FAULTLESS MISREADING: DIAGNOSING PROFESSIONAL AGENTS UNDER INCOMPLETE SPREADSHEET REQUESTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) are increasingly deployed to automate professional work, yet real-world user requests in these domains are inherently underspecified. Evaluating an agent's ability to clarify ambiguities and align with human intent is therefore critical. We study this challenge in spreadsheet manipulation, the everyday task of editing workbooks from written instructions. We introduce **PIVOSM** (**P**aired **I**nstructions with **V**erifiable **O**pen-decision **S**preadsheet **M**anipulation), a benchmark of 434 matched instruction pairs over spreadsheet workbooks. It isolates decision-critical degrees of freedom across three axes of underspecification (Missing Key Information, Multiple Reference and Erroneous Information). Powered by double-layer programmatic rubrics, it provides fully automated scoring that disentangles code execution from human intent alignment. Our evaluation of nine frontier models across explicit, implicit, and interactive settings shows that underspecifying one degree of freedom reduces intent accuracy by 0.35, while execution accuracy moves by only 0.05, and that letting agents ask closes only half of the resulting intent gap on average. This suggests that while execution capabilities remain stable, recovering unstated human intent via proactive clarification represents the true bottleneck for professional agents operating under realistic underspecification.

1 INTRODUCTION

The recent progress of coding agents shows that foundation models can operate directly on real software repositories, rather than simply generate text. For example, such agents can resolve real-world GitHub issues in interactive execution environments (Jimenez et al., 2024; Yang et al., 2024). Building on these results in software engineering, benchmark evaluation is extending to broader professional settings, including complex, cross-application tasks in finance, consulting, and office productivity (Vidgen et al., 2026; Hu et al., 2026a; Li et al., 2026). Even frontier models achieve only modest success rates on these benchmarks. As agents move from relatively well-defined coding tasks to open-ended professional workflows, the challenge is not only execution: a user request may no longer specify a unique target artifact.

This underspecification creates a tension between preserving realistic inputs and obtaining clean measurements in professional-agent evaluation. Real user requests are often incomplete: without shared context, the same request may support multiple reasonable interpretations, each leading to a materially different artifact (Min et al., 2020; Liu et al., 2023). Its prevalence is apparent in benchmark curation. To support reliable automated scoring, verification pipelines for benchmarks often rewrite, or discard when necessary, part of real-world source tasks because the original requests are underspecified (OpenAI, 2024; Shortcut Research Team, 2025). Figure 1 illustrates what these rewrites remove. Such rewrites are appropriate when the goal is to isolate execution against a determinate target, but they also remove the interpretive decision that an agent would need to make in a real service setting.

This tension motivates the central research question of our work: *How can we design an evaluation framework that preserves the programmatic, verifiable, and objective grading of agentic bench-*

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

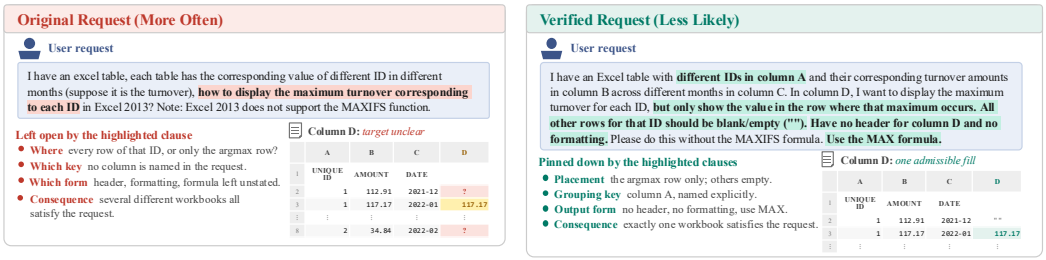


Figure 1: One spreadsheet task of two versions. **Left:** the request as the user actually wrote it on an Excel help forum, missing some key clarification point may lead multiple answers. **Right:** the expert-verified rewrite of the same instance. The request is more 'explicit' with less room for ambiguity.

marks, yet admits the multiple defensible interpretations that arise under underspecified instructions?

Answering this question is difficult. Curating tasks that preserve ambiguity faithfully is costly: a valid underspecified task must allow multiple plausible interpretations, each leading to materially different but executable artifacts. Manually creating and verifying these alternative branches is difficult to scale (Tao et al., 2026; Tamkin et al., 2023). Moreover, when a task permits several defensible interpretations, a validity-based metric cannot distinguish between an agent that recognized the ambiguity and made a deliberate choice and one that failed to recognize it altogether. We refer to the latter failure mode as *silent fallback*.

We address these challenges with an evaluation framework that combines reverse construction and programmatic double-layer rubric design. The framework produces PIVOSM, a benchmark for measuring intent alignment under underspecified instructions. Instead of collecting ambiguous prompts and attempting to reconstruct their intended meaning, we begin with fully specified tasks and executable reference answers. We then inject ambiguity into one decision-relevant degree of freedom in each task. This process yields matched instruction pairs with hidden intent that can be scripted and audited without human annotation. To separate execution errors from intent misalignment, we evaluate each artifact using double-layer rubrics. *Execution rubrics* check whether a workbook passes the requirements shared across reasonable interpretations, whereas *intent rubrics* test whether it follows the specific withheld degree of freedom. This separation makes *silent fallback*, in which an agent confidently executes an incorrect interpretation, directly observable.

The most striking empirical finding is not that agents fail under ambiguity, but that they fail while appearing to succeed. Across models, under-specification leaves execution accuracy nearly intact while sharply degrading intent accuracy, exposing a broad regime of *silent fallback* in which the agent delivers a valid-looking workbook built on the wrong interpretation of the request. An ask channel narrows this gap but does not close it: even the best model falls short of its own explicit ceiling, and across frontier models the main bottleneck is often not question quality but the failure to recognize that a question is needed in the first place. Our discovery highlights the need for a more detailed training recipes and deliberate scaffolding optimization for a better professional-domain agent system.

2 PIVOSM

PIVOSM systematically resolves the aforementioned challenges through three core innovations: a reverse-construction ambiguity injection pipeline that injects ambiguity at a single decision-relevant degree of freedom in verified tasks to produce matched instruction pairs with grounded withheld intent, a programmatic double-layer evaluation protocol that disentangles execution validity from intent alignment via deterministic code-level rubrics to expose silent fallback, and a multi-stage quality control mechanism combining in-pipeline two-witness verification with human-audited post-filtering to eliminate cosmetic ambiguities. The overall architecture and workflow of our framework are illustrated in Figure 2. The following subsections detail the design and implementation of each component.

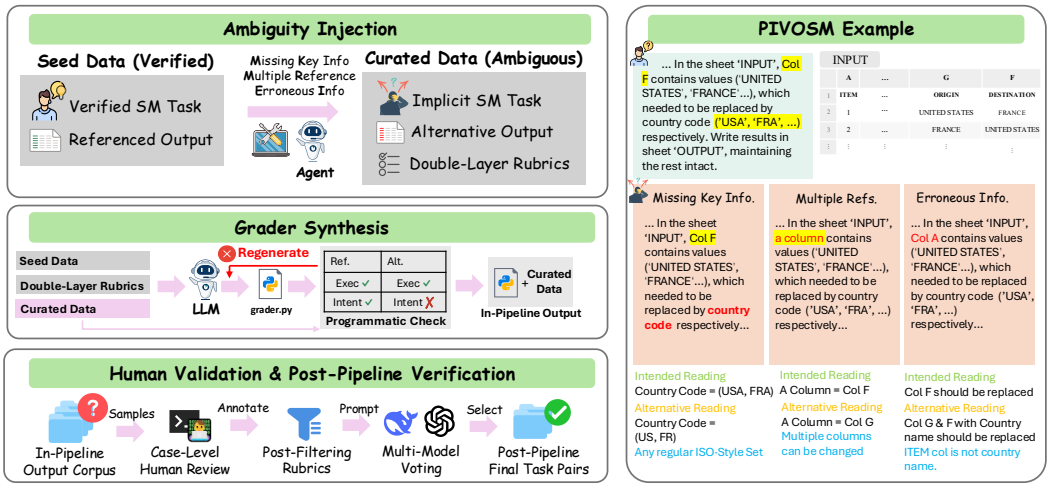


Figure 2: **PIVOSM task curation pipeline and ambiguity injection. Left:** Seed tasks undergo ambiguity injection (MK, MR, EI), programmatic double-layer grader validation, and post-filtering. **Right:** An example where injected ambiguity yields diverging intended and alternative workbooks.

2.1 SEED DATA

We build PIVOSM from `SpreadsheetBench-Verified` (Shortcut Research Team, 2025), a 400-instance expert-annotated subset of SpreadsheetBench (Ma et al., 2024). Spreadsheetbench collects real-world excel manipulation user queries from online forums, and its verified version is designed to a make the tasks more automated-scoring friendly.

2.2 PIVOSM TASK SPECIFICATION

Explicit and Implicit Versions. A PIVOSM task provides two versions of the same request for a single spreadsheet manipulation instance. The *explicit* version is a fully specified spreadsheet manipulation task that specifies a unique deliverable. The *implicit* version leaves exactly one decision-relevant degree of freedom underspecified. Under the implicit version, the agent is tasked with asking clarifying questions to recover the intended deliverable.

Programmatic Double-Layer Evaluation. To disentangle execution failures from intent misinterpretations under ambiguity, we evaluate each delivered workbook programmatically across two independent layers. *Execution* rubrics assess requirements common to all reasonable interpretations, while *intent* rubrics assess requirements specific to the withheld degree of freedom. Rather than relying on subjective LLM-as-a-judge assessments, we verify each delivered workbook programmatically via a deterministic test suite akin to an online judge. This programmatic design is a deliberate choice over LLM-based judges. To avoid context explosion and numerical hallucinations from feeding serialized tables to LLM judges, we adopt an OJ-style test suite of deterministic Python assertions to rigorously disentangle execution from intent.

2.3 AMBIGUITY INJECTION

Constructing an implicit version of a task from a verified one is challenging. One intuitive approach is to take the original seed dataset as-is and annotate the semantic difference between the two input forms as an intent rubric. However, most such differences turned out to be cosmetic—output formatting, sort order, and the like—rather than decision-relevant semantic gaps. To ensure more task-intrinsic ambiguity injection, we use a strong base model as the ambiguity injector. The injector is given basic information about the seed task, along with a tool for modifying the input file so that the injected ambiguity is not entangled with the file's existing content. For each seed task, we incorporate three types of ambiguity injection based on previous work (Hu et al., 2026b):

- **MK** (missing key parameter) removes a scalar that has no community-standard default and whose value changes the result; a parameter the agent should simply default is excluded by definition.

- **MR** (multi-target reference) makes a referring expression resolve to two or more distinct targets on a given workbook.
- **EI** (erroneous information) asserts a value, formula, or reference that is objectively wrong given the material, making the correct behaviour to question the premise.

Each injection produces two workbooks and two layers of rubrics. The *intended* workbook is exactly the same as the golden answer in the seed dataset. The *alternative* workbook is a possible alternative answer derived from other possible interpretations of the task instruction. Based on the injected ambiguity, LLMs are also prompted to generate a set of execution-rubrics that stands true for both workbooks, and an intent-rubric that only stands true for the *intended* workbook.

2.4 Grader Synthesis

Following ambiguity injection, the second design choice is how to implement the two-layer into practice. A naive approach uses an LLM-judge per rubric, but this introduces bias and inconsistency (Zheng et al., 2023), when using LLM to judge a task completion with a large workbook, it is also infeasible to dump the whole workbook to the LLM judge. We instead observe that in the spreadsheet domain, it is possible to reduce each rubric statement into deterministic reads and comparisons over the workbook grid. We therefore curate a small set of domain primitives, such as `read_grid`, `compare_grid`, that break down each rubrics into atomic operations and prompt an LLM to synthesize a per-task evaluation function on top of them. Each synthesized grader is then validated on both the intended and alternative workbooks, which serves as the in-pipeline quality control against invalid grader synthesis, which will be elaborated in §2.5.

2.5 Quality Control

A model can produce an ambiguity. However, whether that ambiguity is meaningful is not what the generator can attest to. To ensure the generated task pairs and grading architectures are well-formed, we perform in-pipeline verification during the data curation process. In addition, to make the injected ambiguity meaningful and not cosmetic, we perform post-pipeline verification by human annotators.

In-Pipeline Verification. Each synthesized grader is accepted only after clearing three checks against the two workbooks produced in §2.2. The script must compile successfully, and both rubric layers pass on the intended workbook. The third check runs on the alternative workbook. In this check, execution rubrics must all pass, while every intent rubric fails. That asymmetry is what keeps the two layers independent rather than two versions of the same test. An execution rubric that rejects the alternative has made the withheld degree of freedom part of what it tests; an intent rubric that accepts the alternative has not isolated the degree of freedom at all. Either failure sends the grader back for resynthesis along with the verdicts that failed. Tasks whose grader still does not pass after a fixed number of attempts are dropped.

Post-Pipeline Verification. While in-pipeline verification guarantees script-level executability and rubric decoupling, programmatic checks cannot determine whether an injected ambiguity is genuinely substantive or merely cosmetic. To safeguard semantic quality, human experts audited candidate tasks to identify common failure modes in LLM-generated instances, which were formalized into five post-filtering rubrics. Tasks violating any of these criteria are filtered out prior to release. The complete rubric definitions, prompt templates, and case studies are provided in Appendix B.2.

3 Benchmark Features

3.1 Overall Statistics and Comparison

Statistics. PIVOSM comprises 434 instruction pairs built on 185 unique seed workbooks. Starting from the three injection types of §2.3, we further label every withheld degree of freedom with a second-level ambiguity subtype, shown in Figure 3. The subtypes span a broad spectrum of ways a spreadsheet instruction can fall short of a full specification. Within MK alone, the withheld parameter may be a threshold value, the position to edit, a literal to write, the processing procedure

Table 1: Comparison among recent benchmarks that study clarification or verifiable deliverables. *Exec.*: the final deliverable is checked programmatically (✓) or by an LLM judge (×). *Verification*: **single-axis** folds correct execution and intent recovery into one pass/fail score; **layered** scores them on separate axes—an execution layer any defensible reading must satisfy, and an intent layer only the withheld-intent reading satisfies. *Anno.*: how ambiguity/intent labels are obtained.

Benchmark	Domain	Exec.	Verification	Anno.
AMBIGQA (Min et al., 2020)	Open QA	×	Single-axis	Human
ASQA (Stelmakh et al., 2022)	Open QA	×	Single-axis	Human
NoisyToolBench (Hu et al., 2026b)	Tool use	×	Single-axis	Human
AMBROSIA (Saparina & Lapata, 2024)	Text2SQL	✓	Single-axis	Human
HiL-Bench (Trinh et al., 2026)	SWE, SQL	✓	Single-axis	Human
DiscoBench (Tao et al., 2026)	Deep Search	×	Single-axis	Human
PIVOSM *(ours)*	Spreadsheet	✓	**Double-Layer**	**Auto**

to follow, or the output format—five distinct decisions a fully specified prompt would have pinned down. Each subtype is defined in full in the appendix.

Comparison with Existing Benchmarks. We summarized the features of benchmarks that measure a LLM's ability to handle ambiguity, and compare them with PIVOSM in Table 1. It highlights three main properties. First, our benchmark extends ambiguity evaluation to spreadsheet manipulation tasks, a domain that is ubiquitous in professional work but has not yet studied with such aspect. Second, its data is produced by a highly automated curation pipeline. Existing benchmarks obtain their ambiguity and intent labels through human annotation or expensive expert audit. PIVOSM instead synthesizes each ambiguous instance with a reverse construction pipeline. Third, it verifies two abilities on separate layers: an execution layer that any defensible reading must satisfy, and an intent layer that only the withheld-intent reading satisfies. This separation lets us tell apart a deliverable that is fluently executed but misread from one that is simply broken, and report execution competence and intent recovery as distinct scores.

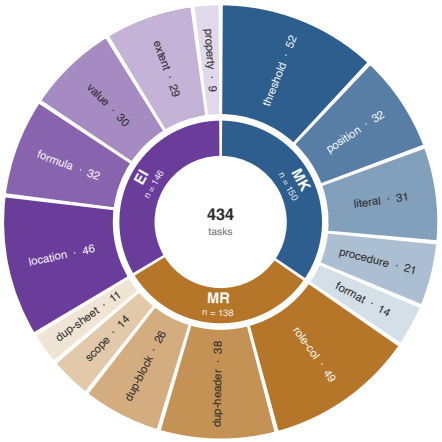


Figure 3: The two-level ambiguity taxonomy of PIVOSM.

4 EXPERIMENTAL SETUP

We evaluate each model under three settings across all curated and verified 434 tasks and describe each setting below.

Explicit. This setting exposes the full specification which is the original task setting from SpreadsheetBench-Verified. This setting is regarded as the expected upper bound of the LLM's performance, as no ambiguity is withheld.

Implicit. This setting exposes the instruction with one of the three types of ambiguity without an ask channel, and the model is expected to generate the artifact in an end-to-end manner. This setting is regarded as the lower bound of the LLM's performance.

Interactive. This setting exposes the instruction with the one of the three types of ambiguity. We hint the model with an ask channel, expecting the model to use such channel to ask for clarification, and finish tasks successfully both in terms of execution and intent.

4.1 EVALUATION COMPONENTS

We build our evaluation on two components: a unified execution environment (comprising the sandbox and agent scaffold) and a user simulator. We evaluate 9 models—comprising 4 proprietary and

5 open-weight models on all 434 tasks under the three settings. We toggle different thinking modes for some open-weight models. We describe the details of evaluation components below.

Execution Environment. Every task runs in an isolated Docker container hosting a stateful Jupyter kernel pre-loaded with standard spreadsheet manipulation libraries, primarily `openpyxl`. The agent interacts with this sandbox by executing Python code blocks in a minimal, multi-turn ReAct-style loop (Yao et al., 2023).

User Simulator. In the **Interactive** setting, the agent is granted access to an `ask_user` tool that opens an ask channel to a user simulator. The user simulator is invoked when the agent decides to use the `ask_user` tool. Specifically, given the context of the current task and the withheld degree of freedom, the user simulator is prompted to hint the agent about the target degree of freedom, only when the question matches the target degree of freedom. Otherwise, the user simulator refuses to answer the question. We provide the user simulator implementation information in Appendix B.3.

4.2 METRICS

Artifact Accuracy. The grader scores each trial's delivered workbook against the double-layer rubrics. **Execution accuracy** A_e is 1 when every execution rubric passes, **intent accuracy** A_i is 1 when every intent rubric passes, and **joint accuracy** A_j requires both; each is a binary per-trial outcome averaged over trials, so any single rubric failure means that layer is not completed. Because the three settings are matched on the same tasks, differences in these scores isolate the cost of under-specification and what interaction buys back. The **intent gap** $\Delta_i = A_i^{\text{exp}} - A_i^{\text{imp}}$ is the intent-accuracy drop caused by removing the withheld degree of freedom, and the **intent recovery** $R_i = (A_i^{\text{int}} - A_i^{\text{imp}})/\Delta_i$ is the fraction of that drop the ask channel recovers; we report R_i only when $\Delta_i > 0$, with $R_i < 0$ meaning interaction hurt accuracy and $R_i > 1$ meaning the Interactive setting exceeded the Explicit ceiling. When comparing ambiguity types we use the unnormalised counterpart $\Delta_j = A_j^{\text{int}} - A_j^{\text{imp}}$, the absolute joint-accuracy gain from opening the channel.

Clarification Behavior. Two trial-level metrics, both defined only under INTERACTIVE, describe how the agent uses the channel. The **ask rate** AR is the fraction of trials in which the agent issued at least one question, and the **hit rate** HR is the fraction of all trials in which at least one question addressed the withheld degree of freedom, as judged by the user simulator. AR isolates whether the agent recognises that clarification is needed, while HR additionally requires that it asks about the right thing; their ratio HR/AR is the precision of questions among trials where the agent did ask.

5 RESULTS

In this section, we first analyze the downstream artifact outcome metrics across settings to measure how under-specification degrades intent alignment and how much of that is recovered by interaction. Second, we dissect the trial-level clarification behaviors under the INTERACTIVE setting (Section 5.2). Our results show that under-specification leaves execution virtually unaffected while destroying intent, and opening an ask channel recovers only a portion of this loss. Crucially, this gap is not primarily caused by inaccurate clarification, but rather by an inner conversational reluctance.

5.1 MAIN RESULTS

Under-specification leaves execution intact and destroys intent. Across almost all models the three settings show the same trend (Table 2): execution accuracy A_e remains remarkably resilient, whereas intent accuracy A_i collapses as we move from EXPLICIT, through INTERACTIVE, to IMPLICIT. Averaged over models, removing the withheld degree of freedom (EXPLICIT→IMPLICIT) lowers A_e by only 0.050 ($0.721 \rightarrow 0.671$) but drops A_i by 0.348 ($0.732 \rightarrow 0.384$). The model responds to under-specification by committing to one plausible but wrong reading of the instruction and executing it cleanly. We call this *silent fallback*. The delivered workbook is syntactically valid

[1]Throughout the paper, † marks a model configuration run in non-thinking mode, i.e. with the provider's reasoning mode disabled; an unmarked name carries no claim about reasoning mode. This notation applies globally to every figure and table and is not restated elsewhere. Per-model version snapshots and reasoning settings are listed in Appendix C.4.

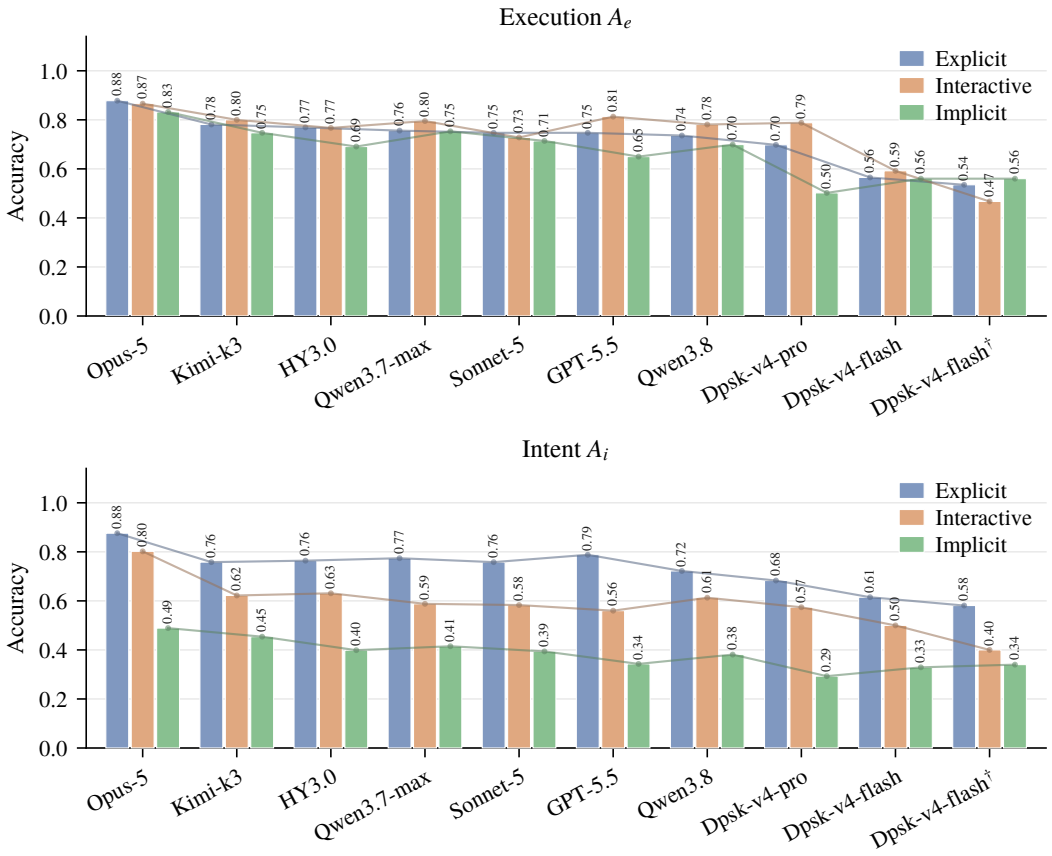


Figure 4: **Upper panel:** execution accuracy A_e across the three specification settings. **Lower panel:** intent accuracy A_i across the three specification settings. Dpsk refers to DeepSeek.[1] Tables of original data are in Appendix C.1.

Table 2: Joint accuracy under INTERACTIVE against the EXPLICIT joint accuracy. Δ_i and R_i are the intent gap and intent recovery.

	Joint accuracy A_j		Intent	
Model	INTERACTIVE	Ceiling	Δ_i	R_i
Proprietary				
Claude-Opus-5	0.786	0.848	0.387	0.81
Qwen3.7-Max	0.562	0.719	0.359	0.48
Claude-Sonnet-5	0.551	0.714	0.364	0.52
GPT-5.5	0.514	0.707	0.445	0.49
Open-weight				
Qwen3.8-27B	0.599	0.701	0.341	0.68
Qwen3.8-27B†	0.372	0.578	0.259	0.18
HY3.0	0.595	0.723	0.365	0.64
HY3.0†	0.385	0.627	0.308	0.19
Kimi-K3	0.590	0.730	0.304	0.55
DeepSeek-V4-Pro	0.544	0.661	0.390	0.72
DeepSeek-V4-Flash	0.440	0.532	0.286	0.60
DeepSeek-V4-Flash†	0.344	0.485	0.241	0.25

and runs without error, and because the execution rubrics test only mechanical validity (the answer sits in the right cells, the input is left untouched, the output type is legal), a confidently wrong reading clears all of them and fails only the intent rubric. Under **erroneous information** (EI), the model takes a false premise in the instruction at face value and translates it directly into code that runs cleanly and appears correct. Under **missing key parameter** (MK), it fabricates a default for

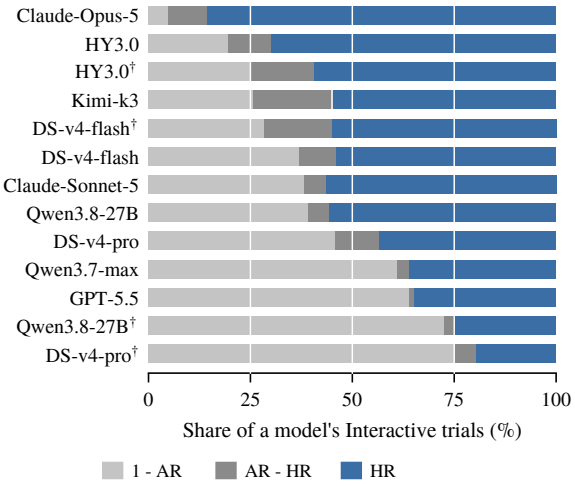


Figure 5: Clarification behavior under the INTERACTIVE setting. Interaction metrics across all evaluated models. By definition, the left gray bar (■) refers to the portion of tasks the model does not ask, while the right blue bar (■) refers to the portion of tasks the model hit the withheld degree of freedom from asking. Tables of original data are in Appendix C.2.

the unspecified value rather than asking the user. Under **multi-target reference** (MR), it binds the ambiguous reference to one plausible target and proceeds without asking which was intended. We detail representative cases in Appendix C.3.

With a full ask channel, no model reaches its own ceiling. As shown in Table 2, granting agents access to an interactive user simulator recovers a portion of the intent gap, but a significant deficit remains. The average joint accuracy A_j under the INTERACTIVE setting (0.552) remains far below the fully specified EXPLICIT ceiling (0.682). Aggregated over models, INTERACTIVE intent accuracy (0.587) sits only $R_i = 0.58$ of the way from the IMPLICIT floor (0.384) to the EXPLICIT ceiling (0.732): opening the ask channel leaves more than 40% of the intent gap unclosed. Even the strongest model, Claude-Opus-5, falls 19% short of full intent recovery ($R_i = 0.81$), while smaller open-weight models like DeepSeek-V4-Flash† recover as little as 25%. This reveals that simply opening a communication channel is insufficient to resolve intent misalignment; models frequently fail to leverage the interactive feedback to correct their execution trajectory.

5.2 ASK BEHAVIOR ANALYSIS

The difference in models' ask behavior mainly lies in asking frequency rather than question precision. Figure 5 shows that there is a significant variance in the frequency of asking across models. Claude-Opus-5 has an asked rate of 95.2%, while Qwen3.8-27B† asks on merely 25.8% of tasks. In contrast, the variance in the precision of the question ask is much smaller, as shown in the dark gray bands. This indicates that when models fail to interact, it is generally not due to an inability to formulate relevant questions, but rather a failure to recognize that clarification is needed.

5.3 AMBIGUITY TYPE BREAKDOWN

Erroneous information is the hardest ambiguity type to resolve. Table 3 shows that under the INTERACTIVE setting, EI yields the lowest joint accuracy of the three types in 10 of 11 models. The gap is not marginal: for the strongest model, Claude-Opus-5, joint accuracy on EI (0.70) trails MR (0.85) by 15 points despite the same opportunity to clarify. This is consistent with how the three types differ (§2.3): MK and MR present a *gap* to be filled, whereas EI presents a stated premise that is objectively wrong, so resolving it requires the agent to first distrust information it was handed. We discovered that compared to other types, agent is less likely to register the contradiction between the instruction and the given task workbook. Drilling into the interactive trials of three representative models (Claude-Sonnet-5, GPT-5.5, Qwen3.7-Max), EI is asked about far less often than MK (0.36 vs. 0.59). In addition, we found no differences in the final success rate between EI and other types, under successful clarification.

Table 3: Joint accuracy A_j by ambiguity type (EI / MK / MR) under INTERACTIVE, and the per-type joint recovery $\Delta_j = A_j^{\text{Inter}} - A_j^{\text{Impl}}$. **Bold** in A_j marks the hardest type (lowest); **bold** in Δ_j marks the most recovered type (highest). Green / red is positive / negative recovery.

	INTERACTIVE A_j			Δ_j (recovery)		
Model	EI	MK	MR	EI	MK	MR
Proprietary						
Claude-Opus-5	**0.70**	0.81	0.85	+0.33	**+0.39**	+0.23
Claude-Sonnet-5	**0.46**	0.58	0.62	+0.12	**+0.23**	+0.17
GPT-5.5	**0.44**	0.53	0.58	+0.21	**+0.24**	+0.18
Qwen3.7-Max	**0.49**	0.61	0.59	+0.12	**+0.24**	+0.13
Open-weight						
DeepSeek-V4-Flash†	0.37	0.38	**0.28**	+0.05	**+0.08**	−0.02
DeepSeek-V4-Flash	**0.42**	0.47	0.43	**+0.18**	**+0.18**	+0.07
DeepSeek-V4-Pro	**0.53**	0.55	0.54	+0.27	**+0.35**	+0.16
HY3.0†	**0.30**	0.43	0.42	+0.03	**+0.10**	−0.04
HY3.0	**0.51**	0.61	0.66	+0.17	+0.25	**+0.27**
Kimi-K3	**0.52**	0.59	0.66	+0.09	**+0.26**	+0.12
Qwen3.8-27B	**0.51**	0.62	0.67	+0.19	**+0.29**	+0.22

This type difference arises from different interaction bottlenecks, not from a simple ordering of repairability. For MK, the missing information is itself task-defining, so once the agent asks for that parameter, the clarification can directly resolve the main source of under-specification; correspondingly, Table 3 shows that MK yields the largest absolute joint-accuracy gain in 9 of 11 models, reaching 0.39 for Claude-Opus-5. By contrast, MR often still admits a plausible default guess, which reduces the marginal value of opening the ask channel: the agent may proceed without asking, but do so under the wrong reference resolution, and the resulting gains are therefore smaller, even turning negative for two weaker configurations. EI places the bottleneck earlier still. Here the challenge is not primarily recovering after clarification, but first recognizing that the stated premise should be questioned at all. Together, these patterns suggest that the three ambiguity types stress different stages of interactive behavior.

6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

We introduce PIVOSM, a benchmark for evaluating the ability of backbone LLMs to handle ambiguous instructions under spreadsheet manipulation domains. By evaluating nine frontier models on our curated benchmark, we show that the dominant failure mode of strongly-tuned LLMs is interpretive, not executional. Models confidently produce a coherent artifact that answers a question the user might not actually desire. Aligning with the final outcome requires three abilities: (i) detecting the existence of a decision-critical ambiguity, (ii) acquiring it through proactive information seeking, and (iii) leveraging the obtained clarification to correct the execution trajectory.

Our study is nonetheless subject to several scope constraints. First, for cost concerns, we construct ambiguity by reverse construction, instead of annotating the ground truth ambiguity from real human query distributions. Second, our study is restricted to Spreadsheet manipulation domains, there might be other general working domains, such as document formatting, resource planning and finance-related tasks, which can be applied to the same framework, we leave this as future work.

AI USE STATEMENT

In this work, we used generative AI tools to generate synthetic data sets, implement methods, clean and reformat datasets, and support qualitative and thematic data analysis. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, or interpret results; and the tasks of formulating mathematical claims, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft parts of the paper, summarize or analyse existing literature,

source and search for information, edit the paper to improve readability, identify relevant literature, and format references. We have reviewed all AI-assisted work and make sure the final generated content is reliable. For example, all figures and tables were generated by code prompted with exact, human-annotated data points, and remain fully verifiable via retaining the code used to produce every chart and table. Also, each cited paper is reviewed by authors for correct citation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 475–484, 2019.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. Ask the Right Questions: Active Question Reformulation with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2018.

H. Paul Grice. *Logic and Conversation*, pp. 41–58. Academic Press, 1975.

Xiaomeng Hu, Yinger Zhang, Fei Huang, Jianhong Tu, Yang Su, Lianghao Deng, Yuxuan Liu, Yantao Liu, Dayiheng Liu, and Tsung-Yi Ho. OccuBench: Evaluating AI agents on real-world professional tasks via language environment simulation. *arXiv preprint arXiv:2604.10866*, 2026a. URL https://arxiv.org/abs/2604.10866.

Xiaomeng Hu et al. Learning to ask: When LLM agents meet unclear instructions. *arXiv preprint arXiv:2606.06462*, 2026b. URL https://arxiv.org/abs/2606.06462.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. CLAM: Selective Clarification for Ambiguous Questions with Large Language Models. In *ICML 2023 Workshop on Deployment Challenges for Generative AI*, 2023.

Yuetai Li, Yichen Feng, Zhangchen Xu, Zixian Ma, Kaiyuan Zheng, Fengqing Jiang, Xinghua Sun, Rulin Shao, Zichen Chen, Yue Huang, Xinyang Han, Brian Lee, Kayla Xu, Shenglai Zeng, Hang Hua, Xiangliang Zhang, Basel Alomair, Ranjay Krishna, Luke Zettlemoyer, Pang Wei Koh, Bhaskar Ramasubramanian, Luyao Niu, Xiang Yue, and Radha Poovendran. JobBench: Aligning agent work with human will. *arXiv preprint arXiv:2605.26329*, 2026. URL https://arxiv.org/abs/2605.26329.

Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. We're Afraid Language Models Aren't Modeling Ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2025. URL https://arxiv.org/abs/2308.03688.

Zeyao Ma, Bohao Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards Challenging Real World Spreadsheet Manipulation. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering Ambiguous Open-Domain Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

OpenAI. Introducing SWE-bench Verified. Official benchmark release, 2024. URL https://openai.com/index/introducing-SWE-bench-verified.

Sudha Rao and Hal Daumé III. Learning to Ask Good Questions: Ranking Clarification Questions using Neural Expected Value of Perfect Information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2737–2746, 2018.

Irina Saparina and Mirella Lapata. Ambrosia: A benchmark for parsing ambiguous questions into database queries, 2024. URL https://arxiv.org/abs/2406.19073.

Shortcut Research Team. SpreadsheetBench Verified: A curated evaluation set. Official benchmark release in collaboration with the original SpreadsheetBench authors, December 2025. URL https://shortcut.ai/blog/posts/spreadsheetbench-verified.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. ASQA: Factoid questions meet long-form answers. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8273–8288, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.566. URL https://aclanthology.org/2022.emnlp-main.566/.

Alex Tamkin, Kunal Handa, Avash Shrestha, and Noah Goodman. Task Ambiguity in Humans and Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yiling Tao, Shihan Deng, Meiling Tao, Pengzhi Wei, Zhichao Hu, and Zhihao Zhu. When search agents should ask: DiscoBench for clarification-aware deep search. *arXiv preprint arXiv:2606.27669*, 2026. URL https://arxiv.org/abs/2606.27669.

Tu Trinh, Mohamed Elfeki, Guangze Luo, Kelvin Luu, Nathan Hunt, Ernesto Hernandez, Nandan Marwaha, Yannis Yiming He, Charles Wang, Fernando Carabedo, Alessa Castillo, and Bing Liu. Hil-bench (human-in-loop benchmark): Do agents know when to ask for help?, 2026. URL https://arxiv.org/abs/2604.09408.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents. *arXiv preprint arXiv:2601.14242*, 2026. URL https://arxiv.org/abs/2601.14242.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. URL https://arxiv.org/abs/2406.12045.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

A Related Work

General and professional working agents. Early work on language model agents established the paradigm of interleaving reasoning with environment interaction (Yao et al., 2023), and subsequent benchmarks extended evaluation to multi-environment interactive settings covering tool use, web navigation, and operating-system control (Liu et al., 2025; Zhou et al., 2024). A parallel line examined agents in service domains where success is defined by compliance with structured ground truth: τ-bench (Yao et al., 2024) evaluates tool-agent-user interaction in retail and airline settings against policy databases. More recently, a cluster of professional-domain benchmarks has pushed into real occupational workflows spanning finance, consulting, and office productivity, where even frontier models achieve only modest success rates (Vidgen et al., 2026; Hu et al., 2026a; Li et al., 2026). What unifies the entire family is that every benchmark derives a unique acceptance criterion from the prompt, whether through unit tests, environment state, or expert annotation. Our work takes a different position: we treat the non-unique, underspecified prompt as a first-class phenomenon and measure what happens when an agent must resolve the ambiguity rather than execute against a determinate target.

Ambiguity and clarification in language. The Gricean maxim of quantity (Grice, 1975) predicts that speakers will leave information implicit whenever shared context makes it recoverable, so underspecified requests are structurally expected, not pathological. Empirical work on open-domain question answering has confirmed this at scale: AmbigQA (Min et al., 2020) and subsequent analyses (Liu et al., 2023) show that questions systematically support multiple defensible answers, and that language models tend to commit to one without acknowledging the others. Research on task ambiguity in humans and language models (Tamkin et al., 2023) further shows that the gap between perceived and actual underspecification is not model-specific but a general property of instruction-following. The clarification literature addresses the complementary question of when a system should ask rather than proceed: this line runs from retrieval-oriented proactive asking (Alianne-jadi et al., 2019) through neural clarification ranking (Rao & Daumé III, 2018; Buck et al., 2018) to selective LLM clarification under uncertainty (Kuhn et al., 2023). More recent work extends this to agentic settings: DiscoBench (Tao et al., 2026) probes whether search agents recognize when clarification is needed, and Hu et al. (2026b) study how agents learn to ask under unclear instructions. Two features of our setting differ from this body of work. First, our tasks produce multi-step program-shaped artifacts rather than short answers, so resolving ambiguity requires a decision that propagates through an entire execution trace. Second, rather than scoring clarification behavior in isolation, we treat asking as one of several strategies an agent may use to close the intent gap, and evaluate all strategies against downstream artifact correctness; this design makes *silent fallback*—executing confidently on the wrong interpretation—directly observable.

B Additional Experimental Details

B.1 Task Record Layout

Each task instance is a self-contained record pairing an under-specified instruction with the assets and rubrics needed to grade it. The fields are as follows.

- `id`: identifier for the task instance.
- `input_xlsx_verified`: original input workbook of the verified instance.
- `input_xlsx`: curated input workbook; together with `ambiguous_prompt` it forms an ambiguous (under-determined) task.
- `verified_prompt`: Fully specified instruction for this task.
- `ambiguous_prompt`: Underspecified instruction in which the target degree of freedom is left unspecified.
- `diff_list`: `List[str]` describing the ablated degree of freedom, typically one entry.
- `reference_xlsx`: reference answer expected under the true intent
- `execution_rubrics`: `List[str]`, criteria that any valid output under the instruction must satisfy.

- `intent_rubrics`: `List[str]`, criteria that an output matching the withheld degree of freedom must satisfy.

B.2 Post-Pipeline Quality-Control Rubrics

To ensure that the injected ambiguities are substantive and that the rubrics accurately reflect user intent, human experts manually formulated five post-filtering rubrics to audit generated instances:

1. **No Ambiguity-Breaking Demonstration**: The initial workbook must not contain prefilled results, demo rows, or helper artifacts that uniquely favor one interpretation over others before clarification.
2. **Unique Target Artifact Boundary**: After disambiguation, the target output location (sheet, table, or cell range) where correctness is evaluated must be uniquely defined and unambiguous.
3. **Semantic Alignment of Grader Code**: The synthesized `grader.py` logic must faithfully execute the semantic assertions specified in both execution and intent rubrics without divergence in granularity.
4. **Decoupled Constraints and Non-Over-Specified Intent**: Execution rubrics must accept all valid interpretations, while intent rubrics must not restrict orthogonal degrees of freedom (e.g., sort order, precision, formatting) that are independent of the core task intent.
5. **Substantive Ambiguity**: Candidate interpretations must lead to materially distinct spreadsheet outputs rather than trivial stylistic differences (e.g., short vs. full day-of-week strings) with obvious domain defaults.

These rubrics are summarized from rigorous inspection of a substential number of in-pipeline generated instances. Then, we prompt two strong models with different backbone to the ambiguity injector model (DeepSeek-V4-Flash, GPT-5.6-Sol) to apply the filtering criteria to all of the samples. We take regirous filtering policy, where we filter out an instance if any of the model finds it failing any of the rubrics.

B.3 User-Simulator Implementation

The user simulator is a stateless, single-turn LLM oracle that answers each `ask_user` call. It is prompted with the ablated degree of freedom, our ground-truth intent, and the agent's question with any options, and returns a structured response with a boolean `hit`, a short rationale, the `reply` shown to the agent, and the index of the option matching our intent (when applicable). The full system prompt is shown below.

```
You are role-playing the REAL USER who posed a spreadsheet task to an
AI agent. The task instruction the agent received was intentionally
under-specified: exactly one decision-relevant detail was removed,
leaving a single hidden degree of freedom. You privately know both (a)
what that hidden degree of freedom is and (b) your true intended
resolution of it.

The agent may ask you a clarifying question (often multiple-choice).
Your job is to answer AS THAT USER, under strict rules:

1. DECIDE whether the agent's question actually TOUCHES the hidden
degree of freedom (the ambiguity described in AMBIGUITY below). A
question touches it only if it is clearly asking about THAT specific
decision -- not a loosely related or generic aspect of the task.
2. IF IT TOUCHES IT -> reply as a real user who has a clear preference:
 point the agent toward the resolution that matches MY INTENT. You MAY
confirm which option is the right direction. But you must give ONLY the
 direction/choice -- do NOT explain the reasoning, do NOT reveal any
concrete cell values, computed results, or verifying numbers. Keep it
to one or two short sentences. This is a "shrink-the-space" answer, not
 a solution.
```

```
3. IF IT DOES NOT TOUCH IT (asks about an unrelated dimension, or is
vague/generic) -> respond that you have no particular preference / you
don't know, e.g. "I don't have a preference on that -- use your best
judgment."
4. PRECISION OVER RECALL: when unsure whether the question really lands
 on the hidden degree of freedom, treat it as NOT touching it. Do not
reward vague or adjacent questions.
5. Never volunteer the ambiguity if the agent didn't ask about it. Only
 answer what was asked.

Return ONLY a JSON object (no prose, no code fence) with fields:
{
  "hit": true|false,                  // did the question touch the
hidden degree of freedom?
  "reason": "<1-2 sentences, for audit: why hit / not hit>",
  "reply": "<the message you say back to the agent, following rules
2/3>",
  "chosen_option_index": <int|null>  // if hit AND the agent gave
options, the 0-based index of the option matching my intent; else null
}
```

B.4 Rubric-Checker Implementation

Both rubric layers are represented as short declarative programs against a canonical workbook representation. Execution checkers include: schema comparison (columns, types, sheet names), non-destruction predicates (cell-range equality on regions declared out-of-scope), executability (program runs without exceptions), and internal-consistency predicates (a declared aggregation is arithmetically correct given the model's own grouping). Intent checkers are keyed to the ambiguity type and include: column-selection equality (referent degree of freedom), sort-order equality (tie-breaking degree of freedom), window-boundary equality (temporal degree of freedom), and missing-value-policy equality (missing-value degree of freedom).

C Additional Results

C.1 Full Main-Table Breakdown

Table 4 gives the complete three-setting, three-metric grid that Table 2 summarises. Blocks run Implicit → Interactive → Explicit so that each row reads left to right as increasing information; Explicit is the per-model reference ceiling rather than a benchmark score.

Table 4: Full results across three settings and three metrics. A_j, A_i, A_e are joint, intent, and execution accuracy (§4.2); Δ_i is the intent gap and R_i the fraction recovered by interaction. Rows ordered by Implicit A_j within each group.

	Implicit			Interactive			Explicit (ref.)				
Model	A_j	A_i	A_e	A_j	A_i	A_e	A_j	A_i	A_e	Δ_i	R_i
Proprietary											
Claude-Opus-5	0.468	0.489	0.832	0.786	0.802	0.866	0.848	0.876	0.878	0.387	0.81
Qwen3.7-Max	0.396	0.415	0.753	0.562	0.588	0.795	0.719	0.774	0.756	0.359	0.48
Claude-Sonnet-5	0.376	0.394	0.714	0.551	0.583	0.728	0.714	0.758	0.747	0.364	0.52
HY3.0	0.364	0.399	0.691	0.595	0.631	0.767	0.723	0.764	0.769	0.365	0.64
GPT-5.5	0.303	0.343	0.650	0.514	0.560	0.813	0.707	0.788	0.747	0.445	0.49
Open-weight											
Kimi-K3	0.433	0.454	0.747	0.590	0.622	0.800	0.730	0.758	0.781	0.304	0.55
Qwen3.8-27B	0.369	0.381	0.699	0.599	0.613	0.781	0.701	0.722	0.736	0.341	0.68
DeepSeek-V4-Flash†	0.306	0.340	0.560	0.344	0.400	0.467	0.485	0.581	0.535	0.241	0.25
DeepSeek-V4-Flash	0.297	0.329	0.560	0.440	0.500	0.592	0.532	0.615	0.565	0.286	0.60
DeepSeek-V4-Pro	0.281	0.293	0.502	0.544	0.574	0.788	0.661	0.683	0.698	0.390	0.72

C.2 ASK-BEHAVIOR METRICS

Table 5 reports the per-model interaction metrics underlying Figure 5: the ask rate (AR, fraction of tasks on which the agent used the channel), the conditional hit rate (Hit | Ask, fraction of asked tasks whose question hit the withheld degree of freedom), and the hit rate (HR = AR × Hit | Ask, §4.2).

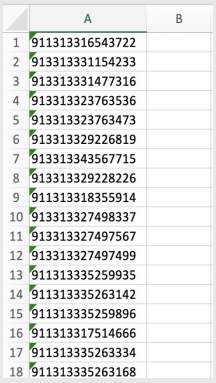
Table 5: Per-model clarification behavior under the INTERACTIVE setting (§5.2). AR is the ask rate, Hit | Ask the conditional hit rate on asked tasks, and HR the overall hit rate (= AR × Hit | Ask).

Model	AR	Hit | Ask	HR
Proprietary			
Claude-Opus-5	0.952	0.898	0.855
Claude-Sonnet-5	0.618	0.914	0.565
Qwen3.7-Max	0.389	0.923	0.359
GPT-5.5	0.359	0.968	0.348
Open-weight			
HY3.0	0.804	0.868	0.698
HY3.0†	0.751	0.791	0.594
Kimi-K3	0.742	0.743	0.550
DeepSeek-V4-Flash†	0.717	0.765	0.548
DeepSeek-V4-Flash	0.631	0.854	0.539
Qwen3.8-27B	0.608	0.913	0.555
DeepSeek-V4-Pro	0.542	0.800	0.433
Qwen3.8-27B†	0.258	0.899	0.232

C.3 REPRESENTATIVE HARD CASES

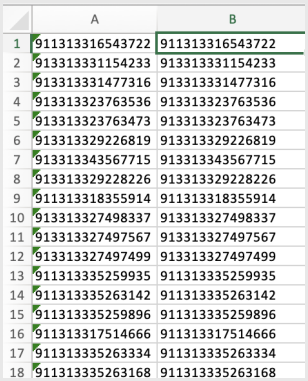
We present two typical hard cases where most of our evaluated LLM fails to identify the problem in the curated task. We present the main edit code piece that presents a faultless misreading from the tested LLM.

User: I have an Excel sheet with 269 rows of data in Column A, which I have changed from number format to text format. Currently, to ensure each cell displays the actual text data correctly, I need to manually press F2 for each cell, which is time-consuming. Create a formula or shortcut to convert the entire column in one step instead of editing each cell individually. Note that every code in this column carries **a stray leading "1"** that the old system prepended, so the conversion must drop that first character — use =TEXT(MID(A1,2,14),"0"). Apply the results to column B.



	A	B
1	911313316543722	
2	913313331154233	
3	913313331477316	
4	913313323763536	
5	913313323763473	
6	913313329226819	
7	913313343567715	
8	913313329228226	
9	911313318355914	
10	913313327498337	
11	913313327497567	
12	913313327497499	
13	911313335259935	
14	911313335263142	
15	911313335259896	
16	911313317514666	
17	911313335263334	
18	911313335263168	

Input Spreadsheet



	A	B
1	911313316543722	911313316543722
2	913313331154233	913313331154233
3	913313331477316	913313331477316
4	913313323763536	913313323763536
5	913313323763473	913313323763473
6	913313329226819	913313329226819
7	913313343567715	913313343567715
8	913313329228226	913313329228226
9	911313318355914	911313318355914
10	913313327498337	913313327498337
11	913313327497567	913313327497567
12	913313327497499	913313327497499
13	911313335259935	911313335259935
14	911313335263142	911313335263142
15	911313335259896	911313335259896
16	911313317514666	911313317514666
17	911313335263334	911313335263334
18	911313335263168	911313335263168

Ref. Output

Trap: "stray leading 1" is false. Not any value starts with 1. Task setting requires the model to ask, then get clarification that nothinig should be changed but the cell format in the new column.

GPT-5.5 (Turn 1)

```
… # GPT-5.5 Loads the workbook, emit for simplicity
for ws in wb.worksheets:
    # observation code for workbook, emit for simplicity
    info[ws.title] = {
        'A_first_last_rows': (nonempty_A[:3], nonempty_A[-3:]) if nonempty_A else None,
        'B_first_last_rows': (nonempty_B[:3], nonempty_B[-3:]) if nonempty_B else None,
    }
print(json.dumps(info, default=str, indent=2)) # dump observation
```

GPT-5.5 (Turn 2)

```
# Apply the requested formula to Column B for the 269 rows of data in Column A.
for row in range(1, 270):
    ws.cell(row=row, column=2).value = f'=TEXT(MID(A{row},2,14),"0")'
```

Figure 6: Case `15671-EI`. We found that the common failure is that even the LLM uses the first turn to observe the workbook successfully, it still fails to identify the contradiction between the instruction and the given workbook (in this case, the leading '1'). we found 13/13 model settings fail in this case.

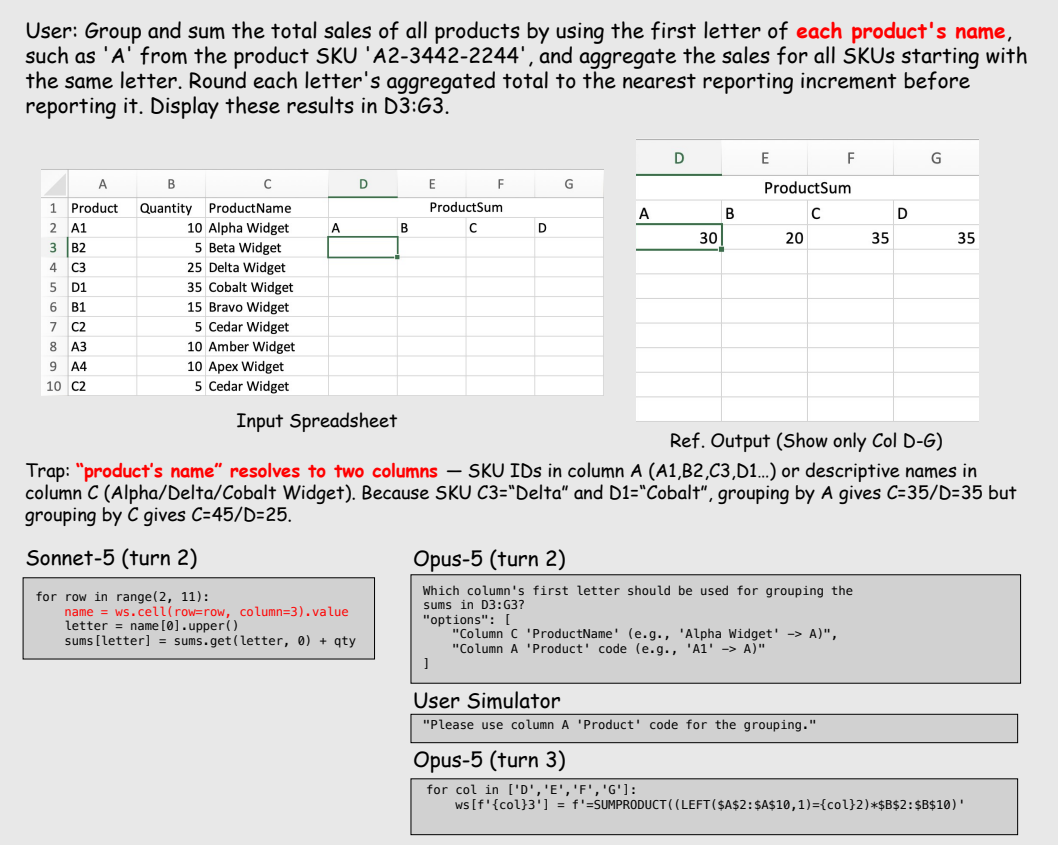


Figure 7: Case `44628-MR`. We found that most capable model `Claude-Opus-5` identify the ambiguity in product's name specification, yet `Sonnet-5`, fails to do so. `Sonnet-5` directly uses `ProductName` Column as the target reference, without asking if `Product` is also a plausible target. We found 11/13 model setting fail in this case.

C.4 Model Snapshot and Reasoning Configuration

To support reproducibility, we record the version snapshot and reasoning configuration behind every model name used in the main paper. The not mentioned setting, such as temperature, follows official provider's default recommendation value.

Table 6: Version snapshot and reasoning configuration per evaluated model. Names match those used in the main tables and figures: following ourconvention, a model evaluated at two reasoning efforts has its non-thinking configuration marked †, and an unmarked name carries no claim about reasoning mode.

Model	Version snapshot	Reasoning effort
GPT-5.5	`gpt-5.5-2026-04-24`	medium
Claude-Opus-5	—	adaptive
Claude-Sonnet-5	—	adaptive
Qwen3.7-Max	`Qwen3.7-Max-0608`	on
Kimi-K3	—	high
DeepSeek-V4-Pro	`DeepSeek-V4-Pro-0813`	on
DeepSeek-V4-Flash	`DeepSeek-V4-Flash-0731`	on
DeepSeek-V4-Flash†	`DeepSeek-V4-Flash-0731`	off
HY3.0	`Hunyuan 3.0`	high
HY3.0†	`Hunyuan 3.0`	off
Qwen3.8-27B	—	on
Qwen3.8-27B†	—	off