an Absolute Success Rate (ASR) of 48.2%. Among open-source models, DeepSeek-R1 achieves the strongest overall performance, surpassing GPT-4o in average benchmarks. For simple intents such as product finding, GPT-4.1 reaches 59.6% ASR, with cumulative average of product relevance (CAR) up to 83.6%. However, the performance of GPT-4.1 drops significantly on complex tasks such as the Coupon & Budget intent, falling to 30.4% ASR. These results highlight substantial room for improvement in handling complex, real-world e-commerce intents.
- **Effect of Synthetic Trajectories:** Motivated by above observation, we synthesize trajectories using GPT-4.1, generating training data via rejection sampling, and employing fine-tuning in conjunction with the GRPO algorithm to train the Qwen3-4B model, enabling it to learn tool-use capabilities. Experimental results indicate that our enhanced model achieved a remarkable improvement, with a 30.7% higher success rate compared to the original Qwen3-4B, even surpasses the performance of GPT-4.1 agent by 0.5% ASR. These findings highlight the effectiveness of our trajectory distillation and training strategy.

Further Analysis

We further analyze the reasons behind the challenges presented by our benchmark, potential areas for improvement, and the rationale of the tool settings.

Failure Breakdown We sampled 60 failed trajectories from the GPT-4.1 agent and manually analyzed the causes of their failure. We identified five distinct types of errors, which are categorized and visualized in Figure 4. The largest proportion of failures is due to missing or mismatched product attributes. This is also evident in Table 1, where the cumulative product relevance is much higher than the absolute success rate, indicating that many failures are caused by partial mismatches in product attributes. Detailed case study can be seen in Appendix D.

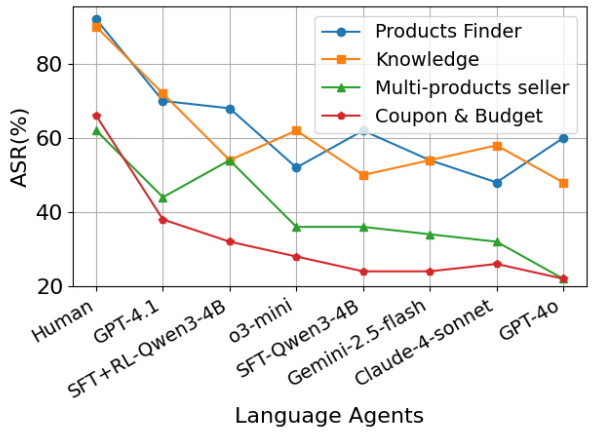


Figure 6: Comparison between humans and Agents.

Models	Knowledge Intent w/o web_search tool		
	ASR(%)	CAR(%)	kw Score(%)
GPT-4.1	50.0 (↓ 12)	56.0 (↓ 11.3)	52.7 (↓ 13.3)
o3-mini	32.0 (↓ 19.3)	37.3 (↓ 20)	34.7 (↓ 19.3)
GPT-4o	39.3 (↓ 10.7)	46.7 (↓ 12)	42.0 (↓ 8)
Gemini	19.3 (↓ 20)	23.3 (↓ 23.4)	20.7 (↓ 22.6)

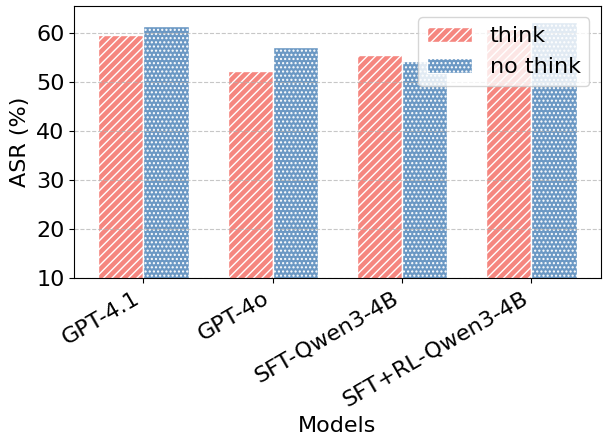
Table 2: Ablation of *web_search* Tool for Knowledge Intent.

Furthermore, we conducted a correlation analysis between different factors and success rates under various intents. We used the Pearson correlation coefficient to quantify these relationships (Figure 5). The analysis revealed that viewing product details is strongly correlated with accuracy across all intents, while the frequency of web_search tool usage is highly correlated with success in the Knowledge intent.

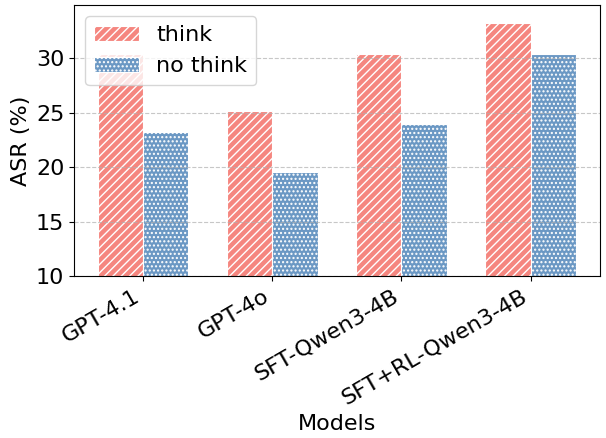
Compared with human performance. We sampled 200 tasks and invite three professional and well-educated individuals to complete each task. As shown in Figure 6, we draw two conclusions: (1) Even human participants have space for improvement on the more challenging intents, highlighting the challenging of our benchmark. (2) Even for state-of-the-art LLMs, there remains a noticeable performance gap compared to human performance.

Effect of Thinking. We explore the effectiveness of incorporating <think> process before action. As shown in Figure 7, we find that for simple intents like searching for a single product, the agents without <think> outperforms the think-based agents. However, for complex intents involving coupons or budget constraints, reasoning enables the language agent to find product combinations that better meet user needs.

Effect of Web Search Tool. As shown in the Table 2, after removing the web search tool, even the strong baselines exhibited varying degrees of performance degradation. This indicates two key points: (1) existing language agents have limitations regarding long-tail knowledge in the e-commerce domain; (2) the information gain provided by online access can effectively compensate for the agents' deficiencies in long-tail knowledge.



(a) Products Finder intent.



(b) Coupon & Budget intent.

Figure 7: Comparison of reasoning for Products Finder and Coupon & Budget .

Conclusion

We introduce ShoppingBench, a large-scale end-to-end benchmark for grounded shopping scenarios, featuring 3,310 diverse user instructions and a realistic sandbox environment of over 2.5 million products. Our proposed simulation framework, automatic evaluation metrics, and trajectory distillation approach set a new standard for agent evaluation. Experiments on 17 language agents and our fine-tuned Qwen3-4B agent, reveal a significant performance gap, highlighting both the challenges and future opportunities in language agent research for e-commerce tasks.

References

Chen, H.; Lv, K.; Hu, C.; Li, Y.; Yuan, Y.; He, Y.; Zhang, X.; Liu, L.; Liu, S.; Su, W.; et al. 2025. Chineseecomqa: A scalable e-commerce concept evaluation benchmark for large language models. *arXiv preprint arXiv:2502.20196*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. D. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Huang, Y.; Shi, J.; Li, Y.; Fan, C.; Zhang, S.; Liu, Y.; Zhou, P.; Wan, Y.; Gong, N.; and Sun, L. 2023. MetaTool Benchmark for Large Language Models: Deciding Whether to Use Tools and Which to Use.

Jia, M.; Liu, R.; Wang, P.; Song, Y.; Xi, Z.; Li, H.; Shen, X.; Chen, M.; Pang, J.; and He, X. 2022. E-ConvRec: a large-scale conversational recommendation dataset for E-commerce customer service. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 5787–5796.

Jin, W.; Mao, H.; Li, Z.; Jiang, H.; Luo, C.; Wen, H.; Han, H.; Lu, H.; Wang, Z.; Li, R.; Li, Z.; Cheng, M.; Goutam, R.; Zhang, H.; Subbian, K.; Wang, S.; Sun, Y.; Tang, J.; Yin, B.; and Tang, X. 2023. Amazon-M2: A Multilingual Multi-locale Shopping Session Dataset for Recommendation and Text Generation.

Jin, Y.; Li, Z.; Zhang, C.; Cao, T.; Gao, Y.; Jayarao, P. S.; Li, M.; Liu, X.; Sarkhel, R.; Tang, X.; et al. 2024. Shopping MMLU: A Massive Multi-Task Online Shopping Benchmark for Large Language Models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Lin, J.; Ma, X.; Lin, S.-C.; Yang, J.-H.; Pradeep, R.; and Nogueira, R. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*, 2356–2362.

Liu, Y.; Zhang, W.; Dong, B.; Fan, Y.; Wang, H.; Feng, F.; Chen, Y.; Zhuang, Z.; Cui, H.; Li, Y.; et al. 2023. U-need: A fine-grained dataset for user needs-centric e-commerce conversational recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 2723–2732.

Mialon, G.; Fourrier, C.; Wolf, T.; LeCun, Y.; and Scialom, T. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Qian, C.; Acikgoz, E. C.; He, Q.; Wang, H.; Chen, X.; Hakkani-Tür, D.; Tur, G.; and Ji, H. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Reddy, C.; Màrquez, L.; Valero, F.; Rao, N.; Zaragoza, H.; Bandyopadhyay, S.; Biswas, A.; Xing, A.; and Subbian, K. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551.

Shao, Z.; Wang, P.; Zhu, Q.; Xu, R.; Song, J.; Bi, X.; Zhang, H.; Zhang, M.; Li, Y.; et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Wang, W.; Cui, L.; Liu, X.; Nag, S.; Xu, W.; Luo, C.; Sarwar, S. M.; Li, Y.; Gu, H.; Liu, H.; et al. 2025. EcomScriptBench: A multi-task benchmark for e-commerce script planning via step-wise intention-driven product association. *arXiv preprint arXiv:2505.15196*.

Wei, J.; Karina, N.; Chung, H. W.; Jiao, Y. J.; Papay, S.; Glaese, A.; Schulman, J.; and Fedus, W. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

Yangning, L.; Ma, S.; Wang, X.; Shen, H.; Jiang, C.; Zheng, H.; Xie, P.; Huang, F.; and Jiang, Y. 2023. EcomGPT: Instruction-tuning Large Language Model with Chain-of-Task Tasks for E-commerce.

Yao, S.; Chen, H.; Yang, J.; and Narasimhan, K. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757.

Yao, S.; Shinn, N.; Razavi, P.; and Narasimhan, K. 2024. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. arXiv:2406.12045.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022b. ReAct: Synergizing Reasoning and Acting in Language Models.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *The Twelfth International Conference on Learning Representations*.