BEYOND REACTIVITY: MEASURING PROACTIVE PROBLEM SOLVING IN LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-based agents are increasingly moving towards proactivity: rather than awaiting instruction, they exercise agency to anticipate user needs and solve them autonomously. However, evaluating proactivity is challenging; current benchmarks are constrained to localized context, limiting their ability to test reasoning across sources and longer time horizons. To address this gap, we present PROBE (**P**roactive **R**esolution **o**f **B**ottl**e**necks). PROBE decomposes proactivity as a pipeline of three core capabilities: (1) searching for unspecified issues, (2) identifying specific bottlenecks, and (3) executing appropriate resolutions. We apply PROBE to evaluate leading LLMs and popular agentic frameworks, showing that even state-of-the-art models struggle to solve this benchmark. Computing our consistent measurements across frontier LLMs and agents, we find that the best end-to-end performance of 40% is achieved by both GPT-5 and Claude Opus-4.1. Additionally, we demonstrate the relative capabilities of each model and analyze mutual failure modes. Our results highlight the current limitations of autonomous action in agentic systems, and show promising future directions.

1 INTRODUCTION

Agentic systems built on Large Language Models (LLMs) have made immense progress, delivering practical value across several real-world applications including coding (Yang et al., 2024; Agashe et al., 2025), computer use (Song et al., 2024), web navigation (Zheng et al., 2024; Zhang et al., 2025), and healthcare (Kim et al., 2024; Sellergren et al., 2025). Despite significant progress, the majority of the agentic systems today are *reactive* - they expect explicit instruction from a user prior to attempting a task (Yao et al., 2023). To transcend their function as tools, agents need to be *proactive*: anticipating user needs from continuous observation, suggesting candidate tasks to address these needs, and executing these tasks reliably.

Prior studies on proactive agents have explored agent proactivity in interacting with physical environments (Zhang et al., 2023), asking follow-up questions (Zhang et al., 2024) and perceiving immediate needs from a personalized environment Lu et al. (2024a); Yang et al. (2025a). However, existing approaches compress evaluation into narrow, immediate temporal context, failing to capture insights that emerge only through longer-term analysis. For instance, proactive agents that look only at current context would not detect and take an appropriate action for a missed deadline from the past (as shown in figure 1). To this end, we operationalize proactivity as a three part construct. Given a set of priorities and a personalized user datastore, agents **search** across documents for user-relevant issues, **identify** the most pertinent ones (which we term *bottlenecks*), and **resolve** said issues by executing appropriate actions.

Constructing a real-world benchmark for proactivity is difficult since collecting long time horizon, multi-document user data raises privacy concerns and creates significant annotation overhead. Building on previous successes in the generation of synthetic datasets NadÇŐÅ§ et al. (2025); Long et al. (2024); Butt et al. (2024), we construct a data generation agent to build our benchmark (we describe this in section 2). The resulting PROBE benchmark comprises of 1,000 diverse samples that challenge AI systems to proactively identify and resolve critical bottlenecks hidden within realistic workplace datastores (see figure1). Our evaluation reveals a striking capability gap: even state-of-the-art LLMs and specialized agentic frameworks achieve no more than 40% success on this end-to-end task, highlighting the substantial challenges that remain in developing truly proactive AI

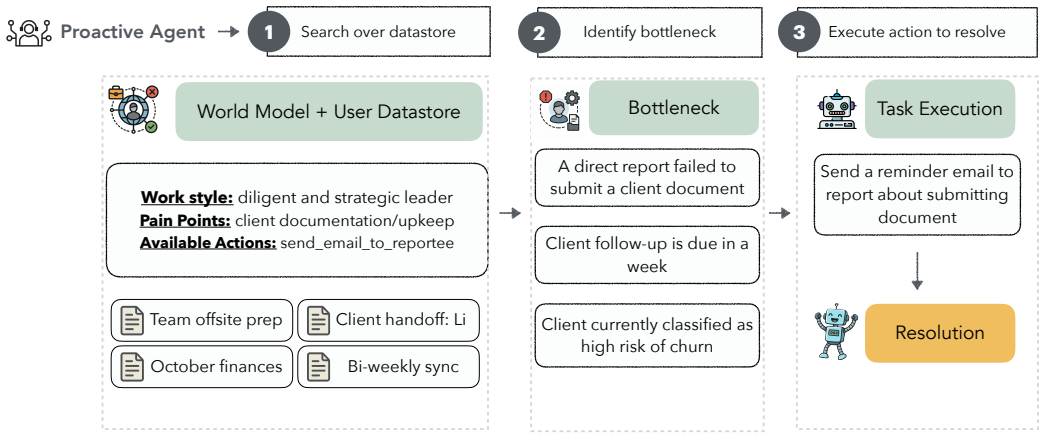

Figure 1: An end to end depiction of the PROBE task setup. The model (or agent) needs to use the world model to (i) search over user datastore, (ii) identify the botteneck and finally (iii) identify the action to be executed. The model is evaluated across all tasks in this pipeline.

systems. From a model perspective, PROBE establishes a joint evaluation protocol for LLMs and agents, as shown in figure 1. In summary, our contribution in this paper is threefold:

- We introduce PROBE - 1000 test samples that systematically evaluates proactive capabilities in AI systems through a unified framework, addressing a critical need for a realistic proactive benchmark.
- We conduct comprehensive evaluations across frontier closed-source and open-source models alongside leading agentic frameworks, revealing a fundamental capability ceiling: even the most advanced models achieve only 40% success on our end-to-end task.
- We present an in-depth analysis of common failure modes that uncovers the specific challenges associated with our benchmark and surfaces opportunities for future work.

2 Methodology

We create a data generation agent to orchestrate our end-to-end workflow (described in figure 2). Starting from real user personas, we build comprehensive world models that capture each simulated user's environment, goals, and constraints. These world models drive the creation of a datastore filled with synthetic documents that mirror a real workplace scenario. We then strategically inject bottlenecks into select documentsâĂŤhidden obstacles that inhibit users from achieving world-model-defined goals. For each bottleneck, our pipeline generates multiple candidate actions, with exactly one resolving bottleneck. This setup forces successful agents to demonstrate true proactivity: they must discover the bottleneck through exploration and identify the right fix among several plausible options. The following sections detail our problem formulation and pipeline components.

2.1 Problem Definition

Setup and notation: Consider a world-model W of a user, let D be a finite "universal" set of documents that constitutes the user's datastore, which is a collection of all of user's accessible documents, and let $b \in B$ denote a fixed *bottleneck*. We define a bottleneck as an issue that is critically important to the individual, actionable, and identifiable through a finite set of documents. For a given b, the rest of the documents that do not pertain to this bottleneck are considered *distractors* with respect to the bottleneck.

We define the binary predicate

$$f(d) : \forall d \in D \to \{0,1\} \text{ evaluated under } W, \qquad f(d) = \begin{cases} 1 & \text{if } d \textit{ conveys} \text{ the bottleneck } b, \\ 0 & \text{otherwise.} \end{cases}$$

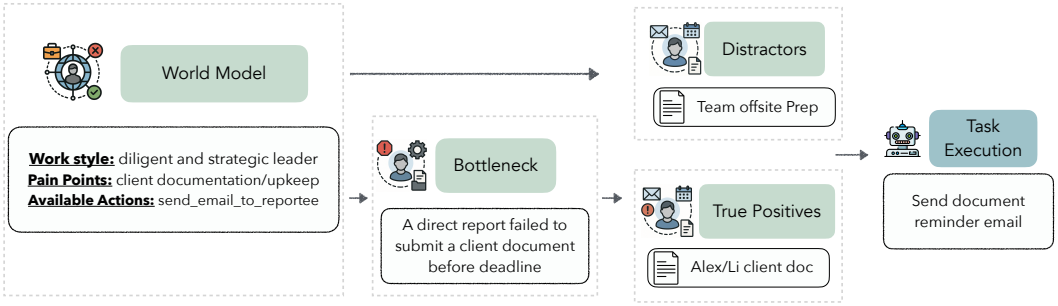

Figure 2: End-to-end proactive benchmark generation pipeline. A synthetic world model (W) is synthesized in reference to a linkedin user profile, followed by generating a bottleneck (b) to resolve. True Positives ($T \subseteq D$) and distractors (other documents unrelated to the bottleneck, $K = D \setminus T$) are then constructed to frame the bottleneck prediction task. Finally, task selection function calls with parameters ($\mathcal{P} = \{P_a\}_{a \in A}$ for action $a \in A$) for resolving the bottleneck are generated, of which only a single execution actually resolves the bottleneck.

A document $d \in D$ is marked as a *true positive* (w.r.t. b) iff $f(d) = 1$, and a *distractor* iff $f(d) = 0$. For the current benchmark, we make the simplifying assumption that a true positive in a single datapoint contains evidence of only a single bottleneck.

Sample generation: Each instance of the benchmark is a tuple $S = (T, K, A, \mathcal{P}, b)$ where

- $T \subseteq D$ is the set of true positives for the sample with $|T| = t$,
- $K = D \setminus T$ is the set of distractors with $|K| = k$,
- A is a finite set of available actions,
- $\mathcal{P} = \{P_a\}_{a \in A}$ assigns to each action $a \in A$ a parameter space P_a,
- b is the bottleneck associated with this sample.

We make the assumption that the true-positive set T is unique to this sample (i.e. different samples may not reuse the same T for simplicity). The observed document set presented to the agent is $D := T \cup K$.

2.2 Proactive Task Setup

An agent (LLM or agentic framework) receives an input D and a set of (a, P_a) tuples and must produce the tuple of outputs $\hat{O} = (\widehat{T},\ \hat{b},\ \hat{a},\ \hat{p})$, where

- $\widehat{T} \subseteq D$ is the agent's predicted set of true positives,
- $\hat{b}$ is the agent's prediction for the bottleneck,
- $\hat{a} \in A$ is the selected action,
- $\hat{p} \in P_{\hat{a}}$ are the selected parameters for $\hat{a}$.

2.3 Data Generation Setup

We design a robust data generation pipeline that scales in both context size and difficulty. Our pipeline starts from a sample of real-world professional profiles and constructs synthetic workplace scenarios as world-model. Figure 2 illustrates the complete pipeline, which comprises four key components[1]:

World Model Construction: We leverage the dataset constructed by Ayoobi et al. (2023) to extract basic persona from real-world LinkedIn profiles as the starting point. Each persona captures

[1] all prompts for individual data generation modules are shown in appendix B

high-level professional information including current workplace, role description, and a professional summary.

From these personas, we synthetically construct comprehensive world model that encode:

- Professional relationships, organizational hierarchies, including colleagues and their roles in relation to the user's persona
- Work patterns and communication styles at the individual level
- Available action space A with corresponding parameter spaces $\mathcal{P}$
- Potential pain points and operational constraints

For instance, given a senior account manager with 20 years of client-facing experience as shown in figure 2, the world model might identify "client documentation upkeep" as a pain point, while also modeling specific client relationships and their respective engagement contexts.

Bottleneck Generation: Using the contextualized world model, we generate bottleneck b: a persona-relevant, actionable user-need that satisfies our formal definition (see Section 2). Each bottleneck b is designed to be identifiable through evidence T in the document set D and resolvable through exactly one action $a \in A$.

User Datastore: For each sample S, we construct the document set $D = T \cup K$ where **True positives** T are documents where $f(d) = 1$, collectively provide sufficient evidence to identify bottleneck b and **Distractors** K are documents where $f(d) = 0$, introducing realistic noise w.r.t the bottleneck. In our current datastore setup, all the generated documents are restricted to emails, calendar events and text documents, as exemplified in Figures 1 and 2.

To mirror real-world complexity, we implement two key design principles: (i) **Evidence distribution**: We often distribute evidence for b across multiple documents in T, requiring agents to synthesize information from $|T| = t$ sources and (ii) **Contextual noise**: We generate $|K| = k$ distractor documents of comparable length and professional relevance, ensuring that bottleneck identification requires careful analysis rather than superficial pattern matching.

Task Execution : Finally, we construct the action set A and parameter space for each action a, defined as $\mathcal{P} = \{P_a\}_{a \in A}$ such that:

- Exactly one action $a^* \in A$ effectively resolves bottleneck b
- Each action a has a set of parameters $p \in P_a$ be specified to resolve the bottleneck.
- Alternative actions represent potentially plausible but suboptimal interventions

For example, given a bottleneck about missing documentation, the optimal action a^* might be *"send reminder email to direct report"* with parameters specifying the recipient, urgency level, and document details. The action set A may include plausible alternatives such as "escalate to management" or "rewrite document", forcing agents to reason about the most effective intervention. All actions available for a bottleneck are populated as "available actions" in the user's world model.

2.4 PROBE - BENCHMARK FOR PROACTIVITY EVALUATION

We use the setup outlined above to generate the final dataset using GPT-4.1 as our primary model for generation. We generate a total of 1000 datapoints generated from about 235 unique personas (described in 2.3). The full dataset stats are shown in Table 1[2].

	min	max	mean	std
Tokens	96294	122098	107,640.5	4,676.5
Actions	24	27	25.27	0.51
Docs	70	81	79.3	3.7

Table 1: Dataset statistics. We show the statistics across number of tokens, number of action and number of documents (true positives + distractors) for each datapoint in the dataset.

To ensure the quality of the dataset, we adapt a modified appraoch from Jiang et al. (2025) by performed multiple rounds of filtering via an adversarial agent (GPT-5) on a small sample set (of 5 datapoints). After each round of

[2]token counts measured using tiktoken `https://platform.openai.com/tokenizer`

sample generation, the adversarial agent was tasked to identify all potential artifacts (the agent was explcitly asked to exploit any pattern it can find) in the sample data, and solve the samples of the benchmark based on the artifacts first and use its own reasoning only when it couldn't solve it using the artifacts. The final set of datapoints were generated only once no sample from our pipeline was solvable with these artifacts alone .

Human Evaluation: To establish human performance benchmarks and validate task difficulty, we conducted a 4-hour annotation study with three annotators, all holding at least a master's degree. Each annotator received identical instructions to those given to LLM systems and completed as many samples as possible within the specified time limit. Additionally, the annotators were instructed to judge if samples were realistic and feasible. To assess this, we asked the following questions: (i) Were the all documents you read realistic documents that you may see in a real workplace? and (ii) Were the actions you read realistic actions that you could see being used to resolve bottlenecks in a real workplace setting?

Annotator ID	Search F1 %	Bottleneck Ident -tification Score (%)	Task Selection Score (%)	Entries Annotated	Samples hour	Realistic Artifacts
1	26.79	0.00	0.00	13	3.25	✓
2	45.24	0.00	14.90	7	1.75	✓
3	20.37	0.00	8.33	6	1.50	✓
Avg	30.28	0.00	5.93	8.67	2.17	✓

Table 2: Annotation numbers showing that humans could not accomplish our task successfully. Annotators on average retrieved 30%, never identified the bottleneck, and selected the correct task at $\sim 2\%$, just above random chance, labeling about 2 samples per hour.

Across 12 total annotator-hours, only 26 samples were successfully completed at an average throughput of 2.17 samples per hour per annotator. The annotation experiment showed the substantial cognitive load required for bottleneck identification across multiple documents, and the time-intensive nature of synthesizing evidence and selecting appropriate actions[3]. For the annotators who answered "yes" to the two questions on judging artifacts, we record a green check mark in the "Realistic Artifacts" column of our results table. The annotation results for all three annotators shown in table 2. All human annotations were evaluated using the same metrics setup as described in section 3.

3 Evaluation

3.1 Metrics

Search: This evaluation measures how well an agent retrieves the relevant documents that are required for identifying the bottleneck precisely. We measure the agent's retrieved document $\widehat{T}$ against the gold bottlenecks T using standard precision, recall and F1 metrics.

Bottleneck Identification: Since this task uses a natural language output, we use the LLM-as-a-judge (Zheng et al., 2023) framework to evaluate whether the LLM identified the bottleneck correctly. We split this evaluation into two subtasks: (i) identifying essential details (who is the blocker, what is the task, root cause,..) and (ii) identifying non-essential details (system/tool names, processes to follow, scope of impact, ..). The scoring rubric is as follows:

$$\text{Score} = \begin{cases} 1.0, & \text{if all essential and all non-essential details are accurate,} \\ 0.5, & \text{if all essential details are accurate but some non-essential details are incorrect,} \\ 0.0, & \text{if any essential detail is wrong or missing.} \end{cases}$$

Task Execution: Our scoring combines exact-match accuracy for the gold action label (assigning 0 if incorrect) with LLM-as-a-judge evaluation of parameter quality when the action is correctly

[3]Annotators found no sample to be unrealistic, and gave feeback that the task was very challenging

identified. The rubric follows:

$$\text{Score} = \begin{cases} 1.0, & \text{if action predicted is correct and all critical parameters are present,} \\ 0.5, & \text{if action predicted is correct and most critical parameters are present,} \\ 0.0, & \text{action wrong or if critical parameters are missing.} \end{cases}$$

We provide the prompts for both the llm-as-a-judge prompts in appendix B.6.

3.2 LLM-AS-A-JUDGE

To validate our use of LLM-as-a-judge, we conducted a measurement study with 50 GPT 4.1 prediction-output pairs. Two human annotators independently evaluated bottleneck identification and parameter judgments for each sample. We achieved 84% inter-annotator agreement and 80% human-LLM agreement across 100 total annotations, supporting our decision to use LLM-based scoring.

3.3 BASELINES

Models: We evaluate our benchmark against several frontier closed source models including OpenAI GPT-5(OpenAI, 2025), GPT-5-mini(OpenAI, 2025c), GPT-4.1(OpenAI, 2025a), GPT-4.1-mini(OpenAI, 2025b), Claude 4.1 Opus(Anthropic, 2025b), Claude 4 Sonnet(Anthropic, 2025a) and the best-performing open-source models including Kimi-K2(Team et al., 2025) and DeepSeek- R1(DeepSeek-AI et al., 2025). Among other open-source models, we test OpenAI GPT OSS(OpenAI, 2025) at both 120B and 20B scales[4].

Agentic Frameworks: We evaluate three leading agentic frameworks: ReACT (Yao et al., 2023), Reflexion-Agent (Shinn et al., 2023), and ReWOO (Xu et al., 2024). Since these frameworks target different problem domains, we adapted each for bottleneck resolution. Our modifications include workflow-specific prompts, structured outputs, and two retrieval toolsâĂŤembedding-based search and SQL queries. Each framework takes a distinct approach. ReACT cycles between reasoning and retrieval, progressively building context until it converges on an action. Reflexion learns from its failures: it runs multiple trials of document retrieval, analysis, and action selection, using LLM-based reflection to improve after each unsuccessful attempt. ReWOO, in contrast, precoordinates the process. After constructing a structured plan, it dispatches specialized workers for search and reasoning tasks, then synthesizes their findings to pinpoint bottlenecks and select interventions. We did not include any pre-existing proactive agent frameworks (Lu et al., 2024b; Yang et al., 2025b) as baselines, as current systems are designed for specialized domains (conversational agents, UI navigation, embodied robotics). Resultantly, the distinct input modalities and task objectives of these frameworks do not trivially transfer to our workflow. All agentic frameworks use GPT-5-mini as the underlying base model. We provide more details in appendix A.

3.4 MODEL COMPARISONS

Frontier Models Pull Ahead: The gap between top models in our benchmark (GPT-5, GPT-4.1, Claude Opus, Claude Sonnet) and the rest of the models is significant. GPT-4.1-mini reaches 0.42 on Bottleneck Identification but only 0.20 on Task Execution, achieving just half the success rate of the best performing models. The other models fare poorly in retrieval and cascade these errors downstream (e.g., GPT-OSS-120b: 0.13 F1 Search, 0.11 Task Execution), underscoring the difficulty of end-to-end bottleneck resolution without strong evidence acquisition.

Frontier models are stronger across the pipeline, but not uniformly: GPT-5 achieves the best search performance with an F1 of 0.65 and the highest task execution score of 0.40, indicating stronger end-to-end capacity to find the right documents and translate diagnosis into an actionable plan. Claude Opus 4.1 and Claude Sonnet 4 achieve the best bottleneck identification score of 0.43 while having a slightly lower search score. This suggests that Claude models have a slight advantage in reasoning capabilities for this task, which can offset potential short-comings in search. From the table of results 3, we also observe that different frontier models have different strengths (while GPT-5 is better at search, it is behind Claude Opus and Sonnet in bottleneck identification), and to get an

[4]We could not test gemini-2.5 due to rate-limiting issues

	Search			Bottleneck Identification	Task Execution
	P	R	F1	Score	Score
GPT-5	**0.73**	**0.59**	**0.65**	0.42	**0.40**
Claude Opus 4.1	0.68	0.41	0.51	**0.43**	**0.40**
Claude Sonnet 4	0.66	0.37	0.47	**0.43**	0.36
GPT-4.1	0.60	0.38	0.46	0.42	0.38
GPT-4.1-mini	0.18	0.20	0.19	0.42	0.20
Deepseek-R1	0.49	0.23	0.29	0.04	0.19
Kimi K-2	0.20	0.17	0.18	0.40	0.18
GPT-OSS-120b	0.27	0.10	0.13	0.35	0.11
GPT-OSS-20b	0.05	0.03	0.04	0.26	0.05

Table 3: Comparative results across several frontier closed and open source models. GPT-5 and Claude Opus-4.1 show the best performance compared to the rest. We also show evaluation across search, bottleneck identification and task execution to show the relative strengths of weaknesses across models. Note: We found GPT-5-mini performance to be close to GPT-4-mini performance across the board, hence removed for brevity.

	Search			Bottleneck Identification	Task Execution
	P	R	F1	Score	Score
ReACT(Yao et al., 2023)	0.08	0.37	0.12	0.02	0.06
Reflexion(Shinn et al., 2023)	0.18	0.11	0.13	0.02	0.05
ReWoo(Xu et al., 2024)	0.27	0.24	0.25	0.01	0.11

Table 4: We show comparison across multiple agent frameworks. In our initial experiments, they significantly lag behind using LLMs out-of-the-box for this task.

overall better performance, models need to strengthen capabilities across all search, identification and task execution. We found the results of Deepseek-R1 to be anomalous in terms of bottleneck identification performance, while performing well in rest of the metrics. On deeper inspection, we found that Deepseek-R1 consistently used generic descriptions of bottlenecks instead of specific details, leading to reduced performance in bottleneck identification.

Retrieval remains challenging: Our analysis reveals a clear performance hierarchy among models. Frontier models (GPT and Claude series) significantly outperform others (Kimi K-2, Deepseek-R1, GPT-OSS series), with all models showing higher precision than recallâĂŤfor example, GPT-5 achieves 0.73 precision but only 0.59 recall. This pattern suggests models retrieve conservatively, struggling to some extent to retrieve all relevant pieces of information needed to identify the bottleneck. This is especially pronounced in smaller Language Models, who seem to massively under-retrieve relevant documents and resultantly struggle with any sort of task selection.

Shortcutting helps overcome search difficulties, but not much: Among the top-performing models (GPT-5, GPT-4.1, Claude Opus, Claude Sonnet), some compensate for weaker retrieval with stronger free-form reasoning during Bottleneck Identification. This yields competitive identification scores without a corresponding improvement in task execution. The gap highlights that being “right for the wrong reasons" does not translate into executable solutions. As the search space grows, this effect will degrade, reinforcing the need for faithful evidence use. We believe that the remaining head-room in this task will be based on faithful evidence use to identify bottlenecks and then resolve them correctly. The best bottleneck identification score reaches only 0.43, while the best task execution score is just 0.40. These low performance ceilings reveal significant gaps in current systems' ability to translate diagnoses into actionable solutions with complete parameters, particularly when retrieval is imperfect.

3.5 AGENT FRAMEWORKS COMPARISON

We evaluated agentic baselines using a constrained setup where agents were provided with a SQL store and embedding-based semantic search for document retrieval. Our rubric enforces three metrics across the pipeline: retrieval, bottleneck identification, and task execution. Across all tested agents, retrieval F1 scores ranged from 0.12 to 0.25, substantially below the frontier models in Table 4. This weak retrieval performance cascaded through the pipeline, resulting in Bottleneck Identification and Task Execution scores of ≤ 0.11.

Even when agents generated plausible reasoning steps, the limited evidence prevented accurate diagnosis and parameter-complete actions. These results partly reflect the mismatch between our experimental constraints and typical agentic frameworks, which usually leverage web search, APIs, and environment interaction. Our benchmark's restriction to SQL and semantic search limits the "act" channel that many agent architectures rely on for iterative knowledge acquisition.

4 ERROR ANALYSIS

For this analysis, we use all failure cases of each model across the dataset. To understand where models struggle most, we analyze failure modes across three hierarchical categories: bottleneck identification, task selection, and parameter specification for the task that was selected (explained in metrics under section 3).

Failure Mode	**Claude Opus**	**Claude Sonnet**	**GPT-4.1**	**GPT-5**	**Kimi K-2**
Identification Failures (% of identification errors)					
Incorrect root cause[5]	64.6%	70.6%	76.8%	72.1%	84.8%
Person attribution error	46.9%	57.9%	61.3%	60.8%	78.0%
Missing/wrong deadline	53.4%	46.7%	43.5%	45.3%	35.0%
Function Selection Failures (% of action errors | identification success)					
Wrong function selected	9.7%	9.5%	9.2%	10.6%	10.9%
Parameter Failures (% of action errors | function success)					
Critical parameters missing	66.2%	75.4%	79.5%	65.2%	71.7%
Incorrectly filled parameters	45.4%	36.1%	35.3%	45.6%	44.3%

Table 5: Failure mode breakdown across frontier models. Root cause identification emerges as the dominant failure mode, while function selection shows consistent competence across models.

Root Cause Identification remains the primary challenge: Incorrect root cause identification dominates across all models, averaging 73.8% of identification failures. This represents the single largest systematic weakness, with even the best-performing Claude Opus failing at root cause analysis in nearly two-thirds of identification errors. This suggests potential avenues to improve reasoning capabilities tailored to our proactivity task.

Interpersonal reasoning: Interpersonal Reasoning - ability of a model to reason about the people who are involved in a bottleneck, remains challenging for models. All models struggle consistently with interpersonal dynamics (46.9%-78.0% failure rates), regardless of their performance on root cause identification. Even Claude Opus, the highest performing reasoning model in our benchmark, fails at interpersonal identification in nearly half of cases where it fails, suggesting that understanding workplace relationships might require additional capabilities.

Action Selection and Parameter Prediction for actions : Action selection and selection of parameters for the action remains independently challenging. GPT-5 and Claude Opus achieve better parameter coverage but higher error rates (45.6% and 45.4% incorrect), while GPT-4.1 and Claude Sonnet show the inverse pattern; more accurate specification (35.3% and 36.1% incorrect) but higher

[5]Incorrect root causes can signify either locating the possible cause and misidentifying the bottleneck or not identifying the possible cause at all.

miss rates. Current models cannot simultaneously achieve higher coverage and precise parameter specification within complex workplace scenarios as generated by this benchmark.

5 Related Work

Reactive vs. Proactive Agents: Most LLM-based agent research has focused on reactive systems that respond to explicit user instructions. Key advances include planning approaches like ReAct (Yao et al., 2023), tool integration via Toolformer(Schick et al., 2023), and self-reflection mechanisms such as Reflexion(Shinn et al., 2023). While these expand agentic capabilities, they remain fundamentally reactive - dependent on explicit requests without capacity to anticipate user needs.

Emerging proactive agents face the challenge of anticipating latent user goals from partial observations and executing actions without explicit instruction. Recent work explores intent inference from behavioral patterns(Zhao et al., 2025), continuous insight generation from data streams (Yang et al., 2025b), and context-aware action generation(Shaikh et al., 2025). However, these systems lack systematic evaluation frameworks that assess end-to-end proactive capabilities.

Agent Benchmarking: Existing benchmarks predominantly evaluate reactive systems: SWE-bench (Jimenez et al., 2024) for software engineering; ToolBench (Qin et al., 2023) for API calling; and GAIA (Mialon et al., 2023) for multi-hop reasoning. Recent proactivity benchmarks like Proactive-VideoQA (Wang et al., 2025) and ProCIS (Samarinas & Zamani, 2024) begin addressing this gap but remain limited to conversational actions. None of the above-mentioned works decompose proactivity into constituent capabilities or test comprehensive task execution across extended contexts and time horizons, motivating the systematic approach found in PROBE .

6 Conclusion

In this work we propose PROBE : a benchmark designed to test proactivity by having agents search over a personal datastore, identify bottlenecks without prompting, and resolve them. We evaluate leading LLMs and modern agentic solutions on this benchmark, and discover that most solutions struggle greatly at all three stages. We also conduct an analysis of failure modes, illustrating the difficulty of our benchmark's subcomponents.

While the work shows the difficulty of proactive assistance, it still only encompasses a part of the challenge: the problems of building good world-models for individual users and figuring out when to act remain unsolved. We leave these challenges to future work, with the hope that ongoing research will work towards personalized, dynamic agents that can identify and resolve the type of bottlenecks found in our benchmark.

7 Limitations and Future Work

While our work advances proactive agent evaluation, several limitations present opportunities for future research. First, we assume a fixed, non-evolving world model across the time dimension. In real-world proactivity settings, personalization is a more fundamental component and represents a complex challenge. User preferences and contexts evolve over time, requiring agents to adapt their understanding dynamically. Second, we assume that for a given state of information, bottlenecks are resolvable by a single action. Many real-world bottlenecks involve complex, multi-step workflows that require dynamic task execution, where each action modifies the agent's state. These multi-step scenarios introduce additional complexity beyond the scope of this paper.

These limitations suggest natural directions for future work: developing benchmarks that incorporate temporal dynamics and evolving user models, and extending evaluation frameworks to handle multi-step bottleneck resolution with interdependent actions. Addressing these challenges will be essential for advancing proactive agent going forward.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lIVRgt4nLv.

Anthropic. Claude sonnet 4. https://www.anthropic.com/claude/sonnet, 2025a. Accessed 2025-09-24.

Anthropic. Claude opus 4.1. https://www.anthropic.com/news/claude-opus-4-1, August 2025b. Accessed: 2025-09-21.

Navid Ayoobi, Sadat Shahriar, and Arjun Mukherjee. The looming threat of fake and llm-generated linkedin profiles: Challenges and opportunities for detection and prevention. In *Proceedings of the 34th ACM Conference on Hypertext and Social Media*, pp. 1–10, 2023.

Natasha Butt, Varun Chandrasekaran, Neel Joshi, Besmira Nushi, and Vidhisha Balachandran. Benchagents: Automated benchmark creation with agent interaction, 2024. URL https://arxiv.org/abs/2410.22584.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Yuru Jiang, Wenxuan Ding, Shangbin Feng, Greg Durrett, and Yulia Tsvetkov. Sparta alignment: Collectively aligning multiple language models through combat, 2025. URL https://arxiv.org/abs/2506.04721.

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

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. MDAgents: An adaptive collaboration of LLMs for medical decision-making. In *The Thirty-eighth Annual Conference on*

Neural Information Processing Systems, 2024. URL https://openreview.net/forum?id=EKdk4vxKO4.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. On llms-driven synthetic data generation, curation, and evaluation: A survey, 2024. URL https://arxiv.org/abs/2406.15126.

Ya-Ting Lu, Shenzhi Yang, Cheng Qian, Gui-Fang Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Proactive agent: Shifting llm agents from reactive responses to active assistance. *ArXiv*, abs/2410.12361, 2024a. URL https://api.semanticscholar.org/CorpusID:273375463.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Proactive agent: Shifting llm agents from reactive responses to active assistance, 2024b. URL https://arxiv.org/abs/2410.12361.

GrÃl'goire Mialon, ClÃl'mentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Mihai NadÇŐÅ§, Laura DioÅ§an, and Andreea Tomescu. Synthetic data generation using large language models: Advances in text and code. *IEEE Access*, 13:134615âĂŞ134633, 2025. ISSN 2169-3536. doi: 10.1109/access.2025.3589503. URL http://dx.doi.org/10.1109/ACCESS.2025.3589503.

OpenAI. Introducing gpt-4.1 in the api. https://openai.com/index/gpt-4-1/, 4 2025a. Accessed 2025-09-24.

OpenAI. Gpt-4.1 mini âĂŤ model documentation. https://platform.openai.com/docs/models/gpt-4.1-mini, 2025b. Accessed 2025-09-24.

OpenAI. Introducing gpt-5 for developers. https://openai.com/index/introducing-gpt-5-for-developers/, 8 2025c. Describes availability of gpt-5, gpt-5-mini, and gpt-5-nano; Accessed 2025-09-24.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

OpenAI. Gpt-5 is here. https://openai.com/gpt-5/, 2025. Accessed: 2025-09-21.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Chris Samarinas and Hamed Zamani. Procis: A benchmark for proactive retrieval in conversations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, pp. 830âĂŞ840. ACM, July 2024. doi: 10.1145/3626772.3657869. URL http://dx.doi.org/10.1145/3626772.3657869.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, et al. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, CÃ■an Hughes, Charles Lau, Justin Chen, Fereshteh Mahvar, Liron Yatziv, Tiffany Chen, Bram Sterling, Stefanie Anna Baby, Susanna Maria Baby, Jeremy Lai, Samuel Schmidgall, Lu Yang, Kejia Chen, Per Bjornsson, Shashir Reddy, Ryan Brush, Kenneth Philbrick, Mercy Asiedu, Ines Mezerreg, Howard Hu, Howard Yang, Richa Tiwari, Sunny Jansen, Preeti Singh, Yun Liu, Shekoofeh Azizi, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre RamÃl', Morgane Riviere, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela

Ramos, Edouard Yvinec, Michelle Casbon, Elena Buchatskaya, Jean-Baptiste Alayrac, Dmitry Lepikhin, Vlad Feinberg, Sebastian Borgeaud, Alek Andreev, Cassidy Hardin, Robert Dadashi, LÃl'onard Hussenot, Armand Joulin, Olivier Bachem, Yossi Matias, Katherine Chou, Avinatan Hassidim, Kavi Goel, Clement Farabet, Joelle Barral, Tris Warkentin, Jonathon Shlens, David Fleet, Victor Cotruta, Omar Sanseviero, Gus Martins, Phoebe Kirk, Anand Rao, Shravya Shetty, David F. Steiner, Can Kirmizibayrak, Rory Pilgrim, Daniel Golden, and Lin Yang. Medgemma technical report, 2025. URL https://arxiv.org/abs/2507.05201.

Omar Shaikh, Shardul Sapkota, Shan Rizvi, Eric Horvitz, Joon Sung Park, Diyi Yang, and Michael S. Bernstein. Creating general user models from computer use, 2025. URL https://arxiv.org/abs/2505.10831.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Yunpeng Song, Yiheng Bian, Yongtao Tang, Guiyu Ma, and Zhongmin Cai. Visiontasker: Mobile task automation using vision based ui understanding and llm task planning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706288. doi: 10.1145/3654777.3676386. URL https://doi.org/10.1145/3654777.3676386.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Yueqian Wang, Xiaojun Meng, Yifan Wang, Huishuai Zhang, and Dongyan Zhao. Proactivevideoqa: A comprehensive benchmark evaluating proactive interactions in video large language models, 2025. URL https://arxiv.org/abs/2507.09313.

Binfeng Xu, Zhiyuan PENG, Bowen Lei, Subhabrata Mukherjee, and Dongkuan Xu. DECOUPLING REASONING FROM OBSERVATIONS FOR EFFICIENT AUGMENTED LANGUAGE MODELS, 2024. URL https://openreview.net/forum?id=CpgoO6j6W1.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *ArXiv*, abs/2505.14668, 2025a. URL https://api.semanticscholar.org/CorpusID:278769319.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions, 2025b. URL https://arxiv.org/abs/2505.14668.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

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

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yi Eve Sun, Chen Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, F. Yin, Yitao Liang, and Yaodong Yang. Proagent: Building proactive cooperative agents with large language models. In *AAAI Conference on Artificial Intelligence*, 2023. URL https://api.semanticscholar.org/CorpusID:261064959.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. Ask-before-plan: Proactive language agents for real-world planning. *ArXiv*, abs/2406.12639, 2024. URL https://api.semanticscholar.org/CorpusID:270561990.

Yao Zhang, Zijian Ma, Yunpu Ma, Zhen Han, Yu Wu, and Volker Tresp. Webpilot: a versatile and autonomous multi-agent system for web task execution with strategic exploration. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i22.34505. URL https://doi.org/10.1609/aaai.v39i22.34505.

Yuheng Zhao, Xueli Shu, Liwen Fan, Lin Gao, Yu Zhang, and Siming Chen. Proactiveva: Proactive visual analytics with llm-based ui agent, 2025. URL https://arxiv.org/abs/2507.18165.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. In *ICML*, 2024. URL https://openreview.net/forum?id=piecKJ2DlB.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

A Appendix A: Baseline Implementations

We provide implementation details for the agentic framework baselines. Full source code is available at https://github.com/anonymized.

All baseline agents share a common set of document retrieval tools and error handling mechanisms. The `semantic_search` tool performs vector-based retrieval using `text-embedding-3-small` embeddings with configurable result limits, while the `sql_reader` tool executes structured SQLite queries over document metadata with schema validation. All agents incorporate robust JSON parsing with fallback mechanisms and graceful error recovery for malformed outputs. These shared components ensure consistent document accessibility and reliable structured output generation across baselines.

A.1 REACT

The ReACT agent follows the canonical Thought-Action-Observation pattern to iteratively reason about potential bottlenecks and search documents before selecting an appropriate response. The agent operates in a turn-based loop where each iteration generates a thought about the current context, selecting an action to take, and processing the resulting observations. The agent leverages both data exploration tools to retrieve relevant documents and action execution tools dynamically loaded from the world model. Once the agent identifies a bottleneck through its exploration and reasoning process, it selects an appropriate action from the available options and terminates, returning the retrieved documents, bottleneck description, and chosen action. To ensure robustness, the implementation includes safeguards such as token usage management for long contexts, maximum turn limits to prevent infinite loops, and fallback strategies for cases where the agent fails to converge on a definitive solution.

A.2 REFLEXION

The Reflexion agent leverages a verbal reinforcement learning approach that operates through iterative trial-and-error with self-reflection. The agent follows a structured three-step workflow: first searching for relevant documents, then analyzing retrieved documents to identify workflow bottlenecks, and finally selecting appropriate actions to address the identified issues. What distinguishes this architecture is its reflection mechanism: when an attempt fails to meet a quality threshold (evaluated by an LLM-based scoring system), the agent generates verbal reflections on what went wrong and incorporates these learnings into subsequent trials. This creates a feedback loop where the agent progressively improves its document retrieval strategies, bottleneck identification accuracy, and action selection through accumulated reflections from previous failures. The system runs multiple trials until either achieving a successful result (score ≥ 0.8) or exhausting the maximum number of attempts, making it particularly effective for complex productivity tasks that require iterative refinement and learning from mistakes.

A.3 REWOO

The ReWOO consists of a three-stage modular workflow for bottleneck identification and resolution. The architecture follows a Plan-Work-Solve paradigm where the system first generates a structured plan to gather evidence, executes that plan using specialized workers, and then synthesizes the evidence to identify bottlenecks and propose actions. The Planner component analyzes the user's world model to create a step-by-step evidence gathering strategy, storing intermediate results in variables ($\#E1$, $\#E2$, etc.). The Worker stage then executes this plan using three specialized tools: `semantic_search` and `sql_reader` for finding relevant documents and LLM reasoning for analysis. Finally, the Solver component reviews all gathered evidence to identify the most critical bottleneck pattern and select the appropriate action from the available options.

B APPENDIX B: DATA GENERATION PROMPTS

This appendix contains all prompt templates used in the PROBE evaluation pipeline for generating synthetic evaluation data. The pipeline consists of five main stages, each with its own set of prompts:

1. **World Model Generation**: Creates comprehensive context from LinkedIn personas
2. **Bottleneck Injection**: Generates realistic productivity bottlenecks
3. **Checklist Generation**: Creates three-step evaluation checklists
4. **True Positive Generation**: Generates corpus items containing evidence
5. **Distractor Generation**: Creates plausible but irrelevant corpus items

Each prompt is designed as a Jinja2 template, allowing dynamic content insertion based on the evaluation context.

B.1 World Model Generation Prompts

The World Model Generator creates comprehensive professional contexts from LinkedIn personas, including relationships, personal context, available actions, and organizational structure.

B.1.1 Generate Actions for Bottleneck

Usage Context

This prompt is used to generate proactive actions that a persona can take to address specific bottlenecks. It runs after bottlenecks have been identified and creates action items that are contextually appropriate for the persona's role and organization.

```
You are tasked with generating proactive actions for a professional based on specific bottlenecks they
      face.

PERSONA INFORMATION:
- Name: {{ persona.name }}
- Occupation: {{ persona.occupation }}
- Location: {{ persona.location }}
- About: {{ persona.about }}

ORGANIZATION CONTEXT:
- Company: {{ org_structure.company_name }}
- Department: {{ org_structure.department }}
- Team Size: {{ org_structure.team_size }}
- Reports To: {{ org_structure.reporting_to }}

BOTTLENECKS TO ADDRESS:
{% for bottleneck in bottlenecks %}
{{ loop.index }}. {{ bottleneck.description }}
{% endfor %}

DIFFICULTY LEVEL: {{ difficulty }}

## CRITICAL REQUIREMENTS:
1. Generate exactly {{ num_actions }} proactive actions total
2. **EXACTLY ONE ACTION** should solve each bottleneck - no more, no less
3. The remaining actions should be realistic and detailed workplace actions that DON'T solve any of the
      bottlenecks
4. Make it clear which action solves which bottleneck through the action's description and parameters
5. **NO NAMED ENTITIES**: Actions must NOT contain specific person names, company names, or proper nouns
       from the bottlenecks

## ACTION CATEGORIES:
- send_email: Send new emails to individuals or groups
- reply_email: Reply to existing email threads
- schedule_meeting: Create new meetings or events
- reschedule_meeting: Move or modify existing meetings
- cancel_meeting: Cancel scheduled meetings
- create_task: Create new tasks or tickets
- delegate_task: Assign tasks to team members
- update_task_status: Update progress on existing tasks
- create_document: Create new documents, reports, or presentations
- update_document: Edit or revise existing documents
- share_document: Share documents with stakeholders
- send_slack_message: Send instant messages via Slack
- make_phone_call: Initiate phone calls
- request_access: Request access to systems or resources
- provide_feedback: Give feedback on work or proposals
- request_approval: Ask for sign-offs or approvals
- escalate_to_manager: Escalate issues up the chain
- create_meeting_agenda: Prepare agenda for meetings
- conduct_review: Perform code or document reviews
- update_project_plan: Modify project timelines or scope

OUTPUT FORMAT:
Return a JSON object with an "actions" array. Each action should follow this structure:

{
  "actions": [
    {
      "id": "unique_action_identifier",
      "type": "action_category",
      "description": "Clear description of what this action does",
      "constraints": ["Array of preconditions or policies this action must respect"],
      "params_schema": {
        "required": ["Array of required parameter names"]
      },
      "solves_bottleneck": null or bottleneck_index (1-based index if this action solves a bottleneck)
    }
  ]
}
```

```
{% if difficulty == "easy" %}
For EASY difficulty:
- Actions that solve bottlenecks should be straightforward and obvious
- Include simple parameters like "recipient", "subject", "content"
- Non-bottleneck actions should be basic routine tasks
{% elif difficulty == "medium" %}
For MEDIUM difficulty:
- Actions that solve bottlenecks should require some thought, ids should avoid using bottleneck keywords
    .
- Include parameters like "priority", "stakeholders", "deadline", "approach"
- Non-bottleneck actions should be moderately complex coordination tasks
- All action descriptions should be a bit general, and not mention the bottleneck or its details in any
    way
{% elif difficulty == "hard" %}
For HARD difficulty:
- Actions that solve bottlenecks should be subtle and ids should avoid using bottleneck keywords.
- Non-bottleneck actions should be strategic and cross-functional
- All action descriptions should be somewhat general and vague, and not mention the bottleneck or its
    details in any way
{% endif %}

EXAMPLE for a bottleneck about "Email to David Kim about security audit findings remains unanswered, and
     he does not have the authority to approve the security audit findings":
{
  "actions": [
    {
      "id": "schedule_team_meeting",
      "type": "schedule_meeting",
      "description": "Schedule a regular team meeting to discuss project updates and coordination.",
      "constraints": ["team_availability", "meeting_room_available"],
      "params_schema": {
        "required": ["attendees", "date", "time", "agenda", "location"]
      },
      "solves_bottleneck": null
    },
    {
      "id": "update_project_status",
      "type": "update_document",
      "description": "Update project status documentation with current progress and milestones.",
      "constraints": ["document_access", "accurate_information"],
      "params_schema": {
        "required": ["document_id", "status_update", "completion_percentage", "next_steps"]
      },
      "solves_bottleneck": null
    },
    {
      "id": "send_weekly_report",
      "type": "send_email",
      "description": "Send weekly progress report to stakeholders and team members.",
      "constraints": ["report_data_available", "stakeholder_list_current"],
      "params_schema": {
        "required": ["recipients", "subject", "report_content", "attachments"]
      },
      "solves_bottleneck": null
    },
    {
      "id": "conduct_code_review",
      "type": "conduct_review",
      "description": "Review code changes submitted by team members for quality and standards compliance
    .",
      "constraints": ["technical_expertise", "time_available"],
      "params_schema": {
        "required": ["pull_request_id", "review_criteria", "feedback_type", "approval_status"]
      },
      "solves_bottleneck": null
     },
      {
      "id": "escalate_issue",
      "type": "escalate_to_manager",
      "description": "Escalate an important issue to management for resolution.",
      "constraints": ["multiple_attempts_made", "deadline_approaching", "requires_higher_authority"],
      "params_schema": {
        "required": ["original_recipient", "escalation_recipient", "urgency_level", "business_impact", "
    attempted_contacts"]
      },
      "solves_bottleneck": 1
    },
    {
      "id": "delegate_routine_task",
      "type": "delegate_task",
      "description": "Delegate routine tasks to appropriate team members to optimize workload
    distribution.",
      "constraints": ["team_capacity", "skill_match"],
      "params_schema": {
        "required": ["assignee", "task_description", "deadline", "priority_level"]
      },
      "solves_bottleneck": null
    }
  ]
}
```

```
Ensure that:
1. Each bottleneck has EXACTLY ONE action that can solve it, all other actions should certainly not
     solve the bottleneck
2. The action description of the correct action should address the bottleneck, but without mentioning
     the bottleneck, keywords, or its details in any way.
3. Other actions are detailed and realistic but explicitly DON'T solve any of the listed bottlenecks
4. Total number of actions equals {{ num_actions }}
5. **CRITICAL**: The actions should not include any mention of the people or situations involved in the
     bottleneck
```

Listing 1: generate_actions_for_bottleneck.j2

B.1.2 GENERATE ORGANIZATION STRUCTURE

Usage Context

This prompt generates the organizational context around a persona, including company structure, team composition, reporting lines, and key processes. It's one of the first prompts executed to establish the professional environment.

```
Generate a realistic organizational structure for the following professional:

Name: {{ persona.name }}
Occupation: {{ persona.occupation }}
Location: {{ persona.location }}
About: {{ persona.about }}

Create a detailed organizational context that includes:
1. Company name and type
2. Department structure
3. Team composition
4. Reporting relationships
5. Key processes and workflows

The organization should be realistic for someone in their role and location.

Provide your response as a JSON object with this structure:
{
  "company_name": "Name of the company",
  "company_type": "Type of company (startup, enterprise, etc.)",
  "department": "Their department name",
  "team_size": 5,
  "direct_reports": 2,
  "reporting_to": "Title of their manager",
  "key_processes": ["Process 1", "Process 2"],
  "typical_meetings": ["Meeting type 1", "Meeting type 2"]
}
```

Listing 2: generate_org_structure.j2

B.1.3 GENERATE PERSONAL CONTEXT

Usage Context

This prompt creates personal work context for the persona, including their work style, preferences, current goals, constraints, and tools they use. This adds depth to the persona beyond their LinkedIn profile.

```
Generate personal work context for the following professional:

PERSONA:
- Name: {{ persona.name }}
- Occupation: {{ persona.occupation }}
- About: {{ persona.about }}

DIFFICULTY: {{ difficulty }}

Create realistic personal context including:
1. Work style and preferences
2. Current goals and priorities
3. Time constraints and challenges
4. Communication preferences
5. Tools and systems they use
```

```
{% if difficulty == "easy" %}
Keep the context simple with straightforward preferences and minimal constraints.
{% elif difficulty == "medium" %}
Include moderate complexity with some competing priorities and constraints.
{% elif difficulty == "hard" %}
Create complex context with multiple competing priorities, significant constraints, and nuanced
     preferences.
{% endif %}

Provide your response as a JSON object with this structure:
{
  "work_style": "Description of how they prefer to work",
  "current_goals": ["Goal 1", "Goal 2", "Goal 3"],
  "constraints": ["Time constraint", "Resource constraint"],
  "communication_preferences": "How they prefer to communicate",
  "tools_used": ["Tool 1", "Tool 2"],
  "peak_productivity_time": "When they work best",
  "biggest_challenges": ["Challenge 1", "Challenge 2"]
}
```

Listing 3: generate_personal_context.j2

B.1.4 GENERATE RELATIONSHIPS

Usage Context

This prompt generates professional relationships for the persona, including colleagues, clients, stakeholders, and collaborators. The number and complexity of relationships varies based on the difficulty level.

```
Generate professional relationships for the following person:

Name: {{ persona.name }}
Occupation: {{ persona.occupation }}
Location: {{ persona.location }}
About: {{ persona.about }}

DIFFICULTY LEVEL: {{ difficulty }}

{% if difficulty == "easy" %}
Generate 3-5 key relationships that are straightforward and clearly defined.
{% elif difficulty == "medium" %}
Generate 5-8 relationships with moderate complexity and some overlapping responsibilities.
{% elif difficulty == "hard" %}
Generate 8-12 relationships with complex interdependencies and nuanced dynamics.
{% endif %}

For each relationship, provide:
1. The person's full name
2. Their role/title
3. Type of relationship (colleague, client, manager, stakeholder, collaborator)
4. How they interact with {{ persona.name }}
5. Current status of the relationship

Provide your response as a JSON object with this structure:
{
  "relationships": [
    {
      "name": "Full name",
      "role": "Their job title",
      "type": "colleague|client|manager|stakeholder|collaborator",
      "interaction": "Description of how they work together",
      "status": "Current state of the relationship",
      "frequency": "How often they interact"
    }
  ]
}
```

Listing 4: generate_relationships.j2

B.2 BOTTLENECK INJECTION PROMPTS

The Bottleneck Injector creates realistic productivity bottlenecks that can be addressed by the persona's available actions.

B.2.1 GENERATE INDIVIDUAL BOTTLENECK

Usage Context

This prompt generates a single, highly specific bottleneck for a persona. It's called multiple times to create a set of bottlenecks, each focusing on a different aspect of the persona's work challenges.

```
You are creating a SINGLE, highly specific productivity bottleneck for {{ persona_name }}.

CONTEXT:
- Occupation: {{ persona_occupation }}
- About: {{ persona_about }}
- Difficulty: {{ difficulty }}

ORGANIZATION:
- Company: {{ org_structure.company_name }}
- Department: {{ org_structure.department }}
- Team Size: {{ org_structure.team_size }}
- Reports To: {{ org_structure.reporting_to }}

KEY RELATIONSHIPS:
{% for rel in relationships[:5] %}
- {{ rel.name }} ({{ rel.role }}): {{ rel.interaction }}
{% endfor %}

PERSONAL CONTEXT:
- Work Style: {{ personal_context.work_style }}
- Current Goals: {{ personal_context.current_goals[:3] | join(', ') }}
- Constraints: {{ personal_context.constraints | join(', ') }}

BOTTLENECK #{{ bottleneck_index }}

Create ONE specific bottleneck that:
1. References REAL NAMES from the relationships
2. Mentions SPECIFIC documents, meetings, or deadlines
3. Has a clear timeline or urgency
4. Can be discovered through search/investigation
5. Is solvable through proactive action

{% if difficulty == "easy" %}
Make it straightforward with clear cause and solution.
{% elif difficulty == "medium" %}
Include some complexity and multiple stakeholders.
{% elif difficulty == "hard" %}
Make it complex with competing priorities and hidden dependencies.
{% endif %}

The bottleneck should be 2-3 sentences maximum and extremely specific.

Example format:
"The Q3 product roadmap review with Sarah Chen is scheduled for next Tuesday, but the feature
     prioritization matrix she requested hasn't been updated since July because the engineering
     estimates from Michael Park's team are still pending in JIRA tickets ENG-4521 through ENG-4525."

Generate a single bottleneck description:
```

Listing 5: generate_bottleneck.j2

B.2.2 GENERATE BOTTLENECKS BATCH

Usage Context

This prompt generates multiple bottlenecks in a single LLM call for efficiency. It ensures variety across different work aspects while maintaining consistency with the persona's context.

```
You are creating {{ num_bottlenecks }} highly specific productivity bottlenecks for {{ persona_name }}.

CONTEXT:
- Occupation: {{ persona_occupation }}
- About: {{ persona_about }}
- Difficulty: {{ difficulty }}

ORGANIZATION:
- Company: {{ org_structure.company_name }}
- Department: {{ org_structure.department }}
- Team Size: {{ org_structure.team_size }}
```

```
KEY RELATIONSHIPS:
{% for rel in relationships %}
- {{ rel.name }} ({{ rel.role }}): {{ rel.interaction }}
{% endfor %}

PERSONAL CONTEXT:
- Work Style: {{ personal_context.work_style }}
- Current Goals: {{ personal_context.current_goals | join(', ') }}
- Constraints: {{ personal_context.constraints | join(', ') }}

Generate {{ num_bottlenecks }} DIFFERENT bottlenecks that:
1. Each references REAL NAMES from the relationships
2. Mentions SPECIFIC artifacts (documents, meetings, systems)
3. Has clear urgency or timeline
4. Can be discovered through search
5. Is solvable through action

ENSURE VARIETY:
- Different types of problems (delays, missing info, conflicts, etc.)
- Different people involved
- Different urgency levels
- Different solutions needed

{% if difficulty == "easy" %}
Make them straightforward with clear causes.
{% elif difficulty == "medium" %}
Include moderate complexity.
{% elif difficulty == "hard" %}
Make them complex with hidden dependencies.
{% endif %}

Provide your response as a JSON object:
{
  "bottlenecks": [
    {
      "description": "Specific 2-3 sentence bottleneck",
      "primary_person": "Main person involved",
      "urgency": "high|medium|low",
      "type": "delay|missing_info|conflict|approval|resource|coordination"
    }
  ]
}
```

Listing 6: generate_bottlenecks_batch.j2

B.3 Checklist Generation Prompts

The Checklist Generator creates three-step evaluation checklists that test an agent's ability to complete the proactive workflow.

B.3.1 Three-Step Checklist

Usage Context

This prompt generates the core three-step checklist (Search âĘŠ Identification âĘŠ Task Selection) for evaluating agent performance on a specific bottleneck. It's the primary evaluation structure.

```
Generate a three-step checklist for addressing the following bottleneck:

BOTTLENECK:
{{ bottleneck.description }}

WORLD MODEL CONTEXT:
- Persona: {{ world_model.persona_full_name }} ({{ world_model.persona_occupation }})
- Company: {{ world_model.organizational_structure.company_name }}
- Difficulty: {{ difficulty }}

KEY RELATIONSHIPS:
{% for rel in world_model.relationships[:5] %}
- {{ rel.name }} ({{ rel.role }}): {{ rel.interaction }}
{% endfor %}

AVAILABLE ACTIONS:
{% for action in available_actions[:8] %}
- {{ action.action_type }}: {{ action.name }}
{% endfor %}

Create a three-step checklist with:
```

```
STEP 1 - SEARCH: What specific information should be searched for?
- Include 3-5 specific search queries or data sources
- Reference actual names, documents, or systems from the bottleneck
- Mix of different search types (emails, documents, calendar, etc.)

STEP 2 - IDENTIFICATION: What key insights should be identified?
- 2-3 specific findings that reveal the root cause
- Reference actual evidence that would be found
- Clear connection to the bottleneck

STEP 3 - TASK SELECTION: What action should be taken?
- Select from available actions
- Include specific parameters (who, what, when)
- Clear resolution to the bottleneck

Provide your response as a JSON object:
{
  "search_step": {
    "description": "What to search for and why",
    "specific_queries": [
      "Query 1 with actual names/docs",
      "Query 2 with specific terms",
      "Query 3 with system references"
    ],
    "expected_sources": ["email", "calendar", "documents"]
  },
  "identification_step": {
    "description": "What insights to identify",
    "key_findings": [
      "Specific finding 1",
      "Specific finding 2"
    ],
    "root_cause": "The underlying issue"
  },
  "task_selection_step": {
    "action_type": "One of the available action types",
    "description": "Specific action to take",
    "parameters": {
      "participants": ["Names"],
      "timeline": "When",
      "deliverables": "What"
    }
  }
}
```

Listing 7: three_step_checklist.j2

B.4 True Positive Generation Prompts

These prompts generate corpus items that contain evidence of bottlenecks, serving as the "ground truth" that agents should find.

B.4.1 Plan Evidence Distribution

Usage Context

This prompt plans how evidence for a bottleneck will be distributed across multiple corpus items. It ensures comprehensive coverage while avoiding contamination from other bottlenecks.

```
You are planning how to distribute evidence for a bottleneck across multiple documents.

BOTTLENECK TO ADDRESS:
{{ bottleneck.description }}

WORLD MODEL CONTEXT:
- Persona: {{ world_model.persona_full_name }} ({{ world_model.persona_occupation }})
- Company: {{ world_model.organizational_structure.company_name }}

KEY RELATIONSHIPS:
{% for rel in world_model.relationships[:5] %}
- {{ rel.name }} ({{ rel.role }})
{% endfor %}

AVAILABLE DOCUMENT TYPES:
- Email (conversations, requests, updates)
- Calendar (meetings, deadlines, events)
- Document (reports, plans, specifications)

OTHER BOTTLENECKS TO AVOID:
```

```
{% for other in other_bottlenecks %}
- {{ other }}
{% endfor %}

Plan how to distribute evidence across {{ num_documents }} documents:
1. Each document should contain a different aspect/angle of the bottleneck
2. Together they should tell the complete story
3. Avoid ANY mention of other bottlenecks
4. Make evidence discoverable but not too obvious

Provide your response as a JSON object:
{
  "evidence_distribution": [
    {
      "document_type": "email|calendar|document",
      "evidence_role": "What aspect this covers",
      "key_information": "Specific info to include",
      "sender_or_creator": "Who creates this",
      "discoverability": "How someone would find this"
    }
  ]
}
```

Listing 8: plan_evidence_distribution.j2

B.4.2 GENERATE EMAIL EVIDENCE

Usage Context

This prompt generates email corpus items that contain evidence of the bottleneck. Emails often contain requests, updates, and clarifications that reveal bottleneck details.

```
Generate a realistic email that contains evidence of the following bottleneck:

BOTTLENECK:
{{ bottleneck.description }}

EVIDENCE ROLE:
This email should specifically show: {{ evidence_role }}

WORLD MODEL:
- Persona: {{ world_model.persona_full_name }} ({{ world_model.persona_occupation }})
- Email: {{ world_model.persona_full_name.lower().replace(' ', '.') }}@{{ world_model.
      organizational_structure.company_name.lower().replace(' ', '').replace(',', '') }}.com

KEY RELATIONSHIPS:
{% for rel in world_model.relationships[:7] %}
- {{ rel.name }} ({{ rel.role }})
{% endfor %}

THINGS TO AVOID:
Do NOT mention or reference these other bottlenecks:
{% for other in other_bottlenecks %}
{{ other }}
{% endfor %}

Generate a complete, realistic email that:
1. Contains clear evidence of the bottleneck
2. Fits the evidence role specified
3. Includes realistic email metadata
4. Uses actual names from relationships
5. Avoids any mention of other bottlenecks
6. Sounds natural and professional

The email should be substantial (200-400 words) and include:
- Proper email headers (From, To, CC, Subject, Date)
- Natural greeting and sign-off
- Specific details that reveal bottleneck information
- Realistic workplace communication style

Format as:
From: sender@company.com
To: recipient@company.com
CC: others@company.com
Subject: Specific subject line
Date: Recent date

Email body...
```

Listing 9: generate_email_evidence.j2

B.4.3 GENERATE CALENDAR EVIDENCE

Usage Context

This prompt generates calendar events that reveal scheduling conflicts, deadlines, or meeting-related bottleneck evidence.

```
Generate a realistic calendar event that contains evidence of the following bottleneck:

BOTTLENECK:
{{ bottleneck.description }}

EVIDENCE ROLE:
This calendar event should show: {{ evidence_role }}

WORLD MODEL:
- Persona: {{ world_model.persona_full_name }}
- Company: {{ world_model.organizational_structure.company_name }}

KEY PEOPLE:
{% for rel in world_model.relationships[:5] %}
- {{ rel.name }} ({{ rel.role }})
{% endfor %}

AVOID MENTIONING:
{% for other in other_bottlenecks %}
- {{ other }}
{% endfor %}

Create a detailed calendar event that:
1. Reveals important timing/scheduling aspects of the bottleneck
2. Includes realistic attendees from relationships
3. Has detailed agenda or description
4. Shows urgency or conflicts if relevant
5. Completely avoids other bottlenecks

Include:
- Title: Specific and professional
- Date/Time: Realistic and relevant to bottleneck
- Duration: Appropriate for the meeting type
- Location: Physical or virtual
- Attendees: Mix of required and optional
- Agenda/Description: Detailed and revealing bottleneck evidence
- Any attached documents or pre-reads

Format your response as a complete calendar event.
```

Listing 10: generate_calendar_evidence.j2

B.4.4 GENERATE DOCUMENT EVIDENCE

Usage Context

This prompt generates longer documents (reports, plans, memos) that contain comprehensive evidence about the bottleneck, often providing context and history.

```
Generate a professional document that contains evidence of the following bottleneck:

BOTTLENECK:
{{ bottleneck.description }}

EVIDENCE ROLE:
This document should provide: {{ evidence_role }}

CONTEXT:
- Author: {{ world_model.persona_full_name }}
- Organization: {{ world_model.organizational_structure.company_name }}
- Department: {{ world_model.organizational_structure.department }}

DOCUMENT REQUIREMENTS:
- Length: {{ min_words }}-{{ max_words }} words
- Type: Report, memo, plan, or specification
- Should reveal key bottleneck information
- Must seem like a natural workplace document

KEY PEOPLE TO REFERENCE:
{% for rel in world_model.relationships[:6] %}
- {{ rel.name }} ({{ rel.role }})
```

```
{% endfor %}

STRICTLY AVOID:
{% for other in other_bottlenecks %}
- {{ other }}
{% endfor %}

Generate a complete professional document that:
1. Has proper header (title, date, author, recipients)
2. Contains multiple sections with clear headings
3. Embeds bottleneck evidence naturally throughout
4. References real people and specific details
5. Maintains professional tone and formatting
6. Includes actionable information
7. Never mentions other bottlenecks

The document should read like an authentic workplace artifact that someone would search for when
     investigating the bottleneck.
```

Listing 11: generate_document_evidence.j2

B.4.5 GENERATE DYNAMIC SOURCES

Usage Context

This prompt identifies additional data sources or systems where evidence might be found, expanding beyond the standard email/calendar/document trinity.

```
Identify specific data sources where evidence for this bottleneck would be found:

BOTTLENECK:
{{ bottleneck.description }}

ORGANIZATION:
- Company: {{ world_model.organizational_structure.company_name }}
- Industry: {{ world_model.organizational_structure.company_type }}
- Department: {{ world_model.organizational_structure.department }}

Suggest 3-5 specific systems, databases, or specialized sources where evidence would exist.

For each source:
1. Name the specific system/platform
2. What evidence would be found there
3. How to search/access it
4. Why it's relevant to this bottleneck

Examples: JIRA tickets, Confluence pages, Slack channels, CRM records, Github PRs, etc.

Provide specific names and identifiers, not generic categories.
```

Listing 12: generate_dynamic_sources.j2

B.4.6 REVIEW EVIDENCE

Usage Context

This prompt is used to review generated evidence for quality, ensuring it properly supports the bottleneck discovery without contamination from other bottlenecks.

```
Review the following evidence for quality and effectiveness:

BOTTLENECK BEING EVIDENCED:
{{ bottleneck.description }}

GENERATED EVIDENCE:
{{ evidence_content }}

OTHER BOTTLENECKS TO AVOID:
{% for other in other_bottlenecks %}
- {{ other }}
{% endfor %}

Evaluate:
1. Does the evidence clearly support discovering this bottleneck?
2. Is it discoverable through realistic search queries?
```

```
3. Does it avoid ALL mentions of other bottlenecks?
4. Is it natural and realistic for the workplace context?
5. Are all names, dates, and details consistent?

Provide:
1. Quality score (1-10)
2. Strengths of the evidence
3. Any issues or contamination found
4. Suggested improvements
5. Search queries that would find this evidence

Format as JSON:
{
  "quality_score": 8,
  "strengths": ["Clear timeline", "Specific names"],
  "issues": ["Might be too obvious"],
  "improvements": ["Add more context about..."],
  "search_queries": ["Michael Park ENG-4521", "Q3 roadmap review"]
}
```

Listing 13: review_evidence.j2

B.5 Distractor Generation Prompts

These prompts generate plausible but irrelevant corpus items that test the agent's ability to filter out noise.

B.5.1 Generate Email Distractors

Usage Context

This prompt generates realistic workplace emails that are plausible distractors - they should seem relevant to the persona's work but not contain evidence of any bottlenecks.

```
Generate {{ count }} realistic workplace emails for {{ world_model.persona_full_name }}'s context.

CONTEXT:
- Role: {{ world_model.persona_occupation }}
- Company: {{ world_model.organizational_structure.company_name }}
- Department: {{ world_model.organizational_structure.department }}

RELATIONSHIPS TO USE:
{% for rel in world_model.relationships %}
- {{ rel.name }} ({{ rel.role }}): {{ rel.interaction }}
{% endfor %}

CRITICAL - AVOID ALL BOTTLENECKS:
{% for bottleneck in bottlenecks %}
 {{ bottleneck.description }}
{% endfor %}

REQUIREMENTS:
1. Emails must be completely unrelated to any bottleneck
2. Should be realistic workplace communications
3. Vary the types: updates, requests, FYIs, discussions
4. Use different senders and recipients
5. Include realistic dates and subjects
6. Length: 150-300 words each

Generate diverse emails about:
- Routine status updates
- General team communications
- Company announcements
- Non-critical planning
- Social/cultural events
- Training or development
- General process discussions

Ensure NONE of the emails could be interpreted as evidence for any bottleneck.

Format each email with proper headers (From, To, Subject, Date) followed by the body.
```

Listing 14: generate_email_distractors.j2

B.5.2 GENERATE CALENDAR DISTRACTORS

Usage Context

This prompt creates calendar events that represent normal workplace meetings and events, serving as noise that agents must filter through.

```
Generate {{ count }} realistic calendar events for {{ world_model.persona_full_name }}.

CONTEXT:
- Role: {{ world_model.persona_occupation }}
- Company: {{ world_model.organizational_structure.company_name }}
- Typical Meetings: {{ world_model.organizational_structure.typical_meetings | join(', ') }}

PEOPLE TO INCLUDE:
{% for rel in world_model.relationships[:8] %}
- {{ rel.name }} ({{ rel.role }})
{% endfor %}

MUST AVOID - NO BOTTLENECK EVIDENCE:
{% for bottleneck in bottlenecks %}
 {{ bottleneck.description }}
{% endfor %}

Create diverse calendar events:
1. Regular recurring meetings (1-on-1s, team standups)
2. Training or development sessions
3. Company-wide events
4. Social activities
5. Planning sessions (unrelated to bottlenecks)
6. Reviews or retrospectives

Each event needs:
- Title: Professional and specific
- Date/Time: Spread across different days/times
- Duration: Realistic for the meeting type
- Attendees: Appropriate mix of people
- Location/Link: Physical or virtual
- Description: Detailed agenda that contains NO bottleneck evidence

Make them indistinguishable from real important meetings but completely unrelated to bottlenecks.
```

Listing 15: generate_calendar_distractors.j2

B.5.3 GENERATE DOCUMENT DISTRACTORS

Usage Context

This prompt generates longer-form documents that serve as distractors, representing typical workplace documentation that doesn't relate to any bottlenecks.

```
Generate {{ count }} professional documents for {{ world_model.persona_full_name }}'s work context.

CONTEXT:
- Role: {{ world_model.persona_occupation }}
- Department: {{ world_model.organizational_structure.department }}
- Company: {{ world_model.organizational_structure.company_name }}

DOCUMENT TYPES TO CREATE:
1. Process documentation
2. Team updates or newsletters
3. Project proposals (unrelated to bottlenecks)
4. Meeting notes
5. Training materials
6. Policy documents

CRITICAL - AVOID ALL BOTTLENECKS:
{% for bottleneck in bottlenecks %}
 DO NOT REFERENCE: {{ bottleneck.description }}
{% endfor %}

Each document should:
- Be 400-800 words
- Have professional formatting and structure
- Reference real people from relationships
- Contain valuable but irrelevant information
- Be discoverable by plausible search terms
- Seem important enough to not ignore
```

```
Include proper headers:
- Title
- Author
- Date
- Document type
- Recipients/Audience

The documents should be high-quality distractors that would naturally appear in search results but
     provide no evidence for any bottleneck.
```

Listing 16: generate_document_distractors.j2

B.5.4 GENERATE NATURAL DISTRACTOR

Usage Context

This is a general-purpose prompt for generating natural distractors of any type, with emphasis on making them realistic and contextually appropriate.

```
Generate a natural {{ kind }} distractor for the following context:

PERSONA: {{ world_model.persona_full_name }} ({{ world_model.persona_occupation }})
COMPANY: {{ world_model.organizational_structure.company_name }}

AVAILABLE RELATIONSHIPS:
{% for rel in world_model.relationships[:6] %}
- {{ rel.name }} ({{ rel.role }})
{% endfor %}

WORK CONTEXT:
- Department: {{ world_model.organizational_structure.department }}
- Current Goals: {{ world_model.personal_context.current_goals[:3] | join(', ') }}
- Tools Used: {{ world_model.personal_context.tools_used | join(', ') }}

MUST AVOID (NO EVIDENCE OF):
{% for bottleneck in bottlenecks %}
- {{ bottleneck.description }}
{% endfor %}

Create a {{ kind }} that:
1. Is completely unrelated to any bottleneck
2. Fits naturally in the persona's work life
3. Could plausibly be important
4. Uses real names and realistic details
5. Matches typical {{ kind }} format and style

Focus on routine work activities that would generate {{ kind }}s but don't relate to the specific
     problems being evaluated.
```

Listing 17: generate_natural_distractor.j2

B.5.5 ENHANCE DISTRACTOR

Usage Context

This prompt enhances basic distractors to make them more realistic and harder to distinguish from true positives, adding details and context.

```
Enhance the following distractor to make it more realistic and detailed:

ORIGINAL DISTRACTOR:
{{ original_content }}

CONTEXT:
- Type: {{ distractor_type }}
- Persona: {{ world_model.persona_full_name }}
- Company: {{ world_model.organizational_structure.company_name }}

ENHANCEMENT GOALS:
1. Add more specific details (names, dates, numbers)
2. Include realistic workplace jargon
3. Add urgency or importance markers
4. Reference real systems or processes
5. Make it harder to distinguish from true positives
```

```
MAINTAIN:
- Core message/purpose
- Avoidance of all bottlenecks
- Professional tone
- Realistic length

The enhanced version should be a high-quality distractor that requires careful analysis to determine it'
     s not relevant to the bottlenecks.
```

Listing 18: enhance_distractor.j2

B.5.6 COUNTERFACTUAL EMAIL TEMPLATE

Usage Context

This prompt generates counterfactual scenarios - communications about what could have happened but didn't, useful for testing agent reasoning about hypotheticals.

```
Create an email discussing a hypothetical or counterfactual scenario.

CONTEXT:
- Sender: {{ sender_name }}
- Recipient: {{ recipient_name }}
- Topic: What might have happened if a different decision was made

The email should:
1. Discuss alternative scenarios or outcomes
2. Be clearly hypothetical/counterfactual
3. Not provide evidence for actual bottlenecks
4. Maintain professional tone

Example: "If we had chosen vendor B instead of vendor A..."
```

Listing 19: counterfactual_min_email.j2

B.5.7 ENTITY SWAP TEMPLATE

Usage Context

This prompt creates distractors by swapping entities (people, projects, systems) to create plausible but incorrect variations of real situations.

```
Create a distractor by swapping key entities in a work scenario.

ORIGINAL ENTITIES:
- Person: {{ original_person }}
- Project: {{ original_project }}
- System: {{ original_system }}

SWAP TO:
- Person: {{ swap_person }}
- Project: {{ swap_project }}
- System: {{ swap_system }}

Generate a {{ content_type }} that uses the swapped entities in a realistic but unrelated context.
```

Listing 20: entity_swap.j2

B.5.8 TIME SCOPE TEMPLATE

Usage Context

This prompt generates distractors that reference different time periods, useful for testing whether agents can distinguish current issues from past or future ones.

```
Generate a {{ content_type }} that references a different time period.

ACTUAL TIMEFRAME: {{ actual_timeframe }}
DISTRACTOR TIMEFRAME: {{ distractor_timeframe }}

Create content that:
1. Clearly references the distractor timeframe
2. Discusses similar topics but in a different temporal context
3. Could be confused with current issues if not reading carefully
4. Maintains realism for that time period
```

Listing 21: time_scope.j2

B.5.9 TEMPLATE CLONE DOCUMENT

Usage Context

This prompt creates documents that follow the same template or format as important documents but contain different, unrelated content.

```
Create a document that follows the same template as important documents but with unrelated content.

TEMPLATE TYPE: {{ template_type }}
TYPICAL SECTIONS: {{ sections | join(', ') }}

Generate a document that:
1. Follows the exact same structure
2. Contains completely different, unrelated content
3. Maintains professional appearance
4. Could be mistaken for an important document if not read carefully
```

Listing 22: template_clone_doc.j2

B.5.10 GENERIC COUNTERFACTUAL

Usage Context

This prompt generates generic counterfactual content that discusses alternatives, hypotheticals, or what-if scenarios unrelated to actual bottlenecks.

```
Generate {{ kind }} content discussing counterfactual scenarios for {{ world_model.persona_full_name }}.

CONTEXT:
- Role: {{ world_model.persona_occupation }}
- Department: {{ world_model.organizational_structure.department }}

Create content about:
- Alternative approaches that weren't taken
- Hypothetical scenarios
- Lessons learned from paths not chosen
- What-if analyses

Ensure the counterfactual discussion:
1. Is clearly marked as hypothetical
2. Doesn't relate to actual bottlenecks
3. Provides interesting but irrelevant information
4. Maintains professional tone
```

Listing 23: generic_counterfactual.j2

B.6 LLM-AS-JUDGE EVALUATION PROMPTS

Usage Context

This prompt is used to for evaluating bottleneck identification as described in section 3

```
You are evaluating whether an AI agent correctly identified a bottleneck using precision-based scoring.

## Context
The agent was given various documents and data sources to analyze, and needed to identify the bottleneck
      affecting productivity.

## Workplace Context
Persona: {{ world_model.persona_full_name }}
Company: {{ world_model.organizational_structure.company_name }}
Department: {{ world_model.organizational_structure.department }}
Team Size: {{ world_model.organizational_structure.team_size }}

## What the Agent Identified
"{{ identified_bottleneck }}"

## The Actual Bottleneck
"{{ actual_bottleneck }}"

## Two-Tier Precision Scoring System

### ESSENTIAL DETAILS (Must be 100% correct for any points)
These are execution-critical details required for immediate action:

1. **WHO_BLOCKED**: The exact person/entity being blocked (specific names, not roles)
2. **WHO_BLOCKER**: The exact person/entity causing the blockage (specific names, not roles)
3. **WHAT_TASK**: The specific deliverable/work item that's blocked (exact names, not generic
     descriptions)
4. **WHY_ROOT_CAUSE**: The true underlying reason for the blockage (not symptoms)

### NON-ESSENTIAL DETAILS (Can have minor errors/omissions)
These provide context but aren't critical for immediate resolution:

5. **WHEN_DEADLINE**: Specific dates and timelines
6. **WHERE_SYSTEM**: Exact system/tool names and locations
7. **HOW_MECHANISM**: Detailed process or workflow information
8. **IMPACT_SCOPE**: Downstream effects and dependencies

## Scoring Rules

**CORRECT (1.0)**: All essential details are accurate AND all non-essential details are accurate
**PARTIALLY_CORRECT (0.5)**: All essential details are accurate BUT has errors/omissions in non-
     essential details
**INCORRECT (0.0)**: Any essential detail is wrong, missing, or too vague

## Essential Detail Requirements
- **Names must be specific**: "Timothy" not "someone from finance"
- **Systems must be exact**: "econo.com" not "financial system"
- **Tasks must be precise**: "Q3 financials initiative" not "quarterly report"
- **Root causes must be accurate**: "Martha stripped access" not "system issues"

## Examples

### Example 1: CORRECT (1.0)
**Actual**: Timothy isn't meeting the March 23rd deadline for his Q3 financials initiative because
     Martha revoked his econo.com access to company financials.

**Agent**: Timothy cannot complete his Q3 financials initiative by the March 23rd deadline because
     Martha removed his access to company financials through econo.com.

**Analysis**:
- âIJĚ WHO_BLOCKED: Timothy (correct)
- âIJĚ WHO_BLOCKER: Martha (correct)
- âIJĚ WHAT_TASK: Q3 financials initiative (correct)
- âIJĚ WHY_ROOT_CAUSE: Martha revoked access (correct)
- âIJĚ All non-essential details accurate

### Example 2: PARTIALLY_CORRECT (0.5)
**Actual**: Timothy isn't meeting the March 23rd deadline for his Q3 financials initiative because
     Martha revoked his econo.com access to company financials.

**Agent**: Timothy cannot complete his Q3 financials initiative because Martha removed his access to
     company financial systems.

**Analysis**:
- âIJĚ WHO_BLOCKED: Timothy (correct)
- âIJĚ WHO_BLOCKER: Martha (correct)
- âIJĚ WHAT_TASK: Q3 financials initiative (correct)
- âIJĚ WHY_ROOT_CAUSE: Martha revoked access (correct)
- âİŇ WHEN_DEADLINE: Missing March 23rd
- âİŇ WHERE_SYSTEM: "financial systems" instead of "econo.com"

### Example 3: INCORRECT (0.0)
**Actual**: Timothy isn't meeting the March 23rd deadline for his Q3 financials initiative because
     Martha revoked his econo.com access to company financials.

**Agent**: Someone from finance is having trouble with their quarterly report due to system access
     issues.

**Analysis**:
- âİŇ WHO_BLOCKED: "someone from finance" instead of Timothy
- âİŇ WHO_BLOCKER: Missing Martha entirely
```

```
- âİŇ WHAT_TASK: "quarterly report" instead of Q3 financials initiative
- âİŇ WHY_ROOT_CAUSE: "system issues" instead of Martha's action

## Output Format
Return a JSON object with:
{
  "judgment": "<CORRECT|PARTIALLY_CORRECT|INCORRECT>",
  "essential_details_analysis": {
    "who_blocked": "<correct|incorrect|missing>",
    "who_blocker": "<correct|incorrect|missing>",
    "what_task": "<correct|incorrect|missing>",
    "why_root_cause": "<correct|incorrect|missing>"
  },
  "non_essential_details_analysis": {
    "when_deadline": "<correct|incorrect|missing|n/a>",
    "where_system": "<correct|incorrect|missing|n/a>",
    "how_mechanism": "<correct|incorrect|missing|n/a>",
    "impact_scope": "<correct|incorrect|missing|n/a>"
  },
  "reasoning": "<explanation of your scoring decision focusing on essential vs non-essential accuracy>"
}
```

Listing 24: judge_bottleneck_identification.j2

Usage Context

This prompt is used to for evaluating parameter scoring as part of the task execution evaluation metric as described in section 3

```
You are an expert evaluator specializing in assessing AI agent action parameter selection in workplace
      automation scenarios. Your task is to evaluate whether an AI agent selected appropriate parameters
      for a correctly identified action.

## Your Role and Expertise
You have deep expertise in:
- Workplace automation and task execution
- API parameter design and semantic equivalence
- Business process optimization
- Contextual reasoning in parameter selection

## Evaluation Context

### The Bottleneck Being Resolved
```
{{ bottleneck.description }}
```

### Workplace Environment
- **Company**: {{ world_model.organizational_structure.company_name }}
- **Department**: {{ world_model.organizational_structure.department }}
- **Persona**: {{ world_model.persona_full_name }}

### Key Relationships Available
{% for rel in world_model.relationships[:5] %}
- **{{ rel.name }}** ({{ rel.type.value }}): {{ rel.department if rel.department else "External" }}{% if
      rel.email %} - {{ rel.email }}{% endif %}
{% endfor %}

### Evidence That Led to Bottleneck Discovery
{% if true_positives %}
{% for tp in true_positives[:3] %}
- **{{ tp.get('type', 'Unknown') }}**: {{ tp.get('title', tp.get('subject', 'N/A')) }}
  - Key info: {{ tp.get('summary', tp.get('content', 'Details not shown'))[:100] }}...
{% endfor %}
{% else %}
- No specific evidence items provided for context
{% endif %}

## What to Evaluate

### Selected Action
- **Action Type**: {{ selected_action.action_id }}
- **Purpose**: To resolve the identified bottleneck

### Agent's Selected Parameters
```json
{{ selected_parameters | tojson(indent=2) }}
```

Expected Parameters (Ground Truth)
```json
{{ expected_parameters | tojson(indent=2) }}
```

Evaluation Framework
```

```
### Step 1: Understand Parameter Intent
For each parameter in the expected set, identify:
1. **Purpose**: What this parameter accomplishes
2. **Criticality**: Is it essential for resolving the bottleneck?
3. **Flexibility**: Can alternatives achieve the same goal?

### Step 2: Map Parameters Semantically
Compare agent's parameters to expected parameters:
1. **Direct matches**: Same parameter name and equivalent value
2. **Semantic matches**: Different representation, same effect
3. **Missing parameters**: Expected but not provided
4. **Extra parameters**: Provided but not expected
5. **Wrong parameters**: Provided but incorrect for the goal

### Step 3: Evaluate Effectiveness
Ask: "Would the agent's parameters successfully resolve the bottleneck?"

## Scoring Rubric

### CORRECT (Score: 1.0)
All of the following must be true:
- âIJŞ All critical parameters are present (directly or semantically)
- âIJŞ Parameter values would achieve the bottleneck resolution
- âIJŞ Any deviations are reasonable improvements or valid alternatives
- âIJŞ No critical information is wrong or missing
- âIJŞ Extra parameters (if any) don't interfere with the goal

### PARTIALLY_CORRECT (Score: 0.5)
The parameters show understanding but have gaps:
- âŽñ Most critical parameters present (70-90%)
- âŽñ Would partially resolve the bottleneck
- âŽñ Missing some important details (timing, specific people, etc.)
- âŽñ Some parameter values are suboptimal but not wrong
- âŽñ May include unnecessary parameters that don't harm

### INCORRECT (Score: 0.0)
Major failures in parameter selection:
- âIJŮ Missing most critical parameters
- âIJŮ Wrong people, systems, or resources specified
- âIJŮ Parameters would not resolve the bottleneck
- âIJŮ Fundamental misunderstanding of what's needed
- âIJŮ Parameters might make the situation worse

## Calibration Examples

### Example 1: CORRECT - Semantic Equivalence
**Bottleneck**: "Rachel needs budget approval from CFO Tom Bradley for Q4 marketing campaign by October
     1st"

**Expected Parameters**:
'''json
{
  "to": ["tom.bradley@company.com"],
  "subject": "Q4 Marketing Budget Approval Request",
  "body": "Request for $50K marketing budget approval",
  "priority": "high"
}
'''

**Agent Selected**:
'''json
{
  "to": ["tom.bradley@company.com"],
  "subject": "Urgent: Q4 Marketing Budget - Approval Needed by Oct 1",
  "body": "Hi Tom, I need approval for the Q4 marketing budget ($50K) to proceed with the campaign.
     Deadline is October 1st.",
  "priority": "high"
}
'''

**Analysis**:
- All critical elements present (recipient, urgency, amount, deadline)
- More detailed subject line improves clarity
- Body includes deadline context
- Would successfully resolve the bottleneck

**Judgment**: CORRECT

### Example 2: PARTIALLY_CORRECT - Missing Key Details
**Bottleneck**: "Project Alpha delayed because Lisa Chen hasn't reviewed technical specifications in
     JIRA ticket ALPHA-234"

**Expected Parameters**:
'''json
{
  "assignee": "lisa.chen",
  "ticket_id": "ALPHA-234",
  "comment": "Hi Lisa, please review the technical specs. This is blocking Project Alpha.",
  "due_date": "2024-03-20",
  "priority": "critical"
}
'''
```

```
**Agent Selected**:
```json
{
  "assignee": "lisa.chen",
  "comment": "Please review the technical specifications as soon as possible.",
  "priority": "high"
}
```

Analysis:
- Correct person assigned
- Missing critical ticket_id (ALPHA-234)
- No due date specified
- Priority close but not "critical"
- Generic message lacks context

Judgment: PARTIALLY_CORRECT - Would reach right person but lacks specificity

Example 3: INCORRECT - Wrong Approach
Bottleneck: "Sales team can't access new CRM because IT hasn't completed Active Directory group
     setup"

Expected Parameters:
```json
{
  "ticket_type": "access_request",
  "group_name": "CRM_Sales_Users",
  "members": ["sales-team@company.com"],
  "system": "Salesforce",
  "urgency": "immediate"
}
```

Agent Selected:
```json
{
  "to": ["sales-team@company.com"],
  "subject": "CRM Access Information",
  "body": "The new CRM system will be available soon. Please wait for IT to complete setup."
}
```

Analysis:
- Completely wrong action type (email vs access request)
- Doesn't actually request the AD group setup
- Informs sales team instead of resolving with IT
- Would not resolve the bottleneck

Judgment: INCORRECT - Misunderstands the required action

Parameter Evaluation Guidelines

Consider Valid Variations
- **Email addresses**: "john@company.com" vs "John Smith <john@company.com>"
- **Dates**: "March 23, 2024" vs "2024-03-23" vs "next Friday"
- **Priority**: "high" vs "urgent" vs "critical" (if contextually similar)
- **Lists**: Order rarely matters unless sequence is critical

Critical vs Optional Parameters
Identify which parameters are:
- **Essential**: Must be present for action to work
- **Important**: Significantly impact effectiveness
- **Optional**: Nice to have but not required
- **Contextual**: Depend on specific situation

Common Pitfalls to Avoid
1. **Over-penalizing format differences**: JSON structure vs semantic meaning
2. **Ignoring context**: Parameters should fit the specific bottleneck
3. **Requiring exact matches**: "ASAP" vs "urgent" may be equivalent
4. **Missing parameter relationships**: Some parameters depend on others

Output Instructions

Analyze systematically, then provide your judgment in this JSON format:

```json
{
  "judgment": "<CORRECT|PARTIALLY_CORRECT|INCORRECT>",
  "reasoning": "<2-3 sentences explaining how the parameters would or wouldn't resolve the bottleneck>",
  "parameter_analysis": {
    "critical_parameters_met": <true|false>,
    "would_resolve_bottleneck": "<yes|partially|no>",
    "missing_parameters": ["<list any critical missing params>"],
    "incorrect_parameters": ["<list any wrong params>"],
    "semantic_matches": ["<list params that match semantically>"]
  },
  "confidence": <0.0-1.0>
}
```
```

```
Remember: Focus on whether the parameters would effectively resolve the specific bottleneck in this
      context.
```

Listing 25: judge_action_parameter_scoring.j2

C NOTE ON THE USE OF LANGUAGE MODELS

We utilized Claude (Anthropic) as an AI writing assistant throughout the preparation of this manuscript. Claude was employed primarily for refining sentence clarity, improving paragraph flow, and ensuring consistency in academic writing style. All scientific content, experimental design, analysis, and intellectual contributions remain solely those of the authors.