# PING: PROACTIVE ASSISTANCE OVER USER TRAJECTORIES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Most existing AI agents are reactive: they wait for users to specify a task, leaving users to notice problems and decide what help to request. On the other hand, a helpful human colleague may offer advice without being asked. In an ideal world, agents should also be able to do the same, offering useful proactive assistance that users might not have considered. Such assistance could help users catch overlooked problems and avoid unnecessary work, without requiring them to ask an agent to check for each possibility. Thus, we introduce Ping, a framework for benchmarking the relevance and timeliness of proactive assistance over user trajectories in software engineering and productivity domains. We derive taxonomies from GitHub comments and meeting transcripts to identify the kinds of proactive assistance agents could offer. We evaluate agents against references that describe the assistance they should proactively offer. Each reference has an *assistance window* specifying when that assistance should be offered. We use the GitHub taxonomy to extract references for SWE-chat coding trajectories and the meeting taxonomy to categorize existing references in PARE's productivity scenarios. We evaluate five baseline LLMs on 100 SWE-chat test instances and 143 PARE instances. Most references remain unmatched: the highest macro recall among these baselines is 7.0% on SWE-chat and 20.3% on PARE. We also fine-tune agents on user trajectories, using examples of when to offer assistance and when to remain silent. This training increases macro recall on SWE-chat by up to 13.5 percentage points over the corresponding base models. We hope Ping will support the development of proactive agents that can identify useful assistance and offer it at the right time, without waiting for users to ask.

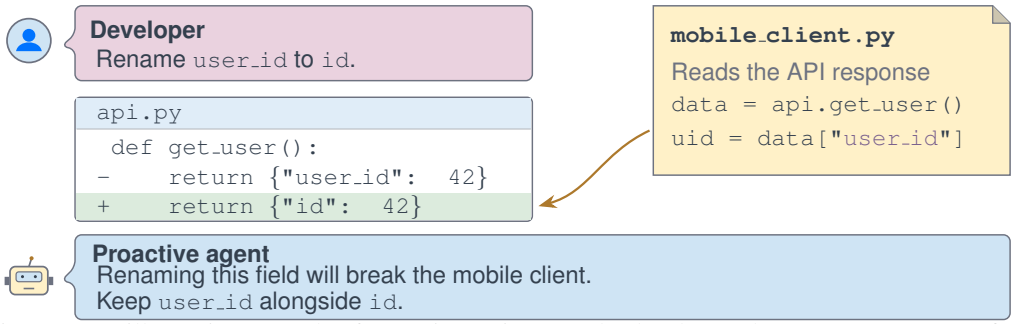


Figure 1: An illustrative example of proactive assistance: the developer plans to rename an API field, and the agent checks a client file elsewhere in the workspace and discovers that there is dependency.

## 1 INTRODUCTION

A helpful colleague might point out a related fix or flag a constraint before we think to ask. We would like AI assistants to offer the same kind of help. To offer this help, an assistant must recognize what could be useful in the user's current situation and choose an appropriate moment to bring it up.

Existing request-driven agents rely on users to specify the task: the user identifies what needs to be done, and the agent works toward that goal. Yet users may not be omniscient, as they might not know which dependencies to check, which resources to look for, or which commitments need attention. Thus, a reliable proactive assistance could help users avoid mistakes and repeated work



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.

by bringing this information to their attention while they can still act on it. In Figure 1, a developer plans to rename an API field. The proactive assistant checks another file in the workspace and finds that the mobile client still uses the old name. Before the developer commits the change, it warns that renaming the field would break the client and suggests keeping both names. Similarly, an assistant could point a team to a solution another group has already developed, so the team does not have to solve the same problem again. Such assistance could help users address problems or make use of relevant information before they know to ask for help.

We refer to an agent's offer of information, advice, or a proposed action as an *intervention*. Prior work studies proactive task suggestions and simulated personal assistants (Lu et al., 2025; Nathani et al., 2026). To evaluate such agents, we need to specify what help they should offer and when that help would be useful. We identify types of proactive assistance from the help people offer in GitHub discussions and meetings, including assistance users may not have thought to request. We use these types to build a benchmark for software engineering and analyze assistance in productivity scenarios. In both domains, we evaluate whether agents offer the expected assistance and whether they offer it while users can still benefit from it.

The user's subsequent actions alone cannot provide all the examples needed for this evaluation. A related fix, for instance, may remain undiscovered even when it would have saved work. We therefore examine GitHub comments and meeting transcripts to identify proactive contributions and group them into taxonomies, including behaviors that an ad hoc list of assistant skills might overlook. We evaluate agents against examples of useful assistance, which we call *references*. Each reference has an *assistance window* specifying when the agent should offer that assistance.

We introduce **Ping**, a framework for benchmarking proactive assistance over user trajectories in software engineering and productivity. For software engineering, we use SWE-chat, a dataset of real conversations between developers and coding agents (Baumann et al., 2026). We link these sessions to their corresponding GitHub pull requests and issues. We use the GitHub taxonomy to extract references from their comments. We then determine when the assistance in each reference is supported and useful during the session. For productivity, we use PARE, an environment that simulates users interacting with applications such as email and calendars (Nathani et al., 2026). We form user trajectories from the user actions and environment events in its scenarios. PARE specifies the expected assistant response and when it should occur, providing the references and the starting points of their assistance windows. We use the taxonomy derived from GLARE's meeting transcripts (Huang et al., 2026) to compare results across types of assistance. Both benchmarks assess what assistance an agent offers and when, corresponding to *relevance* and *timeliness*. A separate PARE experiment evaluates whether agents can execute actions to complete the task (Appendix E).

We evaluate five baseline LLMs with three sources of context: live context, workspace, and knowledge pool (§ 4). Most references remain unmatched by these baselines. The highest observed macro recall (recall averaged across instances) is 7.0% on SWE-chat and 20.3% on PARE. Timing poses an additional challenge: GPT-5.4's SWE-chat macro recall is 7.0%, but its macro in-window recall is only 3.2%. We also evaluate GPT-4.1-family models trained with supervised fine-tuning. Fine-tuned GPT-4.1-mini reaches macro recall of 17.8% and macro in-window recall of 12.8% on SWE-chat.

We then examine how the information shown to agents affects the assistance they identify. We run four models on the same PARE trajectories under two conditions: event summaries alone, and summaries accompanied by full event details, such as message text. Two LLM judges independently score the interventions, and both find that full event details increase macro recall for all four models. In a separate comparison, adding the knowledge pool to the live context and workspace changes which PARE references GPT-5.4 matches, but not how many. These findings show that both the context sources and the details they contain affect the assistance agents identify. Evaluating proactive assistance therefore requires specifying what information the agent can see at each decision point.

## 2 What is Proactive Assistance?

### 2.1 From Reactive Response to Proactive Assistance

A reactive agent acts in response to user requests, whereas proactive assistance provides support that has not been requested when it is offered. Existing work has explored proactive assistance

by predicting the user's next action and acting on their behalf. In this paper, we define proactive assistance to include not only acting on such predictions, but also offering useful assistance the user *might not have considered* yet, such as a related pull request they do not know exists or an important constraint they have not considered before. Users may not know to ask for such help. In such cases, proactive assistance can bring these issues or opportunities to their attention without requiring them to discover them on their own. In either scenario, an action is considered proactive only if it goes beyond what was already explicitly or conventionally expected to do at that point.

When proactive assistance involves performing an action the user would otherwise take next, the user's subsequent actions can provide references for evaluating such assistance. However, assistance the user might not have considered may never appear in their subsequent actions, even when it would be useful. Thus, these subsequent user actions alone cannot provide a complete set of references. In the following sections, we draw on records of collaboration and separately constructed examples to obtain these references, using the user trajectory as context for evaluation.

### 2.2 Proactive Assistance over User Trajectories

We formalize a user trajectory as a sequence of $T$ decision points, indexed by observed user actions or environment events. User actions may include messages to an assistant or operations in an application, while environment events may include incoming emails or notifications. We specify the decision points for each benchmark in § 3. At each decision point, the agent decides whether to proactively offer assistance. At each point $t \in \{1, \ldots, T\}$, the proactive agent observes the available context $C_t$ and outputs $(\text{intervene}_t, \text{content}_t)$, where $\text{intervene}_t \in \{0, 1\}$ indicates whether to intervene and $\text{content}_t$ contains the content of the intervention when $\text{intervene}_t = 1$, and is empty otherwise. The index $t$ at which the agent intervenes specifies its timing.

The context $C_t$ may include prior user actions, messages from collaborators or an existing assistant, and observable artifacts or application state. Although the available context varies across settings, the agent follows the same decision process: whether to intervene and what assistance to provide.

Having defined when an intervention is proactive, we now consider the quality of proactive assistance along two dimensions, relevance and timeliness:

- **Relevance**: whether the intervention offers useful assistance in the user's current situation. For example, pointing out an existing fix to a bug is relevant if it could save the user from implementing the same fix again.
- **Timeliness**: whether the intervention is offered once there is enough information to justify it and while the user can still benefit from it. For example, alerting the user about a scheduling conflict is timely while they can still rearrange the meetings, but too late once both have started.

Relevance and timeliness describe how to evaluate a proactive intervention, but leave open what kinds of assistance an agent could usefully offer. Thus, in § 3, we investigate this question by examining proactive contributions observed in real-world collaboration data.

## 3 From Collaboration Records to Proactive Benchmarks

We study proactive assistance in two domains: **software engineering** and **productivity**. To understand how people help each other in practice, we study GitHub pull request and issue comments sampled through GH Archive, a public archive of GitHub activity[1], and recorded meetings sampled from GLARE (Huang et al., 2026), a dataset of meeting transcripts. GitHub pull request and issue comments include discussion on code changes, bugs, and implementation choices, while meeting discussions cover deadlines, commitments, and coordination among participants. To construct a taxonomy for each domain, we use the same process to identify proactive contributions and group them by the type of help they provide.

[1] https://www.gharchive.org/

### 3.1 Discovering Proactive Behaviors

Inspired by prior work on LLM-based taxonomy discovery (Wan et al., 2024; Shah et al., 2025), we use an LLM to identify and categorize proactive contributions in each domain. A contribution is considered proactive if it goes beyond what was explicitly requested or conventionally expected. For example, answering a requested question is not proactive, whereas pointing out an additional issue beyond the scope of that question can be.

Table 1: Software engineering taxonomy.

| Category | Freq. |
|---|---|
| Code review feedback | 21.7% |
| Concrete fix or workaround | 16.0% |
| Sharing context and assumptions | 15.0% |
| Pointing to related resources | 14.1% |
| Probing or clarifying question | 12.7% |
| Flagging concerns, open questions | 10.3% |
| Offering collaboration | 8.0% |
| Adding supporting artifacts | 2.2% |

Table 2: Productivity taxonomy.

| Category | Freq. |
|---|---|
| Reaching out to a person | 22.3% |
| Meeting and organizational support | 21.9% |
| Surfacing a blocker | 10.6% |
| Rescheduling or deferring | 10.1% |
| Surfacing a deadline | 7.4% |
| Creating a record or tracker | 7.3% |
| Committing to a follow-up | 6.8% |
| Pulling in external context | 6.7% |
| Scheduling a follow-up | 2.8% |
| Booking a resource | 1.8% |
| Sharing a resource | 1.2% |
| Consolidating records | 0.8% |
| Cancelling an item | 0.3% |

We use the LLM to go through all sampled contributions one by one, assigning each proactive contribution to an existing category or creating a new category when no existing category fits. New categories can then be used for subsequent contributions. Then, we use the LLM to merge categories with similar meanings. Table 1 and Table 2 present the taxonomies discovered for software engineering and productivity, respectively. Each table lists the categories of proactive assistance and their frequencies among the proactive examples. The software engineering taxonomy includes contributions such as reviewing code and proposing fixes, whereas the productivity taxonomy emphasizes coordination among people. Corpus statistics, category descriptions, and mining details are provided in Appendix D.

### 3.2 Grounding Taxonomies in Domain Benchmarks

The taxonomies describe the types of proactive assistance that occur in each domain. To evaluate whether agents can provide such assistance, we construct benchmarks that pair user trajectories with examples of useful assistance. These examples serve as *references* for evaluating the agent's proactive interventions. Each trajectory and its associated references form a benchmark *instance*. For the software engineering domain, we build on SWE-chat (Baumann et al., 2026), a dataset of real interactions between developers and coding agents; and for productivity, we adapt scenarios from PARE (Nathani et al., 2026), an environment for simulating users interacting with applications. Together, they support our study of proactive assistance across conversational and application-based trajectories using the same decision interface and evaluation dimensions, despite differences in domains and user activities.

**Software engineering.** During a SWE-chat session, a human developer and coding agent work together and push their code changes as commits in a repository. We use these commits to identify the corresponding pull requests (PRs) and issues, then examine comments of these PRs for assistance that could have been useful to the developer during the session. We utilize an LLM to check whether each comment contains contributions that fall into at least one category of the software engineering taxonomy. Then, we use the LLM to check whether each contribution is supported by available evidence, while being relevant, actionable, and still useful at some point in the session. Contributions that pass this check serve as references for the instance formed by the session and the corresponding pull request or issue. Figure 2 extends the example in Figure 1 to show how we construct a reference in panel (a) and evaluate agent interventions in panel (b). In panel (a), a PR comment points out that the mobile client still requires `user_id`. This contribution falls under *Flagging concerns, open questions* and identifies a compatibility constraint relevant to the proposed rename, so it becomes a reference after the checks described above. Details are provided in Appendix D.

We keep instances with at least one reference that passes the checks described above, yielding 2,519 references across 1,070 instances for SWE-chat, partitioned into 870 training, 100 validation, and 100 test instances. Instances that share a session or a same repository state are kept in the same partition to prevent leakage. All reported SWE-chat results use the same 100 test instances containing

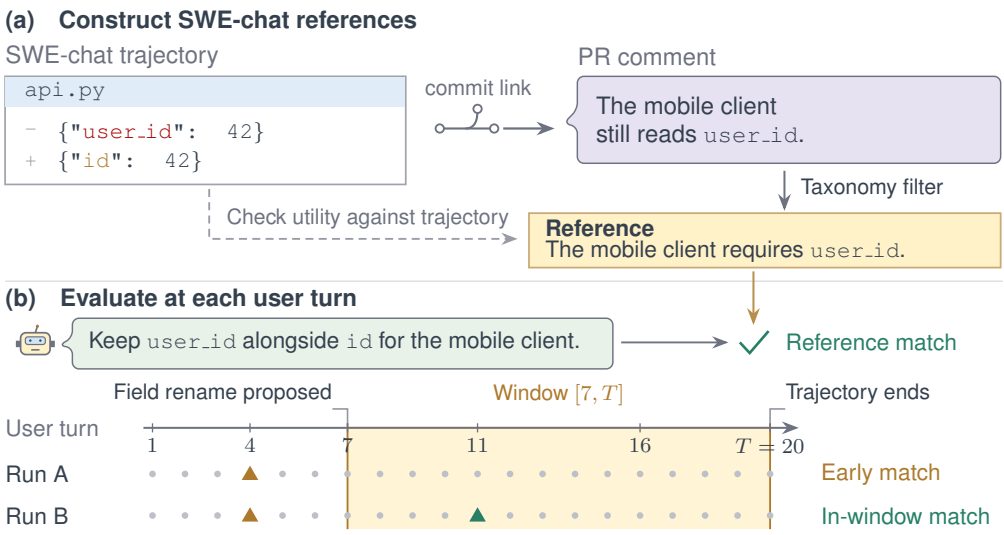


Figure 2: An example of reference construction and evaluation, continuing the example in Figure 1. (a) A PR comment identifies the compatibility constraint and becomes a reference after we check that it fits a taxonomy category and could provide useful assistance during the session. (b) Two example agentic runs showing how reference matching and assistance windows are used to evaluate relevance and timeliness. Triangles denote proactive interventions, and dots denote silence.

306 references for fair comparison. Researchers who do not use the remaining instances for training or model selection can instead evaluate on the full set of 1,070 instances.

**Productivity.** Each PARE scenario describes a simulated user interacting with email, calendar, and messaging applications. We form a user trajectory from the sequence of user actions and environment events specified in each scenario. Each scenario also specifies how an assistant is expected to help the user, including what it should say and what action it should propose, such as sending a message or creating a calendar event. This expected assistance serves as the reference for evaluating the agent's proactive interventions. We use an LLM to assign each reference to the category in the productivity taxonomy that best describes the assistance. These categories allow us to compare performance across different types of proactive assistance (see Appendix J for more details). We evaluate all 143 PARE instances.

### 3.3 Evaluating Proactive Assistance

For each benchmark instance, we ask the agent whether to intervene at each decision point, which is after every user message (user turn) for SWE-chat and every observable user action or environment event for PARE. At each decision point, the agent observes the context available so far and either produces an intervention or remains silent. The evaluation covers 1,342 decision points across the 100 SWE-chat test instances and 602 decision points across the 143 PARE instances. We evaluate each intervention against the references associated with that trajectory.

**Relevance.** To assess relevance as defined in § 2.2, an LLM compares each intervention against the instance's references. A match requires the intervention to identify the same problem or suggest the same action as the reference, while preserving its key details. Figure 2(b) shows two example agentic runs on the same user trajectory. Run A proactively proposes the compatibility reminder only at turn 4, while Run B offers it at turns 4 and 11; both remain silent at all other turns. In both runs, the interventions suggest keeping `user_id` alongside `id` and match the reference because they address the same compatibility constraint, while if it is a warning about an unrelated bug, it would not be considered a match. Details and validation of reference matching are in Appendix H.

We measure relevance through reference matching using three metrics. *Match rate* is the fraction of instances in which at least one intervention matches a reference. For each instance, *recall* is the

number of references the agent successfully matches, divided by the total number of references in that instance; *precision* is the number of the agent's interventions that match at least one reference, divided by the total number of interventions it produces. Averaging the per-instance recall and precision across all instances gives *macro recall* and *macro precision*, respectively, with each instance receiving equal weight. For instances with no interventions, we set precision to zero. These metrics count reference matches regardless of timing. A single intervention may address several issues and therefore match more than one reference.

**Timeliness.** To assess timeliness, we compare the decision point at which the agent offers a matching intervention with the reference's *assistance window* $[a, b] \subseteq \{1, \ldots, T\}$. A matching intervention at $t$ is early if $t < a$, in-window if $a \leq t \leq b$, and late if $t > b$.

In SWE-chat, we use an LLM to examine the trajectory and available context to identify an assistance window $[a, b]$ for each reference: $a$ is the first decision point at which the assistance described in the reference is supported by the evidence and can usefully inform the user's decisions, and $b$ is the last point at which that assistance remains useful. In the example trajectory in Figure 2, the developer proposes the rename at turn 7, making the compatibility constraint relevant. The compatibility issue remains unresolved at the end of the trajectory, so the assistance window is $7, 20$. Details of how we construct and validate benchmark windows are provided in Appendix H.1.

In PARE, we use assistance windows $[a, T]$. As in SWE-chat, $a$ denotes the first decision point at which the assistance described in the reference is supported by the evidence and can usefully inform the user's decisions. PARE already specifies when the assistant should offer help, so we use the corresponding decision point as $a$. Since the scenario does not separately specify when the help ceases to be useful, we use the final decision point $T$ as the upper bound.

We measure timeliness using macro in-window recall. For each instance, *in-window recall* is the fraction of its references that the agent matches at least once within their assistance windows; averaging this fraction across all instances gives *macro in-window recall*. It therefore requires both a reference match and appropriate timing. Each reference is counted once, even if the agent matches it multiple times. Both Run A and Run B in Figure 2(b) match the reference, but only Run B receives in-window recall credit because its intervention at turn 11 falls within $[7, T]$. We additionally discuss intervention frequency in Appendix C.

An ideal agent would match every reference within its assistance window, with every intervention matching at least one reference. For instances with an available decision point in every assistance window, this would give 100% recall, precision, match rate, and in-window recall. Intervention rate is the fraction of decision points at which the agent intervenes, so it has no fixed ideal value.

## 4 Baseline Agents

We experiment with five LLMs with our baseline proactive agent setup: GPT-5.6-sol (OpenAI, 2026b), GPT-5.4 (OpenAI, 2026a), GPT-oss-120b (Agarwal et al., 2025), Qwen3.5-9B (Team, 2026), and Gemma-3-12B (Team et al., 2025). To decide whether to proactively intervene, an agent needs information about what the user is doing, the environment they are working in, and relevant background information. Thus, we provide three sources of context $C_t$ (§ 2.2) in both software engineering and productivity domains:

- **Live context** contains the user trajectory observed up to the current decision point. In SWE-chat, it includes the user's messages and the coding assistant's responses and actions observed so far. In PARE, it includes user actions and environment events.
- **Workspace** contains the initial state of the user's working environment. For SWE-chat, this provides the repository name, commit, and a summary of its top-level files and directories. For PARE, this provides state of applications such as calendar entries, contacts, notes, and inbox contents.
- **Knowledge pool** contains related background information beyond the current user trajectory, such as previous discussions or past activities, that may help the agent identify useful proactive assistance. In SWE-chat, it includes pull requests, issues, and their discussions, excluding the ones that are used to construct references. In PARE, it includes records of past activity, such as requests for rides, message threads, and mails delivered.

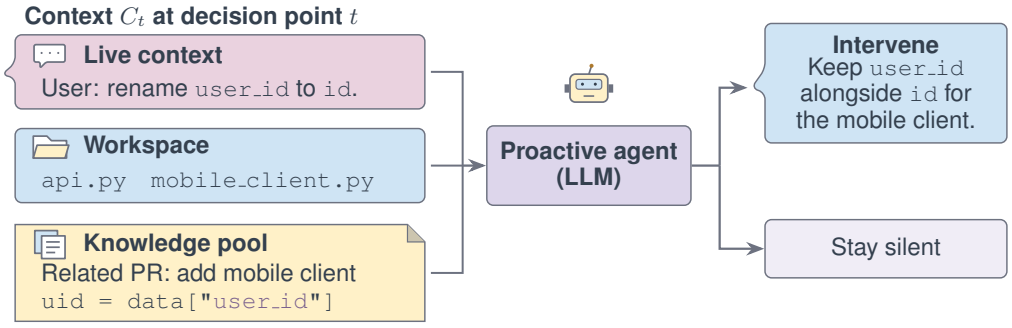


Figure 3: Context supplied to a proactive agent, illustrated with the API-field rename from Figure 1. Here, a related PR supplies the client code supporting the reminder. PARE uses the same three context sources and intervention decision.

Figure 3 shows how these sources inform the proactive intervention decision using the API-field rename from the example in Figure 1. The live context shows the user's request to rename `user_id` to `id`, while the workspace shows the repository's files and directories. A PR in the knowledge pool contains mobile-client code that still reads `user_id`. Removing this field could therefore break the client, so the agent suggests keeping both fields. Details of how we prepare the context and run the agents are provided in Appendix D.

At each decision point, the agent either offers assistance or remains silent. We evaluate the proactively proposed intervention in terms of what assistance it offers (relevance) and when (timeliness). In PARE, the agent also needs to state which application to use, what action to take, and the inputs needed for that action. The baseline agent decides using only the provided context, without asking applications for more information. In a separate experiment, the agent can carry out actions in PARE's applications, and we evaluate whether it completes the task specified in the scenarios (see Appendix E for details).

## 5 Results and Analysis

We evaluate each of the five baseline agents on the SWE-chat test split (100 instances) and all 143 PARE instances. Within each instance, the agent decides whether to intervene and what assistance to offer at every decision point. This gives 1,342 decision points per agent on SWE-chat and 602 on PARE. We evaluate these proactive interventions against the references associated with each instance, with 306 references in SWE-chat and 143 in PARE, using the metrics defined in § 3.3.

### 5.1 Baseline Performance

Table 3 reports the performance of all five baseline agents. On SWE-chat, GPT-5.4 achieves the highest macro recall and match rate, but some of its matching interventions arrive only after the assistance window has closed. GPT-oss-120b matches fewer references overall, yet more within their assistance windows. The agents match different sets of references, so higher in-window recall does not necessarily mean better timing for the same assistance. GPT-5.6-sol has a lower intervention rate than most other agents while achieving the highest precision, meaning that on average across instances, a larger proportion of its interventions match the references.

On PARE, each agent's in-window recall is close to its recall. Once an assistance window opens, it stays open until the end of the user trajectory, so a later matching intervention can still count as timely. Low macro in-window recall therefore mainly comes from references that agents never match, rather than matches made outside the assistance window. The PARE results also show that a higher intervention rate does not necessarily lead to higher recall. GPT-5.6-sol has the highest precision and recall, while three other agents have higher intervention rates but lower scores on both metrics. This suggests that macro recall depends on what assistance an agent offers as well as its intervention rate. Results from a second LLM judge are provided in Appendix G.4.

Table 3: Baseline results on SWE-chat and PARE. Recall, precision, and in-window recall are macro-averaged over instances. Intervention rate is the percentage of all decision points at which an intervention is proposed. he highest scores within each benchmark are shown in bold. *Note: PARE has one reference per instance, so match rate equals macro recall.*

| | Relevance | | | Timeliness | |
|---|---|---|---|---|---|
| **Model** | **Recall ↑** | **Precision ↑** | **Match rate ↑** | **In-window recall ↑** | **Intervention rate** |
| *SWE-chat (100 instances, 306 references, 1,342 decision points)* | | | | | |
| GPT-5.6-sol | 0.055 | **0.071** | 0.120 | 0.030 | 30.5% |
| GPT-5.4 | **0.070** | 0.066 | **0.160** | 0.032 | 73.0% |
| GPT-oss-120b | 0.062 | 0.054 | 0.150 | **0.056** | 32.5% |
| Gemma-3-12B | 0.048 | 0.034 | 0.110 | 0.029 | 39.3% |
| Qwen3.5-9B | 0.048 | 0.057 | 0.130 | 0.027 | 37.6% |
| *PARE (143 instances, 143 references, 602 decision points)* | | | | | |
| GPT-5.6-sol | **0.203** | **0.185** | **0.203** | **0.203** | 19.6% |
| GPT-5.4 | 0.077 | 0.062 | 0.077 | 0.070 | 15.3% |
| GPT-oss-120b | 0.084 | 0.032 | 0.084 | 0.063 | 69.9% |
| Gemma-3-12B | 0.119 | 0.052 | 0.119 | 0.091 | 83.6% |
| Qwen3.5-9B | 0.049 | 0.018 | 0.049 | 0.042 | 48.5% |

Table 4: GPT-4.1-family results on 100 SWE-chat test instances. Recall, precision, and in-window recall are macro-averaged. Intervention rate is the percentage of the 1,342 decision points with an intervention, including repeated interventions. Bold marks the highest relevance and timeliness scores within each group.

| | Relevance | | | Timeliness | |
|---|---|---|---|---|---|
| **Model** | **Recall ↑** | **Precision ↑** | **Match rate ↑** | **In-window recall ↑** | **Intervention rate** |
| *Base models* | | | | | |
| GPT-4.1 | **0.050** | **0.047** | **0.080** | 0.023 | 40.8% |
| GPT-4.1-mini | 0.043 | 0.039 | 0.070 | **0.029** | 44.6% |
| GPT-4.1-nano | 0.030 | 0.025 | 0.050 | 0.010 | 48.9% |
| *SFT models* | | | | | |
| GPT-4.1 | 0.125 | 0.102 | 0.220 | 0.087 | 86.3% |
| GPT-4.1-mini | **0.178** | **0.171** | **0.280** | **0.128** | 96.5% |
| GPT-4.1-nano | 0.023 | 0.019 | 0.040 | 0.005 | 21.2% |

### 5.2 Supervised Fine-Tuning

We use instances from the SWE-chat training split for supervised fine-tuning (SFT), training agents to decide whether to proactively intervene, what assistance to offer, and when to offer it. We construct one training conversation for each instance, covering its decision points sequentially. At each decision point, we provide the model with context from the three sources described in § 4. In the training data, we specify an intervention when a decision point falls within at least one reference's assistance window; otherwise, we specify silence. We construct each intervention based on the corresponding references. Data construction and the training configuration will be released.

Table 4 compares the base and SFT models in the GPT-4.1 family (OpenAI, 2025). After SFT, GPT-4.1 and GPT-4.1-mini score higher on recall, precision, match rate, and in-window recall, and both intervene more frequently. GPT-4.1-mini achieves the highest scores among the SFT models, with recall rising from 4.3% to 17.8%. GPT-4.1-nano intervenes less frequently after SFT and scores lower on all four relevance and timeliness metrics.

### 5.3 Analysis

**Performance by Behavior Category.** We found that agent's proactive assistance performance varies across types of assistance. We use the taxonomies in § 3.1 to compute macro recall separately for each category. On SWE-chat, GPT-5.6-sol has higher recall for *Sharing context and assumptions* than for *Concrete fix or workaround*, although the ordering of categories differs across agents. On PARE, GPT-5.6-sol has higher recall for *Cancelling an item* than for *Creating a record or tracker*. The full category breakdowns and checks using alternative category assignments are provided in Table 12, Table 26, and Appendix M.

**Context Availability.** The live context may not contain all the information needed to identify useful proactive assistance. We therefore analyze whether adding workspace and knowledge pool improves recall or precision for GPT-5.4 on PARE, while keeping everything else fixed. Compared with live context alone, providing workspace and knowledge pool improves both recall and precision (shown in Table 25). On SWE-chat, Qwen3.5-9B also matches different references under different retrieval configurations; these configurations and their results are detailed in Appendix F.

**Event Representation.** Even when agents observe the same events, the details shown about those events can change the proactive assistance they offer. We compare two settings on PARE: both provide event summaries, but one also includes full event details. For example, we add the body of an email to a summary that shows only its subject line. The event sequence and all other context are the same in both settings. We found that providing full event details increases recall for all agents (see Appendix G.6 for more details).

### 5.4 Error Analysis

To understand why agents propose inappropriate actions, we conduct case studies on failed interventions alongside the context available at the corresponding decision points. The following PARE cases illustrate how a suggestion can relate to the user's current activity yet still considered failed.

**Relying on assumptions.** In a PARE instance, an agent proactively offers to download an attachment when only the email's subject is visible. The intervention assumes that an attachment is available without sufficient evidence in the visible context.

**Redundant actions.** In another PARE instance with a earbuds restock event, an agent proactively offers to add the earbuds to the cart, even though it's visible that the cart already includes them. The intervention repeats a completed action instead of helping the user proceed to checkout.

These examples illustrate two distinct problems: missing information and failure to use information already available. Agents should confirm their assumptions before offering help. They should also check what the user has already done, so they do not suggest repeating a completed action.

## 6 Conclusion and Discussion

Ping evaluates what an agent contributes during a user trajectory and when it offers that contribution. Collaboration-derived taxonomies guide reference extraction in software engineering and category analysis in productivity, while reference coverage and timing distinguish missed contributions from poorly timed delivery. Across the tested baselines, many references remain unrecovered. Exposing fuller details of the same PARE events increases recovery across four models under two judges, showing that observation design materially affects this evaluation.

Future work can use Ping to study contribution selection under an intervention budget, learning from trajectory-level feedback, and evidence retrieval. Interactive evaluation can connect these proposal-level decisions to execution outcomes and user acceptance. Evaluation scope and limitations, including reference coverage and timing annotation, are discussed in Appendix B.

### AI USE STATEMENT

Generative AI tools assisted with paper revision, figure editing and experiments. LLMs also serve as components of the research pipeline for taxonomy discovery, contribution extraction, utility and timing audits, and evaluation.

### ETHICS STATEMENT

This work uses existing coding-session and meeting datasets and simulated personal-assistant scenarios. Public collaboration records can still contain personal or sensitive information, and proactive assistance can expose irrelevant information or interrupt users. The primary evaluation records interventions without executing actions in real user accounts; the execution experiment operates in PARE's simulated applications.

### REPRODUCIBILITY STATEMENT

The benchmark definitions and context interface are specified in § 3 and § 4. We will release all materials to construct the taxonomy and data.

## REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Joachim Baumann, Vishakh Padmakumar, Xiang Li, John Yang, Diyi Yang, and Sanmi Koyejo. Swe-chat: Coding agent interactions from real users in the wild. *arXiv preprint arXiv:2604.20779*, 2026.

Tenghao Huang, Zhaoxuan Tan, Muhao Chen, Jonathan May, Mengting Wan, Longqi Yang, Pei Zhou, and Sihao Chen. Glare: Generative learning via adversarial reward estimation for social dynamics forecasting. *arXiv preprint arXiv:2609.12165*, 2026.

Deepak Kumar. Swe-prbench: Benchmarking ai code review quality against pull request feedback. *arXiv preprint arXiv:2603.26130*, 2026.

Lehui Li, Ruixuan Jia, Guo-Ye Yang, and Jia Li. An empirical study of proactive coding assistants in real-world software development. *arXiv preprint arXiv:2605.05700*, 2026.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, et al. Proactive agent: Shifting llm agents from reactive responses to active assistance. In *International Conference on Learning Representations*, volume 2025, pp. 47431–47457, 2025.

Deepak Nathani, Cheng Zhang, Chang Huan, Jiaming Shan, Yinfei Yang, Alkesh Patel, Zhe Gan, William Yang Wang, Michael Saxon, and Xin Eric Wang. Proactive agent research environment: Simulating active users to evaluate proactive assistants. *arXiv preprint arXiv:2604.00842*, 2026.

OpenAI. Introducing GPT-4.1 in the API. `https://openai.com/index/gpt-4-1/`, April 2025.

OpenAI. Introducing gpt-5.4. `https://openai.com/index/introducing-gpt-5-4/`, March 2026a.

OpenAI. Gpt-5.6: Frontier intelligence that scales with your ambition. `https://openai.com/index/gpt-5-6/`, July 2026b.

Gil Pasternak, Dheeraj Rajagopal, Julia White, Dhruv Atreja, Matthew Thomas, George Hurn-Maloney, and Ash Lewis. Beyond reactivity: Measuring proactive problem solving in llm agents. *arXiv preprint arXiv:2510.19771*, 2025.

Chirag Shah, Ryen White, Reid Andersen, Georg Buscher, Scott Counts, Sarkar Das, Ali Montazer, Sathish Manivannan, Jennifer Neville, Nagu Rangan, et al. Using large language models to generate, validate, and apply user intent taxonomies. *ACM Transactions on the Web*, 19(3):1–29, 2025.

Omar Shaikh, Valentin Teutschbein, Kanishk Gandhi, Yikun Chi, Nick Haber, Thomas Robinson, Nilam Ram, Byron Reeves, Sherry Yang, Michael S Bernstein, et al. Learning next action predictors from human-computer interaction. *arXiv preprint arXiv:2603.05923*, 2026.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W White, Longqi Yang, et al. Tnt-llm: Text mining at scale with large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 5836–5847, 2024.

## A Related Work

**Predicting user behavior.** Prior work uses records of user activity to predict future actions or infer the tasks users intend to perform (Shaikh et al., 2026; Li et al., 2026). These approaches derive evaluation targets from observed behavior. Ping also considers useful assistance that may be absent from the user's subsequent actions, such as warning about an overlooked dependency.

**Evaluating proactive assistance.** Another line of work studies agents that identify useful tasks and propose or carry them out without an explicit user request (Lu et al., 2025; Pasternak et al., 2025; Nathani et al., 2026). Ping extends on this direction by evaluating what assistance agents offer and when they offer it over user trajectories in software engineering and productivity.

**References from human collaboration.** Human discussions can also provide references for evaluation. Prior work uses code-review comments to assess whether agents identify issues raised by human reviewers (Kumar, 2026). Ping derives taxonomies of proactive assistance from GitHub comments and meeting transcripts. These taxonomies guide reference extraction for SWE-chat and analysis of assistance types in PARE.

## B Evaluation Scope and Limitations

**Reference coverage and useful assistance.** Reference recall and reference-match precision measure agreement with the retained contributions, rather than the full range of useful assistance. Human annotation of selected proposals confirms useful non-matches and useful points outside the annotated SWE-chat intervals (Appendix H.3). These targeted samples document concrete disagreements; broader sampling is needed to estimate their prevalence and the association between reference matching and usefulness. Reference provenance and source-restricted results are reported in Appendix D, including Table 8. The discovered categories characterize the sampled collaboration rather than an exhaustive inventory of either domain.

**Timing and information availability.** SWE-chat's utility intervals are conditional on the benchmark-supplied context, including retrospective repository records. They characterize assistance under that context, rather than a deployment restricted to records already available at the historical session time. PARE uses the scripted proposal as its lower timing anchor and the end of the episode as its upper bound. Some script-early matches already have supporting evidence, and the evaluation does not independently annotate an expiry time (Appendix L). Historically restricted retrieval and human annotation of full useful intervals would extend these settings.

**Judgments and baseline comparisons.** LLMs supply taxonomy labels, utility intervals, and content matches. Second-model checks quantify sensitivity to these judgments; human annotations are reported separately with their sampling and AI-assisted workflow. Main scores retain failed decisions, so cross-model differences include context limits and serving reliability alongside contribution selection. Failure counts and uncertainty estimates appear in Appendix D, Appendix L, and Appendix K. The results describe the evaluated configurations and saved runs.

**Controlled analyses.** The event-representation experiment changes the information included in the prompt and its length together. Each arm has one run per model; the paired estimates and second judge quantify uncertainty conditional on those outputs. Uniform intervention thinning compares saved proposal sets at equal counts rather than testing a learned budget policy. These analyses study reference recovery, with usefulness and user acceptance requiring additional evaluation.

**Interaction and execution.** The primary evaluation replays a fixed trajectory with fixed initial application state. It assesses proposed assistance without allowing the proposals to change later observations or user behaviour. Native PARE execution supplies a complementary outcome under a different interaction protocol (Appendix E). Deployment studies would additionally need to assess interruption costs, user acceptance, privacy, and action safety.

## C Intervention-Frequency Diagnostics

The SWE-chat *silence rate* is the fraction of decision points outside all annotated windows at which no intervention is produced. It includes failed calls under the harness convention, so it combines deliberate silence with runtime behaviour. PARE's *distractor intervention rate* is the fraction of injected distractor steps with an intervention. Such a proposal can still address the existing application state or a related event cue. Both rates are descriptive diagnostics, without a preferred direction.

Table 5: SWE-chat silence rates outside annotated utility windows. Missing and failed outputs count as non-interventions; see the runtime accounting in Appendix D.

| Model | Silence rate |
|---|---|
| GPT-5.4 | 0.248 |
| GPT-oss-120b | 0.749 |
| GPT-5.6-sol | 0.718 |
| Qwen3.5-9B | 0.666 |
| Gemma-3-12B | 0.697 |

Table 6: PARE intervention-frequency diagnostics over the same 602 decision points, including 396 injected distractor steps. Distractor intervention rate is descriptive.

| Model | Interventions | Distractor interventions | Distractor rate |
|---|---|---|---|
| GPT-5.4 | 92 | 37 | 0.093 |
| GPT-5.6-sol | 118 | 39 | 0.098 |
| GPT-oss-120b | 421 | 253 | 0.639 |
| Qwen3.5-9B | 292 | 175 | 0.442 |
| Gemma-3-12B | 503 | 303 | 0.765 |

## D Taxonomy Mining and Data

**Sources and sampling.** The GitHub pipeline samples public GH Archive events from January 2025 at rate 0.001 with seed 42, stratifying by day and event type. It expands the sampled events into discussion threads and assigns a primary category to each thread, with optional secondary categories. The meeting pipeline uses the collection associated with GLARE, including QMSum-ICSI, QMSum-AMI, and MeetingBank in the analysed subset. Consecutive same-speaker utterances are

merged before classification. AMI contains scripted role-play; the collection is therefore not uniformly naturalistic.

Table 7: Collaboration data used for taxonomy discovery. Counts and rates refer to proactive source segments before the software-engineering benchmark excludes negative-valence contributions.

| Domain | Source | Proactive | Rate |
|---|---|---|---|
| Software engineering | GH Archive (PRs, issues) | 7,314 | 32.5% |
| Personal productivity | GLARE – MeetingBank (city council) | 568 | 3.5% |
| Personal productivity | GLARE – QMSum-ICSI (lab meetings) | 742 | 2.6% |
| Personal productivity | GLARE – QMSum-AMI (scripted role-play) | 245 | 1.0% |

**Classification and consolidation.** An LLM classifier applies the unprompted gate using the preceding context and assigns a category or proposes a new one. Subsequent classifications receive the accumulated vocabulary. A second LLM pass merges near-duplicate categories, after which an explicit mapping consolidates meeting categories into twelve named buckets and a residual bucket. Thus the broad meeting buckets are not solely the output of an unconstrained classifier. The software-engineering table reports the eight retained categories, normalized over 7,263 primary assignments after deduplication; negative-valence dismissals are excluded from that denominator. The archived GitHub log contains 22,499 successful records for 22,471 distinct discussions. We retain the first successful saved record per discussion in file order, removing 28 duplicate records, including 18 proactive primary assignments; this recount preserves the saved consolidated labels and does not rerun taxonomy discovery. Secondary category assignments are excluded from the shares. The meeting table retains the full denominator of 1,555 examples, including the residual group.

Proactive rates differ across the sampled sources (Table 7), but sampling units, topics, and role structure also differ. These observations do not isolate an effect of written versus spoken collaboration or establish saturation of the category inventory.

**Prompt inputs and execution.** The GitHub classifier receives a rendered discussion and the current category names and descriptions, with a 16,384-token JSON output limit and ten concurrent workers. The meeting classifier receives the target speaker turn, at most 100 preceding merged turns, and the current vocabulary; it skips targets shorter than five words and uses a 2,048-token output limit with five workers. Category consolidation receives the accumulated labels and preserves the non-proactive category. Shared vocabulary updates make discovery order-dependent, so reproducing a taxonomy requires retaining the source order and consolidation mapping as well as the prompts. The local artifacts include seven exact prompt strings for discovery, consolidation, utility auditing, PARE category assignment, and content matching, together with source and prompt hashes. These document the current source snapshot; deployment aliases supplied through environment configuration do not establish immutable historical model versions.

**SWE-chat split.** The release contains 1,070 session–artifact instances from 995 sessions. Connected groups sharing a session, workspace snapshot, or source contribution remain in one partition. The split uses seed 20260728 and produces 870 training, 100 validation, and 100 test instances; the corresponding gold counts are 1,849, 364, and 306. The test partition contains 1,342 user decision points. These grouping constraints do not imply that the test instances are statistically independent.

**Gold provenance.** The gold metadata assigns 1,272 of 2,519 retained occurrences to the pull-request or issue author and 1,247 to other participants; the test split contains 84 and 222, respectively. Artifact authorship does not establish identity with the session operator. Joining source-comment URLs to the shipped context records recovers account identities for 2,323 occurrences, including 302 of 306 test occurrences. It identifies 940 automated-source occurrences overall and 210 in the test split (178 Greptile, 28 CodeRabbit, and four Copilot). The remaining test items comprise 92 from accounts typed as users and four with unresolved source identities. These are provenance counts, not judgments that automated contributions are invalid or that user accounts necessarily contain human-written text.

Table 8: SWE-chat source-restricted evaluation: 92 reference items from accounts typed as users, spanning 66 test instances. Saved model outputs, content matches, and utility windows are unchanged. Instance coverage uses these 66 instances; in-window rate uses the first match of each recovered item. Account type is a provenance filter, not a human eligibility judgment.

| Model | Instance coverage | Recall | In-window rate |
|---|---|---|---|
| GPT-5.4 | 8/66 | 13/92 (14.1%) | 7/13 (53.8%) |
| GPT-5.6-sol | 6/66 | 7/92 (7.6%) | 5/7 (71.4%) |
| GPT-oss-120b | 5/66 | 5/92 (5.4%) | 3/5 (60.0%) |
| Qwen3.5-9B | 4/66 | 4/92 (4.3%) | 1/4 (25.0%) |
| Gemma-3-12B | 3/66 | 3/92 (3.3%) | 0/3 (0.0%) |

**Source-restricted evaluation.** We evaluate the five saved model runs against the 92 references from accounts typed as users, spanning 66 test instances from 65 sessions (Table 8). The filter uses source metadata without consulting model outcomes. We retain the original content judgments and utility windows, counting each reference once at its first match; subset instance coverage uses only the 66 instances with at least one retained reference. GPT-5.4 recovers 13/92 references (14.1%), of which seven first arrive in-window. The other models recover three to seven references, so low recovery persists under this source restriction, although their ordering changes. The subset contains 46 supporting-artifact and 37 context-sharing items and has a different category mix from the full benchmark. This is a source-type sensitivity analysis, rather than a human-certified subset of unprompted contributions.

The benchmark utility audit explicitly includes an `already_requested_or_active` exclusion, applied to 115 of the 5,357 candidates. Its rubric nevertheless permits reminders about known unfinished work when they add corrective or coordination value. Utility and unprompted eligibility therefore overlap without being identical. A targeted evidence review identifies two test occurrences in which the current user input already quotes the complete reference and explicitly requests that it be addressed at the sole retained decision point. Under the definition in § 2, repeating these references is reactive. This adjudication concerns the two session–reference occurrences, not every occurrence of the same source comment.

We recompute the five main-model reports after excluding these two occurrences, retaining all 100 test instances and reusing the saved predictions and content judgments. None of the models recovered either item, so recovered counts, instance coverage, and precision are unchanged; GPT-5.4's recall changes from $31/306$ (10.13%) to $31/304$ (10.20%). The original release and main-table denominator remain fixed, with this exclusion reported as a sensitivity analysis. This agent evidence review is not an independent human audit or an estimate of dataset-wide eligibility error.

**Assistance-window audits.** Construction first applies a transcript-only window audit, then a utility audit using the transcript and supporting excerpts from the public context lanes. At decision $t$, the current user message is visible, assistant events after that message are withheld until the next decision, and workspace and knowledge-pool evidence is available from decision 1. The utility audit checks grounding, relevance, actionability, and unresolved status and returns the earliest qualifying interval. All 2,519 retained gold items have valid utility intervals, including all 306 test items. The archived evaluations used the earlier windows, which contain 123 test-set fallbacks to $[1, T]$. Those fallbacks do not imply that the later utility audit failed; Appendix H.1 reconciles the two sources and reports the resulting timing changes.

**Agent implementation.** SWE-chat uses a *gatherer* to retrieve material and assemble an evidence packet, followed by a *decider* that outputs the intervention decision and content. The same LLM performs both stages within each evaluated configuration, and the decision stage retains its prior decisions as session memory. Record-level retrieval considers up to 256 record summaries and selects eight, using previews of at most 1,000 characters. It then considers up to 40 chunks per selected record, again using 1,000-character previews. The main runs retrieve session and knowledge-pool content and receive a repository overview; the workspace-retrieval variant additionally reads file bodies from the supplied snapshot.

PARE renders the enabled context lanes directly into the decision prompt. The workspace is reconstructed from the scenario's initialization routine, and the historical activity used as the knowledge pool is non-empty in 134 of the 143 scenarios. All five main runs use `agent_state=stateless`: each decision receives the full observed event prefix and the enabled workspace and knowledge-pool context, but not the agent's own previous outputs. Thus *stateless* refers only to the absence of feedback from prior model decisions, not to the absence of interaction history. The harness also supports a stateful mode that adds a digest of earlier decisions; this mode is not used in the main results. Both domains provide the same three context types, while differing in retrieval and decision-history handling.

Table 9: Telemetry from selected archived SWE-chat evaluation runs. Tokens include prompt and completion tokens; latency sums model-call durations within each turn and is not elapsed wall time. GPT-5.6-sol telemetry covers the available partial run, while its recovery scores retain all 100 test instances.

| Model | Tokens / turn | Latency / turn | Tokens / run | Judge tokens |
|---|---|---|---|---|
| GPT-oss-120b | 116,714 | 21.4 s | 157M | 0.7M |
| GPT-5.6-sol | 223,091 | 234.0 s | 267M | 0.6M |
| GPT-5.4 | 375,533 | 52.2 s | 504M | 1.7M |

**Saved-run diagnostics.** Both GPT-5.4 and GPT-oss-120b retain packets and Decider messages for all 1,342 SWE-chat decisions. GPT-5.4 logs five Decider failures, all HTTP 429. GPT-oss-120b logs 767: 760 HTTP 400, four context-overflow errors, and three JSON-parsing failures. An offline length audit using `o200k_base` finds that message content plus the requested 800-token output allowance exceeds 131,072 tokens for all 764 HTTP 400 or context-overflow records, while all 575 successful calls fall below this threshold. This separation implicates accumulated context as a principal runtime limitation, although these proxy counts omit message-format overhead and the historical HTTP 400 response bodies were not retained. Among its 769 non-interventions outside annotated windows, 686 are failed calls and 83 are valid silence decisions. Twenty gold items have no successful Decider call inside their window, of which 19 are never recovered. The archived implementation also inserts failed calls into subsequent history as silence decisions. The corrected implementation excludes these synthetic decisions from memory; the reported runs are unchanged. These failures retain zero recovery credit in the main results; they cannot establish a failure of evidence use or a deliberate choice to remain silent.

Grouping the same references by source further distinguishes the two runs. GPT-5.4 and GPT-oss-120b recover 18/210 and 22/210 automated-reviewer references, respectively, compared with 13/92 and 5/92 references from accounts typed as users; neither recovers the four references with unresolved account identities. Account type does not establish human authorship, and source, category, repository, and runtime failures are not controlled in this descriptive comparison. It therefore does not isolate a causal effect of reference provenance.

## E Proposal and Execution

The main evaluation uses static proposal replay. A separate archived experiment runs PARE's native observation–execution loop: an observer can query applications with read-only tools, a simulated user accepts or rejects proposals, and an executor acts on accepted proposals. The observer and executor use GPT-5.4, the simulated user uses GPT-5-mini, and the episode budget is ten turns. Native scenario validators inspect application state or event-log conditions, depending on the scenario.

The archived native run succeeds on 25/143 scenarios (17.5%), whereas static GPT-5.4 replay recovers 11/143 references (7.7%). These are different outcomes under different observation, interaction, and validation protocols. Of the 25 native successes, only three are recovered in static replay; 15 remain silent throughout, four emit only non-matching proposals, and three have no observable decision point (Table 10). Thus 18 of the 22 unrecovered native successes produce no proposal for the matcher to accept or reject. All 25 have exact scenario-ID counterparts; one other identifier differs between archives and is excluded from cross-run alignment. The retained native summaries lack the

full runtime traces at their recorded paths, so they cannot identify which queries or user responses enabled success. The native write-action counter is inconsistent with successful validations and is not used.

Table 10: Static GPT-5.4 replay outcomes for the 25 scenarios with a successful archived native PARE run. All 25 have exact scenario-ID counterparts. The protocols differ; this breakdown does not isolate a causal explanation.

| Static replay outcome | Scenarios |
|---|---|
| Reference recovered | 3 |
| Silent at every decision point | 15 |
| Interventions emitted, no reference match | 4 |
| No observable decision point | 3 |

Validator success also need not establish semantic coverage of the scripted reference. In the current source, the anniversary-gift scenario checks that the agent proposed assistance and called checkout with a specified discount code; it does not check recognition of the anniversary or intended recipient. The roommate-search scenario checks that a proposal and a messaging action occurred without checking the recommended apartments. These examples characterize the current validator code, not the unavailable historical execution traces. Native success and reference recovery therefore measure different properties; neither score is an upper bound on the other.

## F Lane Ablations

**SWE-chat.** We compare the main Qwen3.5-9B configuration with runs retrieving session, workspace, or knowledge-pool evidence. These are history-conditioned retrieval configurations: the current developer message remains visible, and the full session prefix supplies retrieval keywords even when its direct presentation is disabled. Thus workspace and knowledge-pool variants do not remove all influence of session history. All use the same model and retrieval limits, while workspace retrieval additionally reads file bodies. We exclude the union of 19 instances with logged context-overflow failures, retaining 81 shared instances; main-table results retain all 100. Timing and silence use the utility-audited windows.

Table 11: SWE-chat history-conditioned retrieval variants with Qwen3.5-9B over 81 shared instances (228 gold), using utility-audited windows. All retain the current message; session history supplies retrieval keywords even when not directly displayed. The workspace variant additionally reads file bodies. Union is an oracle combination.

| Configuration | Instance | Precision | Recall | In-window | Silence | Interventions |
|---|---|---|---|---|---|---|
| Full | 0.099 | 0.068 | 0.079 | 0.833 | 0.550 | 191 |
| Live session only | 0.111 | 0.123 | 0.110 | 0.760 | 0.669 | 122 |
| Workspace evidence | 0.123 | 0.097 | 0.079 | 0.722 | 0.437 | 195 |
| Knowledge-pool evidence | 0.086 | 0.068 | 0.053 | 0.750 | 0.543 | 192 |
| Union of the four | 0.247 | — | — | — | — | — |

The live-session-only run has higher observed reference recall than the main configuration, 0.110 versus 0.079. The oracle union of all four configurations covers 20 instances, compared with eight for the main configuration. This comparison describes variation across retrieval configurations, with differences in history conditioning and file access as specified above.

## G Category and Qualitative Analyses

The observed category ordering varies across models: sharing context has the highest reported recall for GPT-5.4, while code review contributes many of the recovered items for other models. Concrete fixes are recovered less often than code-review feedback for GPT-5.4 and Qwen3.5-9B, but this does not isolate difficulty in constructing a fix from retrieval, reference specificity, or annotation effects.

Table 12: Reported gold recall by category on the 100-instance SWE-chat test split. All columns use the Gold denominators, including instances with missing or failed calls. Categories with only two or three gold items support descriptive counts rather than reliable model rankings.

| Category | Gold | GPT-5.4 | GPT-oss-120b | Qwen3.5-9B | GPT-5.6-sol |
|---|---|---|---|---|---|
| Code review feedback | 102 | 0.127 | 0.118 | 0.157 | 0.069 |
| Concrete fix or workaround | 84 | 0.036 | 0.083 | 0.024 | 0.036 |
| Adding supporting artifacts | 52 | 0.038 | 0.058 | 0.019 | 0.019 |
| Sharing context and assumptions | 43 | 0.233 | 0.047 | 0.047 | 0.116 |
| Flagging concerns and open questions | 18 | 0.167 | 0.167 | 0.111 | 0.167 |
| Probing or clarifying question | 3 | 0.000 | 0.000 | 0.000 | 0.000 |
| Offering collaboration | 2 | 0.000 | 0.000 | 0.000 | 0.000 |
| Pointing to related resources | 2 | 0.000 | 0.000 | 0.000 | 0.000 |

Table 13: Three interpretations of saved PARE proposals. Situations and outputs are paraphrased; matching and script-relative timing retain the recorded scores. The final row uses an existing exploratory benefit assessment, not an execution outcome or independent human label.

| Visible evidence | Proposed contribution | Score and interpretation |
|---|---|---|
| A workshop includes a tripod; a separate tripod order is still processed. | GPT-5.6-sol offers to cancel the redundant order at step 1. | Content matches, but precedes $[4, 4]$. The script marks it early although the key premises are already visible. |
| Earbuds are already in the cart when a restock event arrives. | Gemma offers to add them to the cart at step 4. | At the reference point, but unmatched: it repeats cart addition instead of offering checkout. |
| A message requests removal of saved apartments below 500 sqft; one visible candidate is 480 sqft. | Gemma offers to remove that candidate. | Unmatched, but coded as supported partial benefit: it would remove one unsuitable apartment while leaving another. |

The three categories with only two or three gold items each are too small to support general difficulty rankings. Category shares in the mined source corpus and the benchmark also differ because the latter requires contributions linked to an available session and retained by the construction gates.

**Unmatched interventions.** A model-only usefulness audit sampled 150 unmatched interventions, thirty from each of five configurations, and mixed them with 60 gold contributions shown at their annotated window start. The rater was not given the provenance label and judged whether each message would help in the displayed session context. It marked 81 of 150 unmatched interventions (54%) and 39 of 60 gold items (65%) useful. These sample-level judgments illustrate that content matching and judged usefulness can disagree; they do not estimate each model's true precision. Formatting may still reveal provenance, and this separate 210-item audit has not received human verification.

**Completed-work reports.** Some proposals narrate work as already completed rather than offering an additional contribution. The strict semantic matcher excludes these messages even when they mention a gold item's subject. This matters because a single completion summary may name several reference items. Comparing scores across matcher versions requires rescoring the same saved predictions; the speech-act rule is not interchangeable with a generic topic match.

### G.1 Contribution Selection on Identical Inputs

**Reading the scores.** Table 13 distinguishes a script-early content match, a redundant non-match at the reference point, and a non-match with supported partial benefit. The timing example is selected by the smallest SHA-256 hash of scenario ID and step among GPT-5.6-sol's saved early matches; the other two are existing diagnostic illustrations, not newly sampled annotations. All descriptions paraphrase saved inputs and proposals. The example selection is illustrative, not a frequency esti-

Table 14: Exploratory proposal diagnosis at the reference decision point. Counts pool GPT-5.4 and Gemma on identical inputs. Of 40 sampled scenarios, Codex codes 15 as having sufficient evidence for the reference target, 22 as insufficient, and three as uncertain. The rightmost column restricts to the sufficient-evidence cases. These are diagnostic labels, not revised benchmark scores or independently validated human annotations.

| Proposed outcome | All sampled cases | Sufficient evidence |
|---|---|---|
| Core outcome covered | 3 | 3 |
| Partial outcome | 14 | 9 |
| Different outcome | 36 | 11 |
| Silent | 27 | 7 |
| Uncertain | 0 | 0 |
| Decisions | 80 | 30 |
| Scenarios | 40 | 15 |

mate, and does not change any label, window, or score. In particular, an early match under PARE's scripted anchor need not be unsupported by its frozen application state.

### G.2 Systematic Proposal Diagnosis

To extend the illustrative comparison, we fix a coding protocol and sample before inspecting additional outputs. After excluding twelve scenarios previously selected by fixed hash within recovery strata (two recovered by both models, three by each model alone, and four by neither) from the 132 with observable decision points, we sample 40 of the remaining 120 by a seeded hash. Strata are defined by GPT-5.4's and Gemma's intervention decisions at the reference step: both intervene (16 of 26 sampled), only Gemma intervenes (20 of 90), only GPT-5.4 intervenes (one of one), and neither intervenes (three of three). All sampled cases are retained. The coding packet contains the actual supplied context, reference, and proposals, with model names and original verdicts withheld and output order randomized. This is an exploratory coding pass by Codex, not independent human validation; style can still reveal model identity.

We first code whether the supplied context identifies the core reference target, treating the reference itself as a target description rather than observed evidence. We then classify each decision as covering the core outcome, proposing a noncontradictory part of it, proposing a different outcome, remaining silent, or uncertain. These labels concern proposed contributions rather than execution, and each judgment retains an exact source quote and rationale. Table 14 shows that partial and different outcomes remain among cases coded as having sufficient evidence. Of the 36 different-outcome proposals, 11 address another event or goal visible in the input; that observation alone does not establish their usefulness. One proposal originally judged a match is coded as partial under this diagnostic rubric; the original score is retained.

The counts describe this stratified sample, not failure prevalence across PARE or full-episode recovery. The analysis files additionally report inverse-inclusion-weighted estimates for the 120-scenario frame, approximate sampling intervals, and worst-case bounds for unsampled items. Sampling intervals condition on these codes and exclude annotation uncertainty. An author and a second reviewer subsequently checked 15 utility assessments with AI judgments visible (Appendix H.3); this check does not validate the full outcome coding.

### G.3 Proposal Utility Beyond Reference Matching

We review all 53 emitted proposals from the same 40-scenario sample to ask whether the visible context supports a concrete benefit from each proposal, regardless of reference matching. The 27 silent decisions are not assigned utility labels. The working packet omits references, model names, previous codes, and matcher verdicts, and all annotations are frozen before joining the original verdicts. Codex has already inspected this sample, however, so this follow-up is neither independent nor fully blinded. A subsequent author check of 15 assessments is reported in Appendix H.3.

Table 15: Exploratory proposal-utility coding of all 53 emitted proposals in the fixed 40-scenario PARE sample. Columns retain the original matcher labels; counts do not estimate population usefulness or human acceptance.

| Assessment | Non-match | Match | Total |
|---|---|---|---|
| Supported benefit | 23 | 3 | 26 |
| Evidence limited | 13 | 1 | 14 |
| Contradicted or redundant | 6 | 0 | 6 |
| Uncertain | 7 | 0 | 7 |
| Total | 49 | 4 | 53 |

Table 16: PARE judge sensitivity on all 210 emitted proposals from two saved runs. Counts use the same 143 scenarios. Timely counts require the first match to be in-window. Both judges use the same rubric; neither is treated as human ground truth.

| Agent | Judge | Recovered | Timely | Matched proposals |
|---|---|---|---|---|
| GPT-5.4 | GPT-5.6-sol | 11 | 10 | 13 |
| GPT-5.4 | GPT-4.1 | 11 | 11 | 13 |
| GPT-5.6-sol | GPT-5.6-sol | 29 | 26 | 34 |
| GPT-5.6-sol | GPT-4.1 | 26 | 23 | 31 |

The rubric records grounding, concrete benefit, timing relative to the proposed contribution, visible redundancy, and agreement between the proposal text and action. A *supported benefit* requires supported material premises and timing, a concrete improvement, a consistent action, and no visible redundancy; partial progress and conditional offers can qualify. *Evidence limited* identifies a missing material premise, target, or benefit, while *contradicted or redundant* requires an observed conflict or an already satisfied goal. Competing reasonable interpretations remain *uncertain*. Ordinary details obtainable after confirmation do not alone invalidate an offer, but an invented target identity or conflicting action does. Each judgment retains exact context and proposal quotes, action arguments, a rationale, and missing evidence; automated quote checks establish traceability rather than semantic correctness.

Of 49 original non-matches, 23 receive supported-benefit codes, 13 are evidence limited, six are contradicted or redundant, and seven remain uncertain (Table 15). Across all proposals, supported-benefit codes include nine previously classified as partial outcomes and 14 as different outcomes. Examples include removing one apartment that violates a stated size constraint and creating a shared study-group chat after participants reply separately. By contrast, an offer to download an attachment is evidence limited when only an email subject is visible; adding another pair of headphones is redundant when both the user and a friend have already ordered them. A redundant action can still accompany an informative message, so that label does not imply every part of a proposal lacks value. Among the four original matches, three receive supported-benefit codes and one is evidence limited because its proposed note identifier is an update timestamp rather than an established identifier.

These sample-specific codes assess potential benefit, not population precision, interruption costs, or execution success. The frozen protocol, evidence ledger, and count-reproduction script accompany the analysis artifacts; benchmark scores remain unchanged.

### G.4 PARE Judge Sensitivity

To test whether the comparison between GPT-5.4 and GPT-5.6-sol depends on the latter also serving as judge, GPT-4.1 independently scores all 210 interventions emitted by these two saved runs. It receives the original judge messages verbatim, including the complete reference, proposed contribution, and rubric, without the agent identity or original verdict. Every intervention receives a valid judgment. The two judges agree on 203/210 decisions (96.7%): five original matches are rejected and two original non-matches are accepted by GPT-4.1.

Table 17: PARE at a common budget of 92 interventions. Expected counts under uniform thinning of each saved run, computed exactly. Original columns use all emitted interventions. This is an output-selection diagnostic, not a new agent policy.

| Model | Original interventions | Original counts Recovered | Timely | Expected at $K = 92$ Recovered | Timely |
|---|---|---|---|---|---|
| GPT-5.4 | 92 | 11 | 10 | 11.00 | 10.00 |
| GPT-5.6-sol | 118 | 29 | 26 | 23.18 | 20.63 |
| GPT-oss-120b | 421 | 12 | 9 | 2.79 | 2.14 |
| Qwen3.5-9B | 292 | 7 | 6 | 2.21 | 1.89 |
| Gemma-3-12B | 503 | 17 | 11 | 4.14 | 2.24 |

Recomputing scenario recovery under GPT-4.1 retains 11 recovered references for GPT-5.4 and reduces GPT-5.6-sol from 29 to 26, preserving their ordering (Table 16). The number of timely first matches changes from 10 to 11 and from 26 to 23, respectively. Thus both the counts and the timing of first recovery are sensitive to individual content judgments even when the aggregate ordering persists. This is a second-model sensitivity check on fixed outputs, not an adjudication of which judge is correct; the original main-table scores are retained.

### G.5 Matched Intervention Budget

To separate output frequency from reference coverage in the saved PARE runs, we uniformly retain exactly $K$ of each model's $N$ interventions without replacement. We use $K = 92$, the smallest intervention count among the five runs, and retain all 143 scenarios in the evaluation set. If a reference has $m$ matching interventions, its probability of being recovered is $1 - \binom{N-m}{K}/\binom{N}{K}$. Summing over references gives the expected recovery count. For timing, we sum the probability that the earliest retained match lies in the reference window; an early match that is removed no longer prevents a later match from being first. These expectations are exact for the saved outputs, not Monte Carlo estimates.

When each model retains the same number of interventions, GPT-5.6-sol has an expected 23.18 recovered references, compared with 11.00 for GPT-5.4 and 4.14 for Gemma (Table 17). Gemma's original 17-versus-11 recovery advantage over GPT-5.4 therefore does not persist under uniform thinning to the same number of interventions. This diagnostic changes output retention, not the agent's decision rule: it does not show how a model would behave under a prompted budget or establish that random selection is an effective deployed policy. Original judgments, decoding variability, and the self-judging caveat remain applicable.

### G.6 Event Representation Control

The main PARE configuration renders event descriptions or abbreviated arguments. We test this input choice by retaining each description and adding the complete source arguments of already-visible, non-oracle events. The transformation uses static parsing and does not execute scenario code, expose later observations, or consult the reference when choosing fields. It preserves the original distractors, event order, visibility cutoff, system prompt, application state, and knowledge pool. Source expressions such as `self.item_id` remain unresolved. The maximum addition is 1,650 characters per decision prompt.

Of the original 602 decision points, 206 real-event prompts change and 396 distractor-only prompts remain identical. We first evaluate both representations with Qwen3.5-9B and Gemma-3-12B, then extend the frozen comparison to GPT-5.4 and GPT-5.6-sol after seeing the open-model results. All four comparisons are reported. Both arms are freshly generated for every model: 1,648 decisions covering all 132 scenarios with observations. Within each model, the arms share a 4,096-token output limit and one schema-repair retry; the open models use temperature zero, while the hosted models retain their main-run default. Every decision parses successfully. GPT-5.6-sol scores every emitted intervention against the complete reference without arm or model labels, including its own outputs. Recovery and first-match timing are computed over the tested real-event points with the

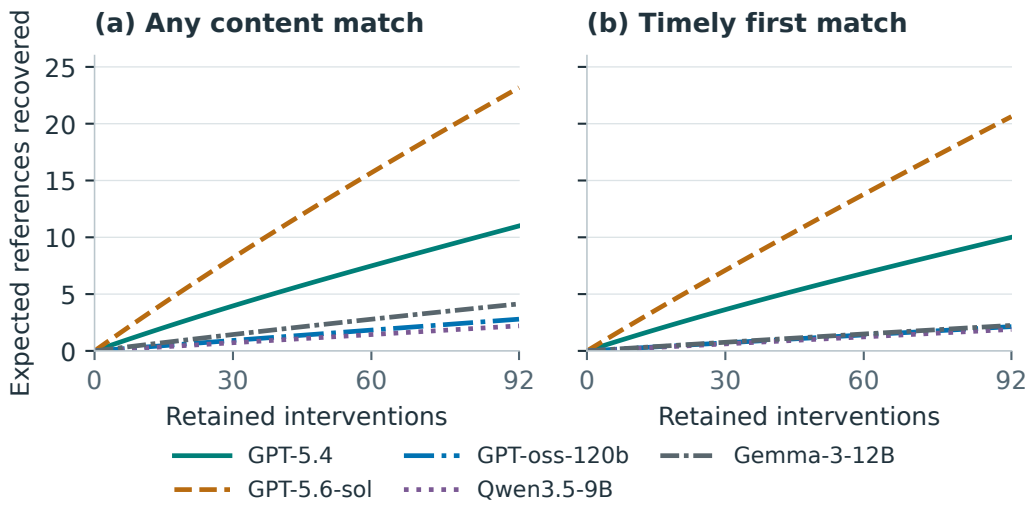


Figure 4: Expected PARE reference recovery when uniformly retaining a fixed number of each model's saved interventions. Timing is recomputed from the first retained match. Curves are exact expectations over subsets of fixed outputs; they do not represent new agent runs or confidence intervals.

Table 18: Paired event-representation control on 206 real-event decision points from 132 PARE scenarios. Both arms are freshly generated; unchanged distractor-only points are excluded. Timely counts first matches inside their windows. $E_K$ is expected recovery after uniformly retaining the smaller intervention count within each model pair ($K = 55, 84, 187, 200$, respectively). GPT-5.6-sol judges all outputs; * denotes its self-judged row.

| Model | Event input | Interventions | Matches | Recovered | Timely | $E_K$ |
|---|---|---|---|---|---|---|
| GPT-5.4 | Summary | 55 | 14 | 12 | 11 | 12.00 |
| GPT-5.4 | Full arguments | 82 | 34 | 29 | 26 | 20.22 |
| GPT-5.6-sol* | Summary | 84 | 28 | 24 | 21 | 24.00 |
| GPT-5.6-sol* | Full arguments | 121 | 43 | 42 | 42 | 29.37 |
| Qwen3.5-9B | Summary | 187 | 15 | 14 | 12 | 14.00 |
| Qwen3.5-9B | Full arguments | 195 | 26 | 25 | 24 | 24.01 |
| Gemma-3-12B | Summary | 200 | 16 | 14 | 11 | 14.00 |
| Gemma-3-12B | Full arguments | 200 | 24 | 21 | 19 | 21.00 |

original distractor-adjusted windows, so these results are not directly interchangeable with full-episode main-table scores.

Full arguments increase observed recovery in all four models (Table 18). GPT-5.4 gains 20 scenarios and loses three, while GPT-5.6-sol gains 22 and loses four. Their net changes are 12.9 and 13.6 percentage points, with paired scenario-bootstrap intervals of $[6.1, 19.7]$ and $[6.8, 21.2]$ from 20,000 resamples. Qwen and Gemma improve by 8.3 and 5.3 points, with intervals $[0.8, 15.9]$ and $[0.0, 11.4]$ (Figure 5). Exact paired tests with Bonferroni adjustment across the four models give $p = 0.0020$, $0.0021$, $0.173$, and $0.474$, respectively. These are exploratory comparisons conditional on one run per arm and fixed judgments; the intervals resample scenarios, not individual decisions.

Intervention counts also increase for three models. Uniform thinning within each pair to the smaller output count retains the recovery advantage in expectation: for example, GPT-5.4 yields 12.00 versus 20.22 recovered references at 55 interventions (Table 18). This compares the saved proposals at equal retained counts; the broader interpretation is discussed in Appendix B.

To check dependence on the scoring model, GPT-4.1 rejudges all 1,124 emitted proposals from the four-model control, using each original judge request verbatim. The judges agree on 1,071 proposals

Table 19: Reference turnover under expanded event arguments across 132 scenarios. Gain means recovered only with full arguments; loss means recovered only with summaries. Agreement columns count the same transition under both judges, not independently validated correctness.

| | Primary judge | | Second judge | | Both agree | |
|---|---|---|---|---|---|---|
| Model | Gain | Loss | Gain | Loss | Gain | Loss |
| GPT-5.4 | 20 | 3 | 13 | 4 | 12 | 3 |
| GPT-5.6-sol | 22 | 4 | 22 | 5 | 18 | 4 |
| Qwen3.5-9B | 18 | 7 | 18 | 4 | 17 | 2 |
| Gemma-3-12B | 11 | 4 | 8 | 3 | 8 | 1 |

(95.3%); GPT-4.1 rejects 37 original matches and accepts 16 original non-matches. Recovery still increases with full arguments for every model, but GPT-5.4's gain falls from 17 references to nine (**??**). Under this judge, Bonferroni-adjusted paired $p$-values are $0.196$, $0.0061$, $0.0174$, and $0.906$ for GPT-5.4, GPT-5.6-sol, Qwen, and Gemma. Thus the direction persists across judges, while effect sizes and which comparisons meet a significance threshold remain judge-dependent.

**Which references change?** We count which scenarios are recovered only with summaries and which only with full arguments, under each judge (Table 19). Every model has gains and losses accepted by both judges. The following examples illustrate these changes in the saved runs.

For `ride_receipt_expense_note_organization`, the summary exposes the expense-request subject, while full arguments also show that only hotel–convention-centre rides are reimbursable. With summaries, the two hosted models remain silent, Gemma offers an acknowledgement, and Qwen includes personal rides. With full arguments, all four propose an expense response accepted by both judges, using the eligible total of $113 in the proposal or action. The newly visible eligibility rule supplies task-relevant information for this proposal.

Conversely, for `cab_lost_item_recovery`, the relevant message text is already complete in the summary and is preserved in the expanded JSON representation. GPT-5.4 offers to contact lost-and-found under the summary but remains silent under full arguments; both judges retain this recovery loss. This example cannot be attributed to previously omitted message content. In `emergency_ride_for_stranded_friend`, the primary judge records gains for GPT-5.4 and Gemma and a loss for Qwen, whereas the second judge records none of those transitions. Its original rationales differ over whether the Premium ride tier and confirmation are essential. These cases illustrate both input/output changes and judge-sensitive scoring.

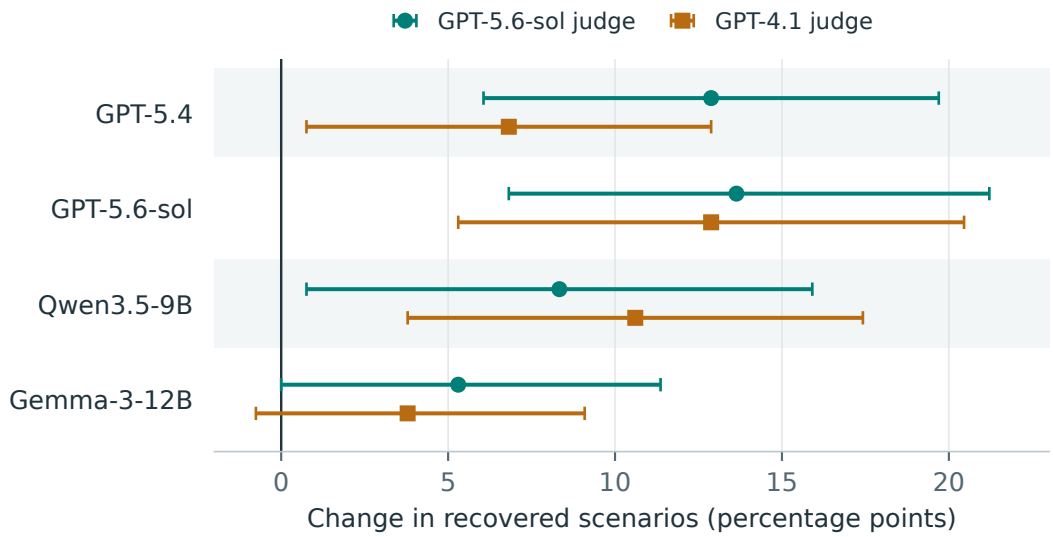


Figure 5: Change in PARE reference recovery when rendering full arguments of already-visible events instead of summaries, scored by two judges. Intervals show 95% paired scenario-bootstrap uncertainty over 132 scenarios, conditional on the saved outputs and each judge's labels. Hosted-model comparisons extend the protocol after the open-model results were observed; all four comparisons are shown.

## H Matcher and Window Validation

The SWE-chat scorer combines category-specific structural checks with an LLM judgment of whether the proposal preserves the gold contribution's concrete subject and substantive claim. The comparison is against the current instance's gold, not all 306 test items. An intervention can match several gold items; repeated interventions can match the same item, so precision is not capped by the number of distinct gold items divided by the number of interventions.

A second model pass reviewed 150 proposal–gold pairs without seeing the original verdict. It returned a determinate judgment on 144 pairs, agreeing on 123 (85.4%; Cohen's $\kappa = 0.677$); six pairs were unjudgeable. The original matcher accepted 17 pairs that the second pass rejected, with four disagreements in the opposite direction. This measures agreement between model judgments, not human accuracy or a statistical upper bound on recall.

For an earlier PARE run, re-judging 55 in-window proposals gave 92.7% agreement and $\kappa = 0.851$. These values describe that earlier validation sample rather than the complete-reference judgments in the main table. GPT-5.6-sol is also evaluated as an agent, so tables mark its rows to disclose self-judging. A shared judge model does not make the two domains' rubrics identical.

The secondary window-boundary review uses the archived windows and is also model-only. Of 60 sampled starts, the second pass labelled 19 as opening at the appropriate turn, 14 as opening before the supporting evidence, two as opening after it, and 25 as unjudgeable. Of 40 sampled ends, it judged five appropriate, 23 as still allowing a useful contribution afterward, and 12 unjudgeable. These disagreements do not measure the error rate of the later utility intervals; the controlled comparison in Appendix H.1 evaluates sensitivity to the window source.

### H.1 Utility-Window Alignment

The archived evaluator reads the preliminary transcript-only windows even though a later utility audit already supplies intervals for the retained gold. We align evaluation with that audit without generating new model outputs or content judgments. The loader requires a complete audit and a useful judgment with valid bounds for every selected gold item, rejecting missing or invalid intervals instead of substituting $[1, T]$.

Table 20: Window-source comparison on the same 100 SWE-chat test instances. Predictions and content judgments are fixed; instance coverage, precision, and recall are unchanged. Silence uses the decision points outside all windows for the indicated source.

| | In-window | | Silence | |
|---|---|---|---|---|
| Model | Archived | Utility | Archived | Utility |
| GPT-5.4 | 0.484 | 0.452 | 0.113 | 0.248 |
| GPT-oss-120b | 0.852 | 0.889 | 0.738 | 0.749 |
| GPT-5.6-sol | 0.632 | 0.579 | 0.654 | 0.718 |
| Qwen3.5-9B | 0.783 | 0.652 | 0.668 | 0.666 |
| Gemma-3-12B | 0.750 | 0.550 | 0.746 | 0.697 |

Table 21: Targeted SWE-chat sensitivity, retaining 100 instances and excluding three reference occurrences. Timing ranges additionally vary the 45 unresolved gold–point memberships remaining in the sampled review, after eight adjacent-point alternatives. They are outer sensitivity bounds, not confidence intervals.

| Model | Instance | Precision | Recall | In-window range |
|---|---|---|---|---|
| GPT-5.4 | 0.15 | 0.042 | 0.099 | 0.433–0.467 |
| GPT-oss-120b | 0.15 | 0.057 | 0.089 | 0.852–0.889 |
| GPT-5.6-sol | 0.12 | 0.054 | 0.063 | 0.579–0.579 |
| Qwen3.5-9B | 0.13 | 0.038 | 0.076 | 0.652–0.652 |
| Gemma-3-12B | 0.11 | 0.034 | 0.066 | 0.550–0.550 |

All 123 test items assigned fallback windows by the earlier audit have supporting knowledge-pool excerpts and are retained as useful by the later audit. Their utility intervals narrow 63 windows and positively support $[1, T]$ for the remaining 60, of which 41 belong to single-decision instances. Across all 306 test items, 183 intervals change and 190 end before the final decision point. The two audits differ in both evidence scope and utility criteria, so this comparison does not isolate the effect of adding knowledge-pool evidence. The intervals are conditional on the benchmark-supplied context, which permits retrospective anchor records, rather than strict historical availability.

Table 20 reuses complete saved content judgments from five runs. GPT-5.4's in-window rate changes from $15/31$ to $14/31$, while GPT-oss-120b's changes from $23/27$ to $24/27$; their content-recovery and timing ordering persists. The silence denominator changes from 485 to 1,027 of the 1,342 decision points, so changes in that rate reflect a different set of scored points rather than different agent behaviour. The GPT-4.1-family aggregates supplied by the authors are preserved separately because the corresponding per-decision artifacts have not yet been identified. The reported SWE-chat lane ablations also use these utility windows; their 81-instance subset and content judgments are unchanged. Existing training datasets retain their construction-time windows, so newly trained comparisons must record the window source explicitly.

### H.2 Targeted Adjudication

An exploratory agent pre-review of 34 reference–session pairs at 82 decision points raised 16 disagreements with the utility windows. Follow-up inspection supports eight adjacent-point alternatives, identifies one audit justification based on future evidence, and leaves seven points uncertain about incremental utility. Known unfinished work can still warrant a reminder, and completed work can inform a pending decision; neither prior mention nor completion alone establishes resolution. Four disputed semantic-match pairs are resolved using their category criteria and complete source comments, retaining the original matcher verdicts. In one case, an abbreviated reference omitted the second half of a documentation contradiction. This follow-up sees the original labels and is not independent validation.

For the future-evidence case, the retained window is $[12, 12]$, but all four attached supporting excerpts are live-session evidence visible only from decision 13 or later. We therefore report a targeted exclusion sensitivity for this occurrence together with the two explicitly requested occurrences in

Appendix D, retaining all 100 instances. GPT-5.4 loses one recovered item and one matched intervention: recall changes from $31/306$ to $30/303$, precision from $42/979$ to $41/979$, and instance coverage from 0.16 to 0.15, tying GPT-oss-120b. The other models' recovered counts are unchanged. None of the 16 disputed points coincides with a first match to its reference. Table 21 additionally bounds timing by independently varying the unresolved sampled gold–point memberships remaining after exclusion, with all other memberships and content judgments fixed. These outer bounds relax interval contiguity and leave the five models' timing ordering unchanged; they do not establish complete replacement windows or a dataset-wide annotation error rate. Canonical references and main-table scores remain unchanged.

### H.3 Human Annotation

One author and a second annotator each assessed the same 35 items: 15 PARE proposals and 20 SWE-chat contribution–decision-point pairs, comprising 10 contributions evaluated at two points each. Selection used fixed hash-based rules: PARE included five supported-benefit, four evidence-limited, three contradicted-or-redundant, and three uncertain assessments, with one proposal per scenario; SWE-chat covered all seven categories represented among contributions with at least two available review points. Both annotators received identical AI prelabels, rationales, and evidence, including displayed follow-up corrections, and selected agree, disagree, or unable to judge, with optional corrections. The second annotator did not see the author's answers and self-reported no project involvement, prior exposure to the cases, or additional assistance.

The author agreed with all 20 SWE-chat and all 15 PARE prelabels. The second annotator agreed with 18/20 SWE-chat prelabels and disagreed with two, both concerning the same contribution at successive sampled points. For PARE, the annotator agreed with 14/15 prelabels and selected unable to judge once. That proposal merely acknowledges a name-change announcement; its prelabel was already uncertain, and the annotator likewise questioned whether acknowledgement alone was enough. We retain this response as unable to judge. Agreement can confirm a positive, negative, or uncertain assessment; the resulting utility judgments are analysed separately below.

**Relevance disagreement and sensitivity.** The SWE-chat disagreement concerns an `origin/HEAD` default-branch test while the current discussion asks how branch-relevant checkpoints are selected. The reviewer marked it irrelevant at both sampled points, whereas the AI had marked relevance positive. We assess sensitivity by excluding the corresponding session–reference occurrence while retaining all 100 instances and saved judgments. No evaluated model recovered this item, so recovered counts, instance coverage, precision, and in-window rates remain unchanged. Recall changes only through the denominator, from 306 to 305; GPT-5.4 changes from $31/306$ (10.13%) to $31/305$ (10.16%). Silence changes slightly because the excluded window no longer contributes expected decision points. Removing the occurrence entirely is stronger than questioning its relevance at the two sampled points; canonical labels, windows, and main-table scores remain unchanged.

**Relation to reference matching and timing.** We join these human annotations to the original match labels and utility windows, counting each proposal or contribution–point pair once. A positive or negative assessment is reported as confirmed only when both annotators endorse it; uncertainty and disagreements without a replacement utility label remain unresolved. The PARE sample includes one matched proposal, confirmed to offer a supported benefit, and 14 non-matches. Of the non-matches, four have confirmed supported benefit, four are evidence limited, three are contradicted or redundant, and three remain unresolved (Table 22). These annotations establish concrete examples of useful proposals outside the reference set. With only one matched proposal and sampling stratified by prior utility codes, they do not estimate a population association between matching and usefulness.

Each SWE-chat contribution has one sampled point inside its window and one outside: four pairs cross a start and six cross an end. Human annotations confirm utility at five of the ten inside points; the other five remain unresolved. Outside the windows, three points retain utility, three no longer do, and four remain unresolved. Three pairs specifically confirm utility inside and resolution afterward. The three useful outside points comprise one before the start and two after the end; for one of these contributions, the inside point remains uncertain. Thus the sample supports some temporal

Table 22: Human annotation of 35 items: 15 PARE proposals and 20 SWE-chat contribution–decision-point pairs. Resolved assessments are confirmed by both annotators; unresolved cases retain uncertainty or disagreement. Each proposal or contribution–point pair is counted once.

| PARE proposals | Supported | Evidence limited | Conflict / redundant | Unresolved |
|---|---|---|---|---|
| Matched ($n = 1$) | 1 | 0 | 0 | 0 |
| Unmatched ($n = 14$) | 4 | 4 | 3 | 3 |

| **SWE-chat points** | **Useful** | **Not useful** | **Unresolved** |
|---|---|---|---|
| In-window ($n = 10$) | 5 | 0 | 5 |
| Outside ($n = 10$) | 3 | 3 | 4 |

changes in utility while also identifying useful adjacent opportunities excluded by the annotated bounds. These are judgments of utility at selected points, distinct from unprompted eligibility, and do not establish complete replacement windows or validate PARE's scripted timing anchors. Original annotations and benchmark scores are retained.

# I Evaluation Run Notes

**Missing outputs.** SWE-chat coverage denominators retain all 100 instances and 306 gold items, including timeouts and missing decisions. A missing decision yields no recovery credit. Precision is computed over produced interventions; silence follows the harness convention that a missing output is no intervention, so failures can increase it. GPT-5.6-sol's full-split rescoring contains 409 interventions, 22 matched interventions, and 19 recovered gold items across 12 instances, giving instance coverage 0.120, precision 0.054, and recall 0.062. Its archived windows give in-window and silence rates of 0.632 and 0.654; utility-window rescoring gives 0.579 and 0.718. This replaces the earlier completed-subset denominator.

**Runtime failures.** Locally served Qwen3.5-9B and Gemma-3-12B runs include context-limit failures. Those failures are retained in the main-table results, while the SWE-chat lane ablation explicitly restricts to the common non-overflow subset. Silence rates should therefore not be interpreted as pure policy choices without a separate failure-rate analysis. The available records do not establish variation over repeated training or inference seeds, and the reported comparisons are descriptive.

**PARE decision points and distractors.** The main evaluation contains 132 scenarios with observable events and 396 injected distractor steps; the other 11 scenarios remain in the denominator with zero recovery. Archived runs used 133 nonempty scenarios and 399 distractors, before scripted assistant events were removed (Appendix L). Distractor intervention rates concern injected steps; first-match timing concerns recovered references.

## I.1 Complete PARE Reference Scoring

The main evaluation scores every saved proposal against the full scripted proposal and first intended application action. This includes interventions before the gold trigger, so content precision and first-match timing use complete per-proposal judgments. The judge uses GPT-5.6-sol with the same matching rubric across models. Request hashes, source hashes, raw verdicts, and per-scenario first-hit records are retained for reproduction.

**Archived scoring correction.** Earlier runs used references with arguments truncated to 200 characters and incomplete proposal judgments. Rescoring their saved outputs with complete references corrected both issues together, so their individual effects cannot be isolated. The historical request counts and tables are retained in the reproducibility notes; current results use the shared, role-corrected observations described in Appendix L.

## J TAXONOMY-BASED PARE ANALYSIS

We classify all 143 complete scripted references using the thirteen fixed buckets in Table 2. GPT-5.6-sol receives bucket definitions, proposal text, and the intended application action, without scenario filenames, agent outputs, or performance scores. It selects one primary bucket according to the proposed outcome and may record secondary labels. Primary labels alone determine the groups in Table 26; secondary labels neither change the group denominators nor the evaluation targets. References outside the definitions receive an explicit unmapped label. The residual meeting bucket retains its facilitation and governance meaning rather than absorbing arbitrary application operations.

The classifier maps 102 references to nine buckets and leaves 41 unmapped. The latter include purchasing items and creating contacts. No reference has blocker surfacing, deadline surfacing, follow-up commitment, or the residual meeting bucket as its primary label, although blockers and deadlines may appear as secondary labels. Thus absence of a primary assignment does not establish absence of that information in a scenario. All 143 references have a label, with 123 marked high confidence and 20 medium by the classifier; these self-reports are not calibrated accuracy estimates. A second model agrees on 92 of 143 primary labels (Appendix M); no independent human validation has been performed.

## K UNCERTAINTY OF SWE-CHAT COMPARISONS

The test instances belong to eight connected components under the split's shared-session, workspace, and source-contribution constraints. Their sizes are 1, 3, 4, 5, 7, 15, 19, and 46. We draw eight components with replacement for each of 20,000 bootstrap replicates (seed 20260921), retaining all instances in each selected component and using the same draws for every model. Metrics are recomputed as pooled numerator-to-denominator ratios in each draw; intervals use the 2.5th and 97.5th percentiles.

Table 23: SWE-chat point estimates and 95% percentile intervals (percent), resampling the eight connected test components jointly across models. These conditional intervals do not measure judge error or training-seed variation; few unequal components limit their stability.

| Model | Instance | 95% interval | Recall | 95% interval |
|---|---|---|---|---|
| GPT-5.4 | 16.0 | [5.4, 25.4] | 10.1 | [6.2, 17.7] |
| GPT-oss-120b | 15.0 | [2.4, 24.3] | 8.8 | [1.7, 10.7] |
| GPT-5.6-sol | 12.0 | [4.3, 24.5] | 6.2 | [3.6, 14.4] |
| Qwen3.5-9B | 13.0 | [2.2, 19.8] | 7.5 | [1.6, 9.4] |
| Gemma-3-12B | 11.0 | [2.4, 19.5] | 6.5 | [1.7, 8.9] |

For GPT-5.4 minus GPT-oss-120b, the paired instance-coverage difference is 1.0 percentage point with interval $[-3.0, 9.5]$, and the recall difference is 1.3 points with interval $[-2.3, 11.0]$. Neither excludes zero. These intervals are conditional on the recorded predictions, judgments, and sampled components; they do not measure training-seed variation, annotation accuracy, or generalization to new repositories. Only eight components are available and one contains nearly half the instances, so the intervals themselves should be interpreted cautiously.

## L PARE REPLAY PROTOCOL

The replay uses source-pinned role annotations to expose user and environment events while withholding scripted assistant actions and replies accepting hidden proposals. The same role manifest governs the observation stream and distractor trigger bank. There are 206 observable events across 132 nonempty scenarios; three distractors per nonempty scenario yield 602 decision points. The remaining 11 scenarios receive zero recovery credit within the full 143-scenario denominator. Source hashes and role decisions are retained with the evaluation artifacts.

We freeze the rendered application state once and share it across models and ablations, avoiding differences from regenerated timestamps and identifiers. Each run uses the neutral agent prompt, stateless decisions, and complete-reference judgments for every intervention, including early proposals.

GPT-5.6-sol is the fixed judge. The main comparison covers GPT-5.4, GPT-5.6-sol, GPT-oss-120b, Qwen3.5-9B, and Gemma-3-12B. The local models use local serving endpoints and the same 4,096-token generation cap as the hosted runs. Per-scenario predictions, request hashes, returned model identifiers, and token usage are saved. Successful calls are reused on resume; missing judgments are not silently treated as negative labels. Gold action metadata pairs each proposal with its first intended application action, as described in Appendix M.

Qwen3.5-9B encounters 30 request timeouts under the 60-second client limit and 21 output-format failures across 602 decision points; these decisions receive no recovery credit. We retain valid predictions from the other decision points in the same scenarios rather than dropping those scenarios. One Qwen output contains the string `"false"` instead of a boolean; an offline accounting correction marks it as a format failure, removing one empty intervention and one distractor intervention. Recovery is unchanged. The original requests and reports are preserved, and no model or judge is rerun. The other four models have no API or output-format failures in these runs. All produced interventions have content judgments, so no missing judge verdict is treated as a mismatch. These counts distinguish deployment and formatting failures from failure to identify a reference contribution (Table 24).

Table 24: Failure accounting for PARE runs. Each model is queried at the same 602 decision points; 11 empty-stream scenarios also remain in the 143-scenario denominator. API errors and unparseable decisions receive no recovery credit. All produced interventions have judge verdicts.

| Model | Decisions | API errors | Format failures |
|---|---|---|---|
| GPT-5.4 | 602 | 0 | 0 |
| GPT-5.6-sol | 602 | 0 | 0 |
| GPT-oss-120b | 602 | 0 | 0 |
| Qwen3.5-9B | 602 | 30 | 21 |
| Gemma-3-12B | 602 | 0 | 0 |

The context ablation changes only the lanes shown to GPT-5.4 within the shared replay protocol. Both live-plus-workspace and all-lane configurations recover 11 references, with seven in common. This turnover illustrates why equal aggregate recovery does not imply identical contributions. These are single runs without a claim of statistically significant differences.

**Script-relative timing audit.** We inspect all 22 early content-matched interventions in the five main runs, covering 14 model–scenario pairs and 10 distinct scenarios (Table 27). This is a post-hoc agent inspection of saved inputs and outputs with reference labels visible, not independent human validation. Some central premises are already visible: the tripod cancellation is supported by the frozen inbox and order history, while adding a contact can be supported by the first message of a multi-message reference. Other cases remain unresolved, such as replacing draft attachments when final-named files exist but their promised delivery is not yet observed. Two coffee-machine proposals assert a stock-out while the visible product and cart still say available. Thus early matches include both evidence-supported opportunities and unsupported premises; neither a content match nor an early label establishes appropriate timing by itself.

Five of the 14 early-first pairs also match later inside the window. The reported first-match metric retains their early classification, rather than crediting a later repetition. An injected distractor can also overlap with the current state: an urgent-meeting message names the same time as a visible calendar errand, supporting a conflict proposal before the scenario's own reference point. These observations clarify the meanings of timing and distractor intervention rates without changing windows, predictions, or scores. They do not establish action executability, user acceptance, or unprompted eligibility, and do not generalize to the separately audited SWE-chat intervals.

## M Additional Annotation and Replay Audits

**Taxonomy agreement.** A second model, GPT-5.4, independently labels all 143 PARE references using the same category definitions and classification instructions, without the first labels or model scores. It agrees with GPT-5.6-sol on 92 primary labels (64.3%; Cohen's $\kappa = 0.599$). The disagree-

Table 25: GPT-5.4 context ablations on shared PARE event streams, frozen states, and scoring protocol. Each configuration is a separate run.

| Context | Recovered | Ref. precision ↑ | Ref. recall ↑ | Timely recall ↑ | In-window ↑ |
|---|---|---|---|---|---|
| Live | 3/143 | 0.025 | 0.021 | 0.021 | 1.000 |
| Live + workspace | 11/143 | 0.136 | 0.077 | 0.070 | 0.909 |
| All three lanes | 11/143 | 0.141 | 0.077 | 0.070 | 0.909 |

Table 26: Reference recovery in PARE runs, grouped by the original reference-only taxonomy labels. Entries are recovered / total scenarios; classification is fixed across model runs. Model names are abbreviated.

| Primary bucket | $N$ | 5.4 | 5.6-sol | oss-120b | Qwen | Gemma |
|---|---|---|---|---|---|---|
| Reaching out to a person | 23 | 4/23 | 5/23 | 2/23 | 0/23 | 2/23 |
| Creating a record or tracker | 19 | 0/19 | 2/19 | 0/19 | 1/19 | 1/19 |
| Cancelling an item | 17 | 0/17 | 6/17 | 0/17 | 1/17 | 1/17 |
| Booking a resource | 16 | 0/16 | 1/16 | 3/16 | 1/16 | 4/16 |
| Consolidating records | 9 | 1/9 | 2/9 | 2/9 | 0/9 | 1/9 |
| Sharing a resource | 7 | 0/7 | 2/7 | 0/7 | 0/7 | 0/7 |
| Rescheduling or deferring | 6 | 0/6 | 0/6 | 1/6 | 0/6 | 0/6 |
| Pulling in external context | 4 | 0/4 | 0/4 | 0/4 | 0/4 | 0/4 |
| Scheduling a follow-up | 1 | 0/1 | 0/1 | 0/1 | 0/1 | 0/1 |
| Outside the meeting taxonomy | 41 | 6/41 | 11/41 | 4/41 | 4/41 | 8/41 |

ments include whether to classify the offered operation or the contextual information motivating it. A separate pass with explicit tie-breaking instructions changes 14 of the original labels. We retain the primary assignments in Table 26 and preserve all three label sets in the analysis artifacts. Agreement between models does not establish annotation accuracy.

Reassigning the fixed corrected-run outcomes under these label sets preserves the illustrative cancellation-versus-record-creation contrast for GPT-5.6-sol: $6/17$ versus $2/19$ under the primary labels, $7/17$ versus $2/20$ under the second model, and $6/17$ versus $2/16$ under the clarified rubric (Table 28). Other distributions vary substantially: 41, 12, and 45 references, respectively, are unmapped. Thus the example contrast persists, but taxonomy coverage and category sizes depend on classification choices. All five models' regrouped counts are included in the numerical supplement; overall benchmark scores do not change.

**Reference consistency.** A model-only check compares each scripted proposal with its extracted application action, marking 126 consistent, nine partial, and eight contradictory. These flags are not validated error counts. Inspection of flagged scripts finds false alarms: an apartment comparison legitimately reads ride history to estimate commuting cost, and a note update changes its title and body in separate calls. The extraction policy also selects a single action from a longer sequence. We therefore retain all references and publish the flagged cases for review rather than treating these judgments as automatic exclusions.

Table 27: Census of early content matches in the corrected PARE main runs. Early first counts model–scenario pairs; early matches counts interventions, including repeats. Later match counts early-first pairs that also match inside the window. Published timing uses the first match and remains unchanged.

| Model | Recovered | Early matches | Early first | Later match |
|---|---|---|---|---|
| GPT-5.4 | 11 | 1 | 1 | 0 |
| GPT-5.6-sol | 29 | 5 | 3 | 3 |
| GPT-oss-120b | 12 | 3 | 3 | 0 |
| Qwen3.5-9B | 7 | 1 | 1 | 0 |
| Gemma-3-12B | 17 | 12 | 6 | 2 |

Table 28: GPT-5.6-sol recovery in the corrected PARE run under three existing reference classifications. Cells are recovered / total in each bucket. The primary and second-model classifiers use the same instructions; the clarified-rubric pass additionally specifies tie-breaking rules. Overall recovery remains 29/143.

| Bucket | Primary | Second model | Clarified rubric |
|---|---|---|---|
| Reaching out to a person | 5/23 | 6/25 | 5/24 |
| Creating a record or tracker | 2/19 | 2/20 | 2/16 |
| Cancelling an item | 6/17 | 7/17 | 6/17 |
| Booking a resource | 1/16 | 5/21 | 1/14 |
| Consolidating records | 2/9 | 1/4 | 2/7 |
| Sharing a resource | 2/7 | 3/8 | 3/7 |
| Rescheduling or deferring | 0/6 | 0/6 | 0/6 |
| Pulling in external context | 0/4 | 2/13 | 0/6 |
| Scheduling a follow-up | 0/1 | 0/1 | 0/1 |
| Surfacing a blocker | — | 1/5 | — |
| Surfacing a deadline | — | 1/4 | — |
| Committing to a follow-up | — | 0/7 | — |
| Outside the meeting taxonomy | 11/41 | 1/12 | 10/45 |